dit Documentation

Release 1.0

dit Contributors

November 08, 2013

Contents

`dit` is *the* python module for all your discrete information theory needs.

Contents:

CHAPTER 1

Basic Usage

The basic usage of `dit` corresponds to creating distributions, modifying them if need be, and then computing properties of those distributions. For example:

```
>>> from dit.example_dists import Xor
>>> from dit.algorithms import entropy
>>> d = Xor()
>>> print(d)
Class:          Distribution
Alphabet:       ('0', '1') for all rvs
Base:           linear
Outcome Class:  str
Outcome Length: 3
RV Names:       None

x     p(x)
000   0.25
011   0.25
101   0.25
110   0.25
>>> print(entropy(d))
2.0
```

Here, we imported an example distribution constructor (that of the logical exclusive or) and the entropy function. Then we instantiated the XOR distribution, printed it, and computed its entropy.

Notation

dit is a scientific tool, and so, much of this documentation will contain mathematical expressions. Here we will describe this notation.

2.1 Basic Notation

Many distributions are *joint* distribution. In the absence of variable names, we index each random variable with a subscript. For example, a distribution over three variables is written $X_0 X_1 X_2$. As a shorthand, we also denote those random variables as $X_{0:3}$, meaning start with X_0 and go through, but not including X_3 — just like python slice notation.

If we ever need to describe an infinitely long chain of variables we drop the index from the side that is infinite. So $X_{:0} = \ldots X_{-3} X_{-2} X_{-1}$ and $X_{0:} = X_0 X_1 X_2 \ldots$. For an arbitrary set of indices A, the corresponding collection of random variables is denoted X_A. For example, if $A = \{0, 2, 4\}$, then $X_A = X_0 X_2 X_4$. The complement of A (with respect to some universal set) is denoted $\bar{A}$.

2.2 Advanced Notation

When there exists a function $Y = f(X)$ we write $X \succeq Y$ meaning that X is *informationally richer* than Y. Similarly, if $f(Y) = X$ then we write $X \preceq Y$ and say that X is *informationally poorer* than Y. Of all the variables that are poorer than both X and Y, there is a richest one. This variable is known as the *meet* of X and Y and is denoted $X \curlywedge Y$. By definition, $\forall Z s.t. Z \preceq X$ and $Z \preceq Y, Z \preceq X \curlywedge Y$. Similarly of all variables richer than both X and Y, there is a poorest. This variable is known as the *join* of X and Y and is denoted $X \curlyvee Y$. The joint random variable (X, Y) is equivalent to the join.

CHAPTER 3

Distributions

Here we describe how to create, modify, and manipulate distribution objects.

3.1 Numpy-based ScalarDistribution

ScalarDistribution specific stuff.

3.2 Numpy-based Distribution

Distribution specific stuff.

Information Measures

`dit` supports many information measures, ranging from as standard as the Shannon entropy to as exotic as Gács-Körner common information (with even more esoteric measure coming soon!). We organize these quantities into the following groups.

We first have the Shannon-like measures. These quantities are based on sums and differences of entropies, conditional entropies, or mutual informations of random variables:

4.1 Basic Shannon measures

The information on this page is drawn from [3].

4.1.1 Entropy

The entropy measures how much information is in a random variable X.

$$\mathrm{H}[X] = -\sum_{x \in X} p(x) \log_2 p(x)$$

`entropy` (*dist*, *rvs=None*, *rv_names=None*)

Returns the entropy H[X] over the random variables in *rvs*.

If the distribution represents linear probabilities, then the entropy is calculated with units of ‘bits’ (base-2).

Parameters

- **dist** (*Distribution or float*) – The distribution from which the entropy is calculated. If a float, then we calculate the binary entropy.
- **rvs** (*list, None*) – The indexes of the random variable used to calculate the entropy. If None, then the entropy is calculated over all random variables. This should remain *None* for ScalarDistributions.
- **rv_names** (*bool*) – If *True*, then the elements of *rvs* are treated as random variable names. If *False*, then the elements of *rvs* are treated as random variable indexes. If *None*, then the value *True* is used if the distribution has specified names for its random variables.

Returns **H** (*float*) – The entropy of the distribution.

4.1.2 Conditional Entropy

The conditional entropy is the amount of information in variable X beyond that which is in variable Y.

$$\mathrm{H}[X|Y] = \sum_{x \in X, y \in Y} p(x,y) \log_2 p(x|y)$$

conditional_entropy(*dist*, *rvs_X*, *rvs_Y*, *rv_names=None*)

Returns the conditional entropy of H[X|Y].

If the distribution represents linear probabilities, then the entropy is calculated with units of 'bits' (base-2).

Parameters

- **dist** (*Distribution*) – The distribution from which the conditional entropy is calculated.
- **rvs_X** (*list, None*) – The indexes of the random variables defining X.
- **rvs_Y** (*list, None*) – The indexes of the random variables defining Y.
- **rv_names** (*bool*) – If *True*, then the elements of *rvs_X* and *rvs_Y* are treated as random variable names. If *False*, then their elements are treated as random variable indexes. If *None*, then the value *True* is used if the distribution has specified names for its random variables.

Returns **H_XgY** (*float*) – The conditional entropy H[X|Y].

4.1.3 Mutual Information

The mutual information is the amount of information shared by X and Y.

$$\begin{aligned}\mathrm{I}[X:Y] &= \mathrm{H}[X,Y] - \mathrm{H}[X|Y] - \mathrm{H}[Y|X] \\ &= \mathrm{H}[X] + \mathrm{H}[Y] - \mathrm{H}[X,Y] \\ &= \sum_{x \in X, y \in Y} p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)}\end{aligned}$$

mutual_information(*dist*, *rvs_X*, *rvs_Y*, *rv_names=None*)

Returns the mutual information I[X:Y].

If the distribution represents linear probabilities, then the entropy is calculated with units of 'bits' (base-2).

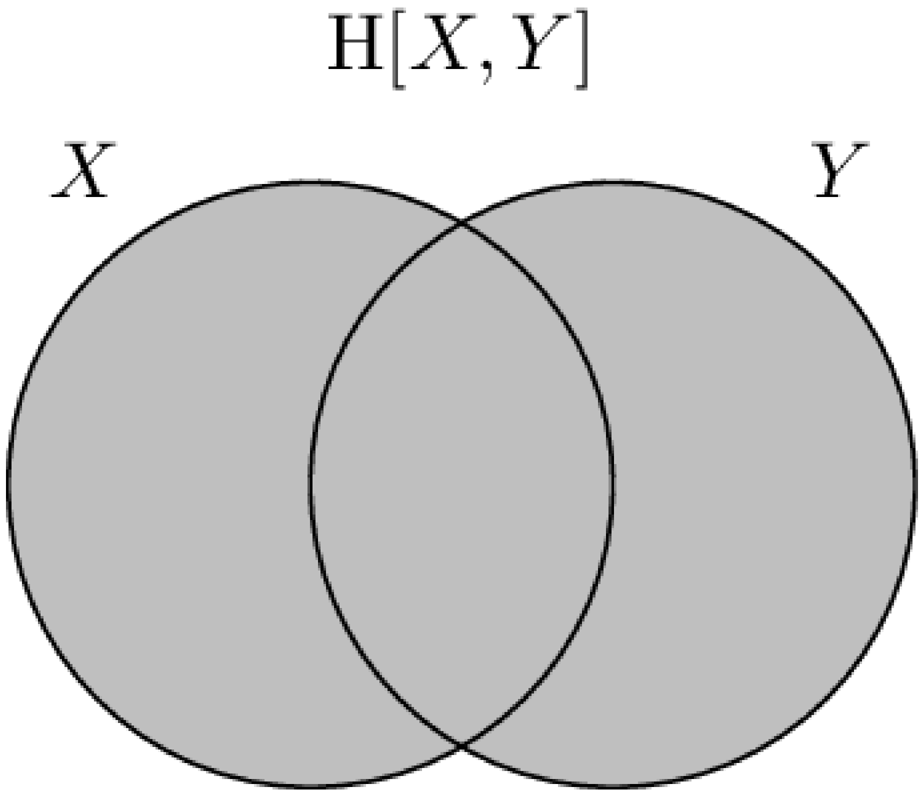
Parameters

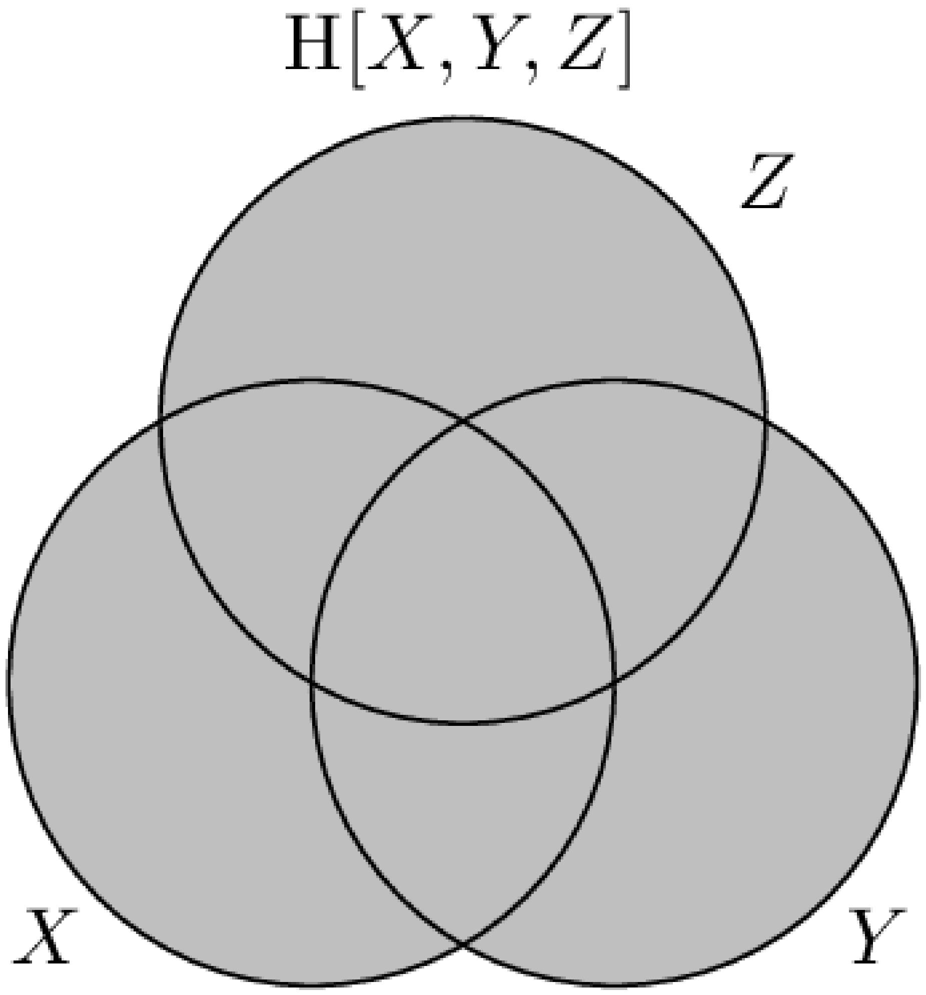
- **dist** (*Distribution*) – The distribution from which the mutual information is calculated.
- **rvs_X** (*list, None*) – The indexes of the random variables defining X.
- **rvs_Y** (*list, None*) – The indexes of the random variables defining Y.
- **rv_names** (*bool*) – If *True*, then the elements of *rvs_X* and *rvs_Y* are treated as random variable names. If *False*, then their elements are treated as random variable indexes. If *None*, then the value *True* is used if the distribution has specified names for its random variables.

Returns **I** (*float*) – The mutual information I[X:Y].

4.2 Entropy

This is a general entropy function, handling conditional joint entropy.





entropy2 (*dist*, *rvs=None*, *crvs=None*, *rv_names=None*)
Compute the conditional joint entropy.

Parameters

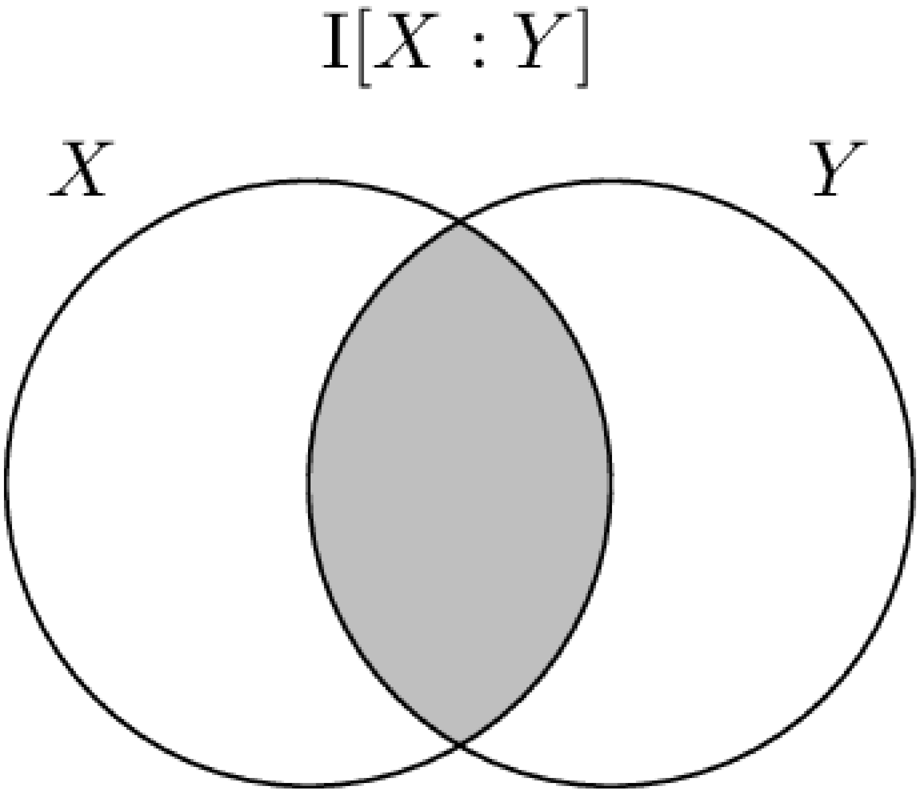
- **dist** (*Distribution*) – The distribution from which the entropy is calculated.
- **rvs** (*list, None*) – The indexes of the random variable used to calculate the entropy. If None, then the entropy is calculated over all random variables.
- **crvs** (*list, None*) – The indexes of the random variables to condition on. If None, then no variables are condition on.
- **rv_names** (*bool*) – If *True*, then the elements of *rvs* are treated as random variable names. If *False*, then the elements of *rvs* are treated as random variable indexes. If *None*, then the value *True* is used if the distribution has specified names for its random variables.

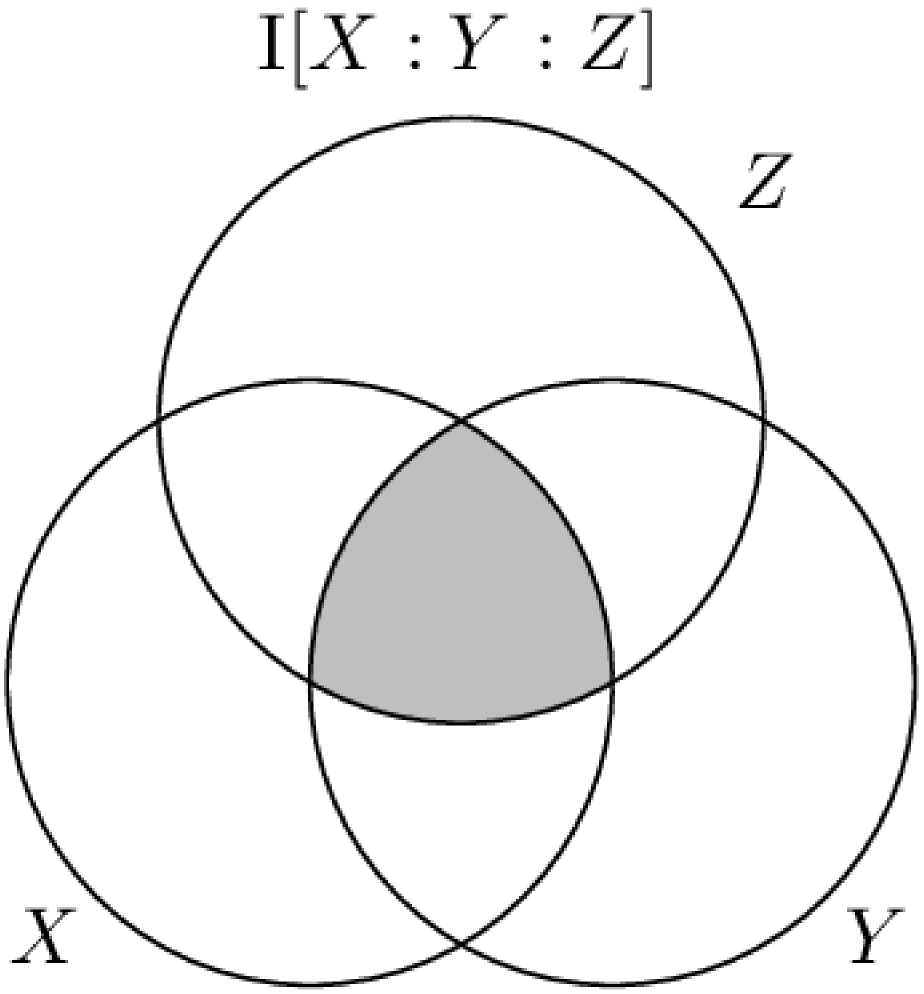
Returns **H** (*float*) – The entropy.

4.3 Co-Information

The co-information [2] is one generalization of the mutual information to multiple variables. It is defined via an inclusion/exclusion sum:

$$
\begin{aligned}
\mathrm{I}[X_{0:n}] &= -\sum_{y \in \mathcal{P}(\{0..n\})} (-1)^{|y|} \, \mathrm{H}[X_y] \\
&= \sum_{x_{0:n} \in X_{0:n}} p(x_{0:n}) \log_2 \prod_{y \in \mathcal{P}(\{0..n\})} p(y)^{(-1)^{|y|}}
\end{aligned}
$$





coinformation (*dist*, *rvs=None*, *crvs=None*, *rv_names=None*)
Calculates the coinformation.

Parameters

- **dist** (*Distribution*) – The distribution from which the coinformation is calculated.
- **rvs** (*list, None*) – The indexes of the random variable used to calculate the coinformation between. If None, then the coinformation is calculated over all random variables.
- **crvs** (*list, None*) – The indexes of the random variables to condition on. If None, then no variables are condition on.
- **rv_names** (*bool*) – If *True*, then the elements of *rvs* are treated as random variable names. If *False*, then the elements of *rvs* are treated as random variable indexes. If *None*, then the value *True* is used if the distribution has specified names for its random variables.

Returns **I** (*float*) – The coinformation.

Raises `ditException` – Raised if *dist* is not a joint distribution.

4.4 Total Correlation

The total correlation [8], denoted T, also known as the multi-information or integration, is one generalization of the mutual information. It is defined as the amount of information each individual variable carries above and beyond the joint entropy:

$$\begin{aligned}\mathrm{T}[X_{0:n}] &= \sum \mathrm{H}[X_i] - \mathrm{H}[X_{0:n}] \\ &= \sum_{x_{0:n} \in X_{0:n}} p(x_{0:n}) \log_2 \frac{p(x_{0:n})}{\prod p(x_i)}\end{aligned}$$

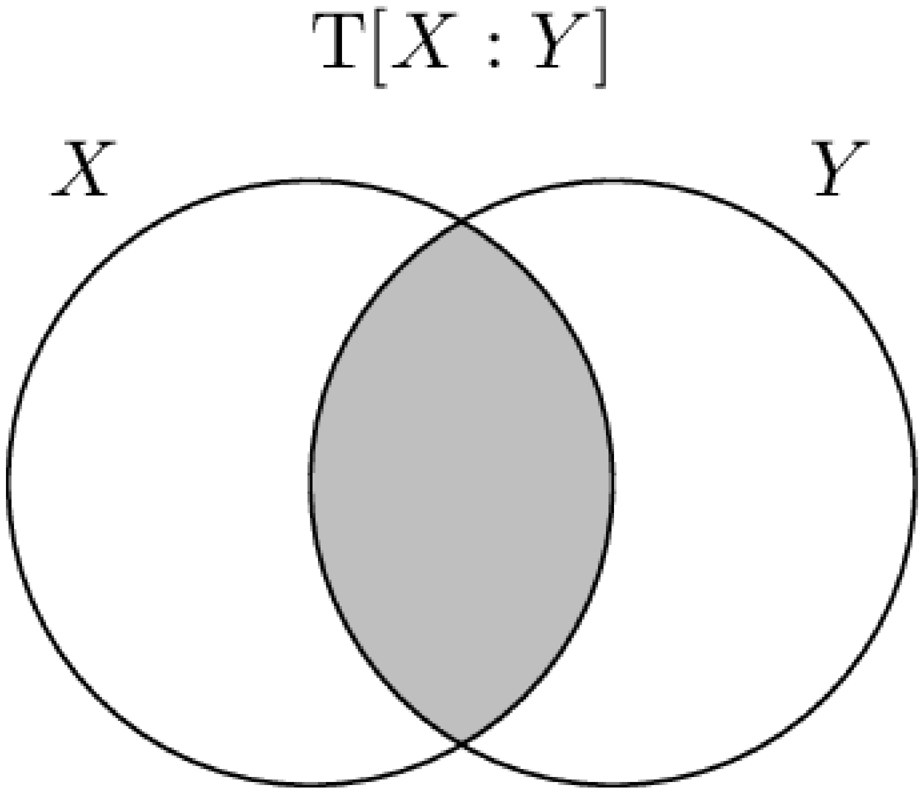
Two nice features of the total correlation are that it is non-negative and that it is zero if and only if the random variables $X_{0:n}$ are all independent. Some baseline behavior is good to note also. First its behavior when applied to "giant bit" distributions:

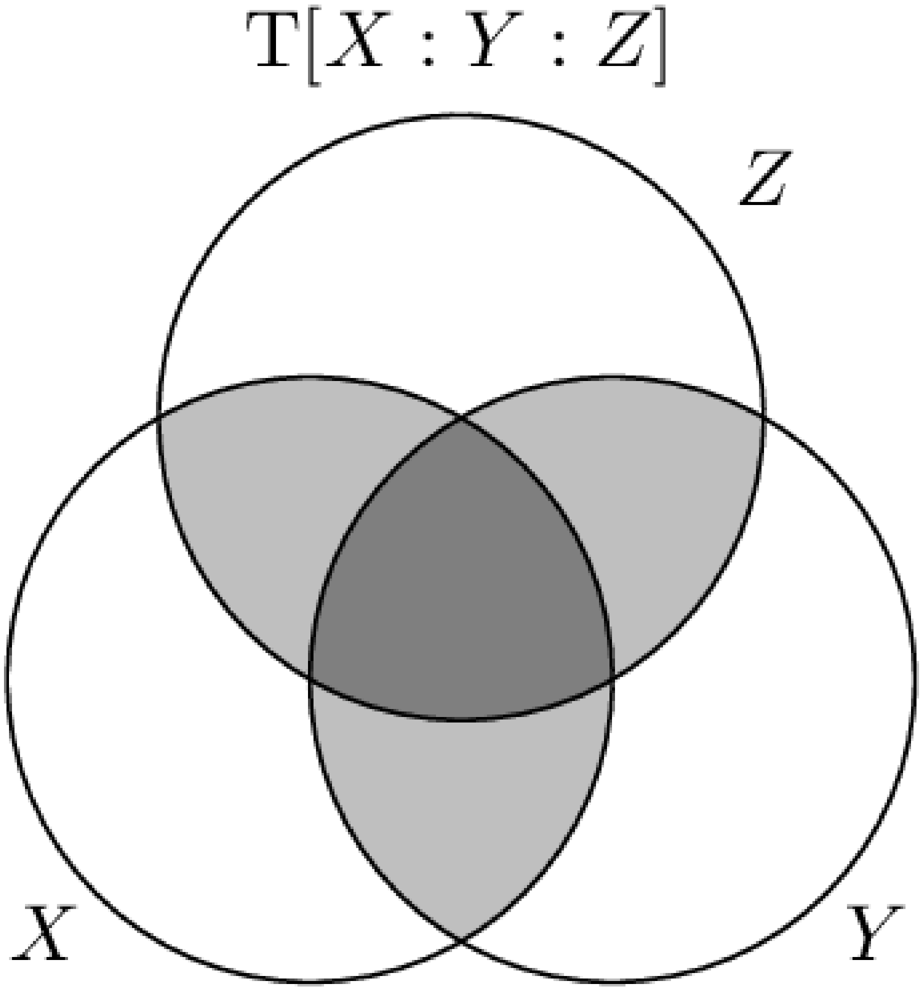
```
>>> from dit import Distribution as D
>>> from dit.algorithms import total_correlation as T
>>> [ T(D(['0'*n, '1'*n], [0.5, 0.5])) for n in range(2, 6) ]
[1.0, 2.0, 3.0, 4.0]
```

So we see that for giant bit distributions, the total correlation is equal to one less than the number of variables. The second type of distribution to consider is general parity distributions:

```
>>> from dit.example_dists import n_mod_m
>>> [ T(n_mod_m(n, 2)) for n in range(3, 6) ]
[1.0, 1.0, 1.0]
>>> [ T(n_mod_m(3, m)) for m in range(2, 5) ]
[1.0, 1.58496250072, 2.0]
```

Here we see that the total correlation is equal to $\log_2 m$ regardless of n.





total_correlation(*dist*, *rvs=None*, *crvs=None*, *rv_names=None*)

Parameters

- **dist** (*Distribution*) – The distribution from which the total correlation is calculated.
- **rvs** (*list, None*) – The indexes of the random variable used to calculate the total correlation. If None, then the total correlation is calculated over all random variables.
- **crvs** (*list, None*) – The indexes of the random variables to condition on. If None, then no variables are condition on.
- **rv_names** (*bool, None*) – If *True*, then the elements of *rvs* are treated as random variable names. If *False*, then the elements of *rvs* are treated as random variable indexes. If *None*, then the value *True* is used if the distribution has specified names for its random variables.

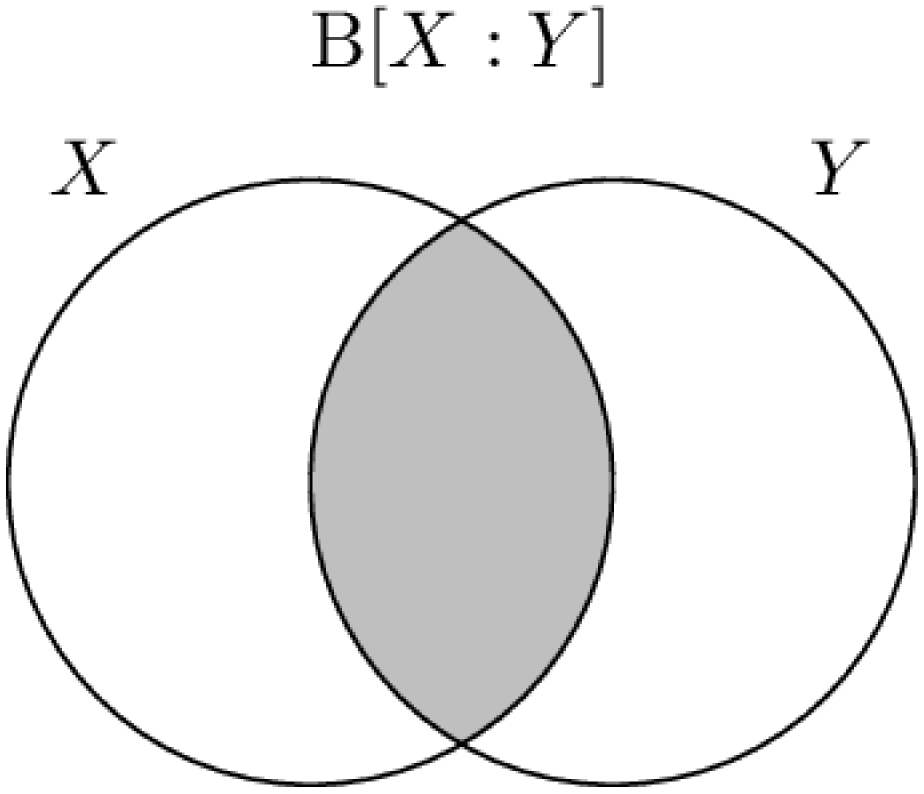
Returns **T** (*float*) – The total correlation

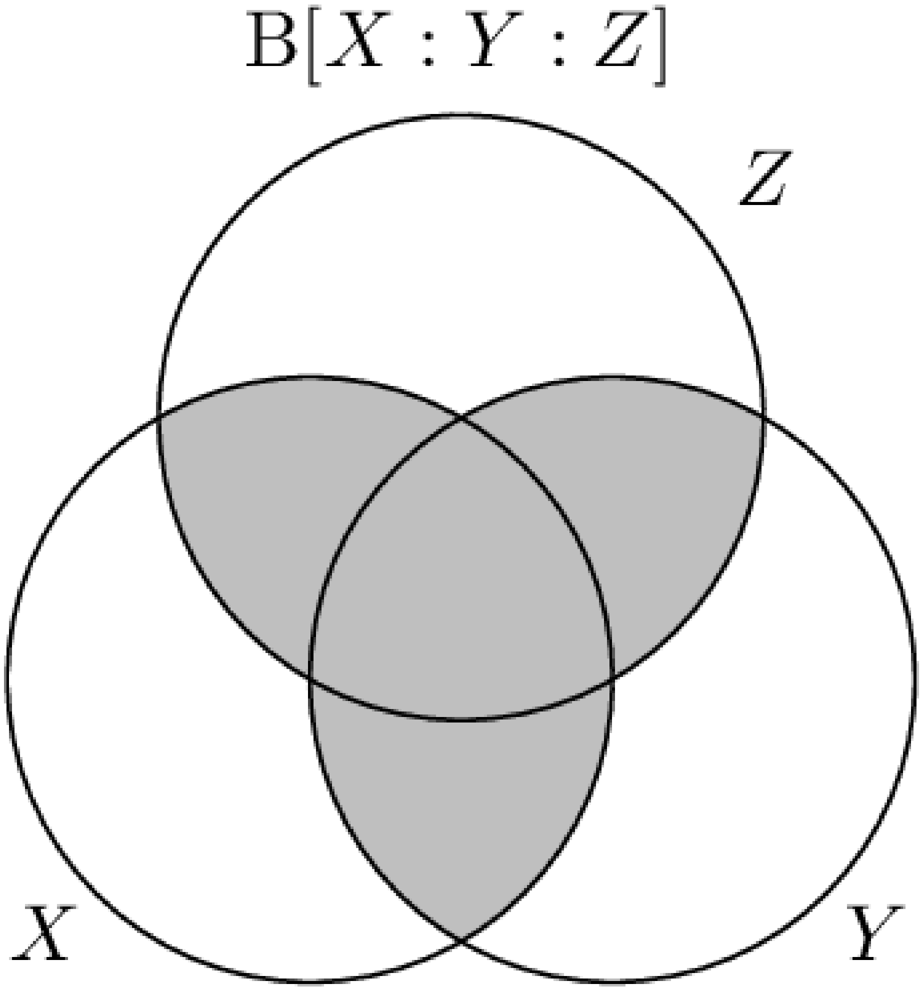
Raises `ditException` – Raised if *dist* is not a joint distribution.

4.5 Binding Information

The binding information [1], or dual total correlation, is yet another generalization of the mutual information. It is defined as:

$$\begin{aligned}\mathrm{B}[X_{0:n}] &= \mathrm{H}[X_{0:n}] - \sum \mathrm{H}[X_i | X_{\{0..n\}/i}] \\ &= - \sum_{x_{0:n} \in X_{0:n}} p(x_{0:n}) \log_2 \frac{p(x_{0:n})}{\prod p(x_i | x_{\{0:n\}/i})}\end{aligned}$$





binding_information (*dist*, *rvs=None*, *crvs=None*, *rv_names=None*)

Parameters

- **dist** (*Distribution*) – The distribution from which the binding information is calculated.
- **rvs** (*list, None*) – The indexes of the random variable used to calculate the binding information. If None, then the binding information is calculated over all random variables.
- **crvs** (*list, None*) – The indexes of the random variables to condition on. If None, then no variables are condition on.
- **rv_names** (*bool*) – If *True*, then the elements of *rvs* are treated as random variable names. If *False*, then the elements of *rvs* are treated as random variable indexes. If *None*, then the value *True* is used if the distribution has specified names for its random variables.

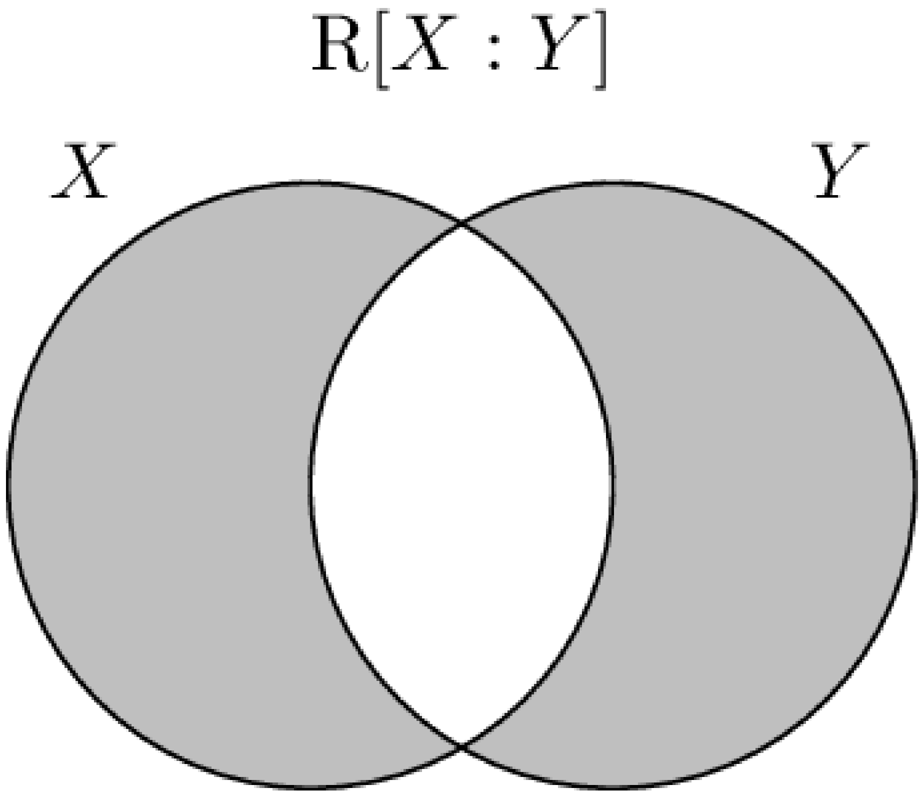
Returns **B** (*float*) – The binding information

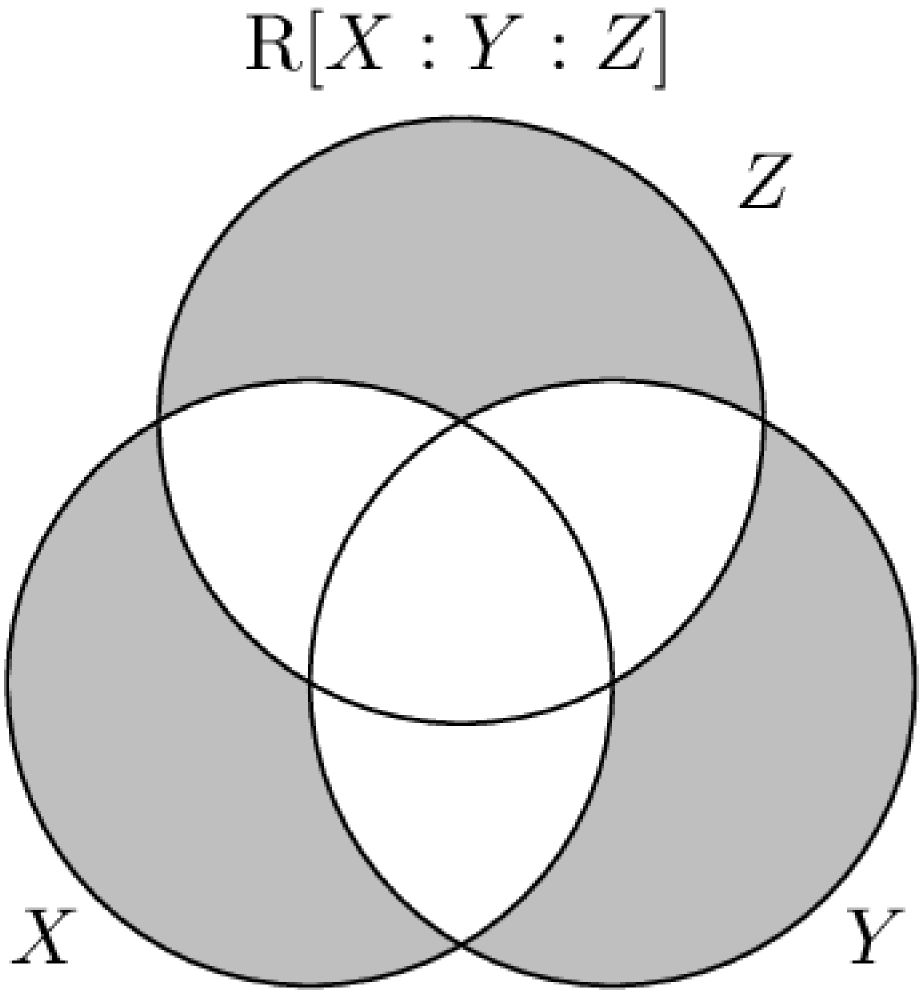
Raises `ditException` – Raised if *dist* is not a joint distribution.

4.6 Residual Entropy

The residual entropy, or erasure entropy, is a dual to the binding information.

$$\begin{aligned} \mathrm{R}[X_{0:n}] &= \sum \mathrm{H}[X_i | X_{\{0..n\}/i}] \\ &= -\sum_{x_{0:n} \in X_{0:n}} p(x_{0:n}) \log_2 \prod p(x_i | x_{\{0:n\}/i}) \end{aligned}$$





residual_entropy(*dist*, *rvs=None*, *crvs=None*, *rv_names=None*)

Parameters

- **dist** (*Distribution*) – The distribution from which the residual entropy is calculated.
- **rvs** (*list, None*) – The indexes of the random variable used to calculate the residual entropy. If None, then the total correlation is calculated over all random variables.
- **crvs** (*list, None*) – The indexes of the random variables to condition on. If None, then no variables are condition on.
- **rv_names** (*bool*) – If *True*, then the elements of *rvs* are treated as random variable names. If *False*, then the elements of *rvs* are treated as random variable indexes. If *None*, then the value *True* is used if the distribution has specified names for its random variables.

Returns **R** (*float*) – The residual entropy

Raises `ditException` – Raised if *dist* is not a joint distribution.

The next group of measures are Shannon-esque measures. These are measure that, while not quite based directly on the canonical Shannon measures like above, they are directly comparable and can be expressed on information-diagrams:

4.7 Interaction Information

The interaction information is equal in magnitude to the co-information, but has the opposite sign when taken over an odd number of variables:

$$\mathrm{II}(X_{0:n}) = (-1)^n \cdot \mathrm{I}(X_{0:n})$$

Interaction information was first studied in the 3-variable case which, for $X_{0:3} = X_0 X_1 X_2$, takes the following form:

$$\mathrm{II}(X_0 : X_1 : X_2) = \mathrm{I}(X_0 : X_1 | X_2) - \mathrm{I}(X_0 : X_1)$$

The extension to $n > 3$ proceeds recursively. For example,

$$\begin{aligned}\mathrm{II}(X_0 : X_1 : X_2 : X_3) &= \mathrm{II}(X_0 : X_1 : X_2 | X_3) - \mathrm{II}(X_0 : X_1 : X_2) \\ &= \mathrm{I}(X_0 : X_1 | X_2, X_3) - \mathrm{I}(X_0 : X_1 | X_3) \\ &\quad - \mathrm{I}(X_0 : X_1 | X_2) + \mathrm{I}(X_0 : X_1)\end{aligned}$$

`interaction_information`(*dist*, *rvs=None*, *crvs=None*, *rv_names=None*)

Calculates the interaction information.

Parameters

- **dist** (*Distribution*) – The distribution from which the interaction information is calculated.
- **rvs** (*list, None*) – The indexes of the random variable used to calculate the interaction information between. If None, then the interaction information is calculated over all random variables.
- **crvs** (*list, None*) – The indexes of the random variables to condition on. If None, then no variables are condition on.
- **rv_names** (*bool*) – If *True*, then the elements of *rvs* are treated as random variable names. If *False*, then the elements of *rvs* are treated as random variable indexes. If *None*, then the value *True* is used if the distribution has specified names for its random variables.

Returns **II** (*float*) – The interaction information.

Raises `ditException` – Raised if *dist* is not a joint distribution.

4.8 Gács-Körner Common Information

The Gács-Körner common information [5] take a very direct approach to the idea of common information. It extracts a random variable that is contained within each of the random variables under consideration.

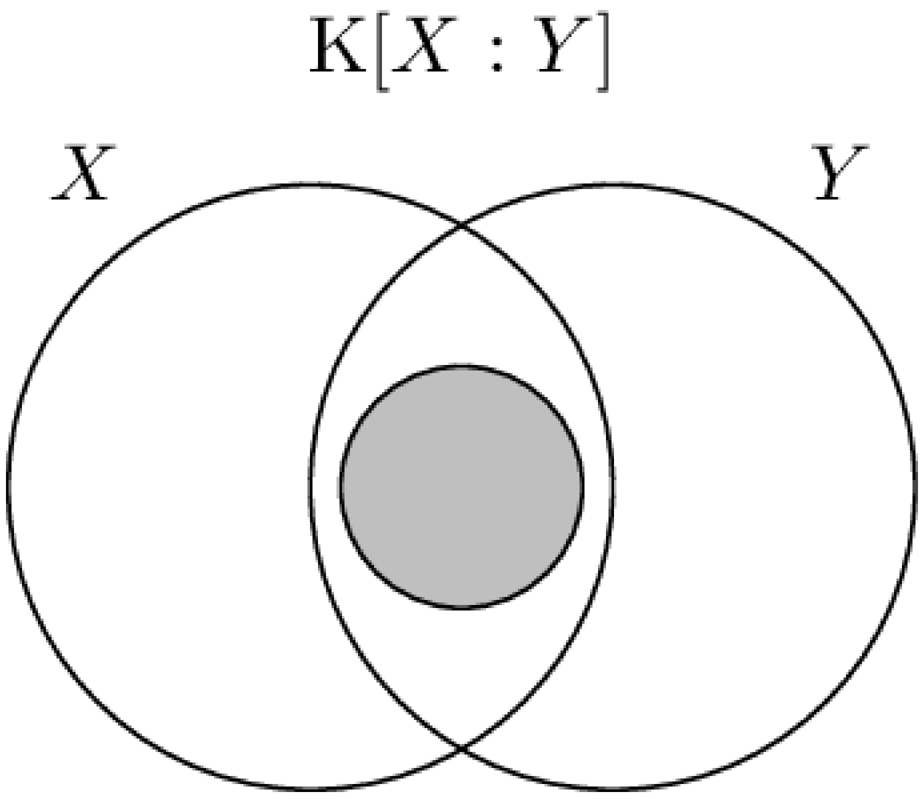
4.8.1 The Common Information Game

Let's play a game. We have an n-variable joint distribution, and one player for each variable. Each player is given the probability mass function of the joint distribution then isolated from each other. Each round of the game the a joint outcome is generated from the distribution and each player is told the symbol that their particular variable took. The goal of the game is for the players to simultaneously write the same symbol on a piece of paper, and for the entropy of the players' symbols to be maximized. They must do this using only their knowledge of the joint random variable and the particular outcome of their marginal variable. The matching symbols produced by the players are called the *common random variable* and the entropy of that variable is the Gács-Körner common information, K.

4.8.2 Two Variables

Consider a joint distribution over X_0 and X_1. Given any particular outcome from that joint, we want a function $f(X_0)$ and a function $g(X_1)$ such that $\forall x_0 x_1 = X_0 X_1, f(x_0) = g(x_1) = v$. Of all possible pairs of functions $f(X_0) = g(X_1) = V$, there exists a "largest" one, and it is known as the common random variable. The entropy of that common random variable is the Gács-Körner common information:

$$\begin{aligned} \mathrm{K}[X_0 : X_1] &= \max_{f(X_0)=g(X_1)=V} \mathrm{H}[V] \\ &= \mathrm{H}[X_0 \curlywedge X_1] \end{aligned}$$



As a canonical example, consider the following:

```
>>> from __future__ import division
>>> from dit import Distribution as D
>>> from dit.algorithms import common_information as K
>>> outcomes = ['00', '01', '10', '11', '22', '33']
>>> pmf = [1/8, 1/8, 1/8, 1/8, 1/4, 1/4]
>>> d = D(outcomes, pmf)
>>> K(d)
1.5
```

So, the Gács-Körner common information is 1.5 bits. But what is the common random variable?

```
>>> from dit.algorithms import insert_meet
>>> crv = insert_meet(d, -1, [[0],[1]])
>>> print(crv)
Class:          Distribution
Alphabet:       (('0', '1', '2', '3'), ('0', '1', '2', '3'), ('2', '0', '1'))
Base:           linear
Outcome Class:  str
Outcome Length: 3
RV Names:       None

x     p(x)
002   0.125
012   0.125
102   0.125
112   0.125
220   0.25
331   0.25
```

Looking at the third index of the outcomes, we see that the common random variable maps 2 to 0 and 3 to 1, maintaining the information from those values. When X_0 or X_1 are either 0 or 1, however, it maps them to 2. This is because f and g must act independently: if x_0 is a 0 or a 1, there is no way to know if x_1 is a 0 or a 1 and vice versa. Therefore we aggregate 0s and 1s into 2.

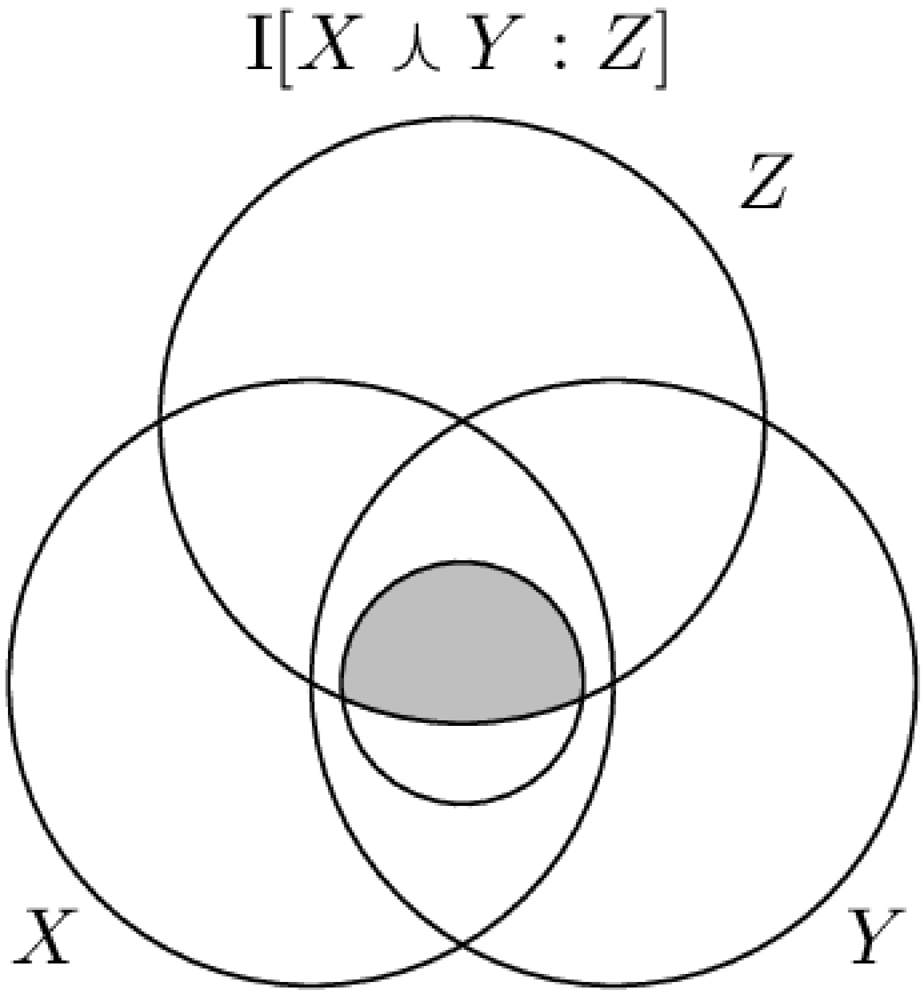
Properties & Uses

The Gács-Körner common information satisfies an important inequality:

$$0 \leq \mathrm{K}[X_0 : X_1] \leq \mathrm{I}[X_0 : X_1]$$

One usage of the common information is as a measure of *redundancy* [4]. Consider a function that takes two inputs, X_0 and X_1, and produces a single output Y. The output can be influenced redundantly by both inputs, uniquely from either one, or together they can synergistically influence the output. Determining how to compute the amount of redundancy is an open problem, but one proposal is:

$$\mathrm{I}[X_0 \curlywedge X_1 : Y]$$



This quantity can be computed easily using dit:

```
>>> from dit.example_dists import RdnXor
>>> from dit.algorithms import insert_meet, mutual_information as I
>>> d = RdnXor()
>>> d = insert_meet(d, -1, [[0], [1]])
>>> I(d, [3], [2])
1.0
```

4.8.3 n-Variables

With an arbitrary number of variables, the Gács-Körner common information is defined similarly:

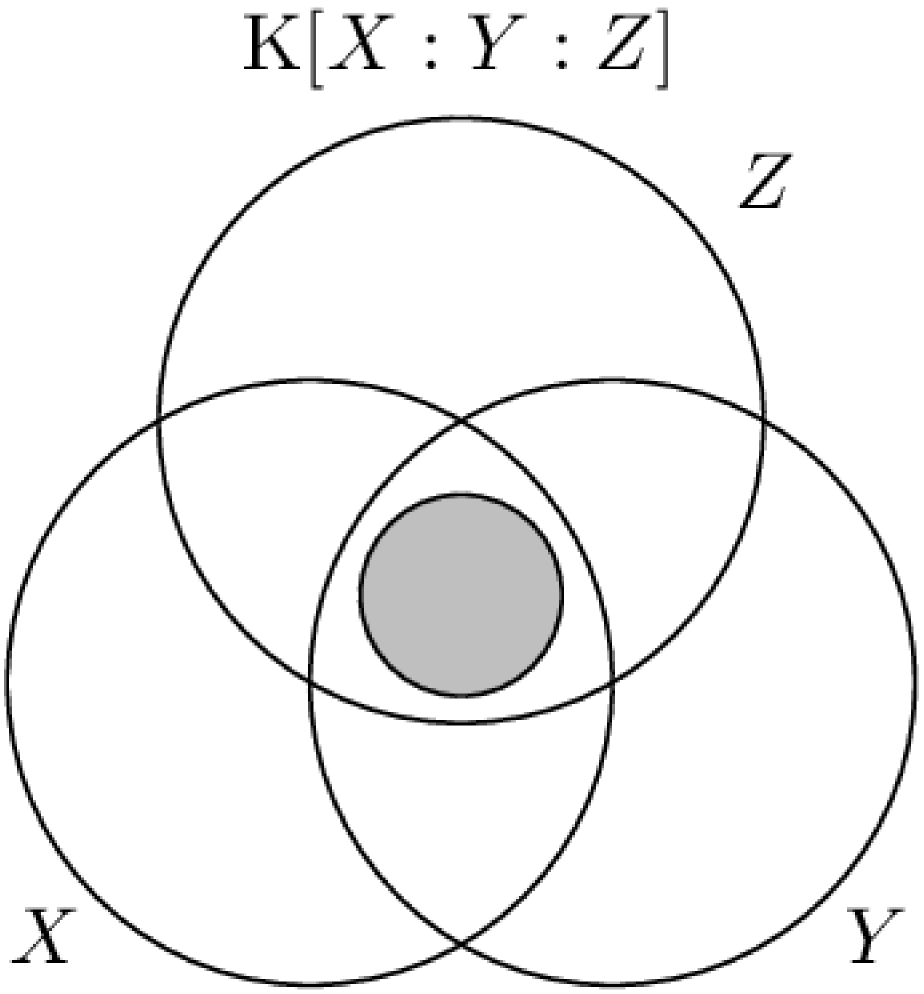
$$\begin{aligned}\mathrm{K}[X_0 : \ldots : X_n] &= \max_{\substack{V=f_0(X_0)\\ \vdots \\ V=f_n(X_n)}} \mathrm{H}[V] \\ &= \mathrm{H}[X_0 \curlywedge \ldots \curlywedge X_n]\end{aligned}$$

The common information is a monotonically decreasing function:

$$\mathrm{K}[X_0 : \ldots : X_{n-1}] \geq \mathrm{K}[X_0 : \ldots : X_n]$$

The multivariate common information follows a similar inequality as the two variate version:

$$0 \leq \mathrm{K}[X_0 : \cdots : X_n] \leq \min_{i,j \in \{0..n\}} \mathrm{I}[X_i : X_j]$$



`common_information` (*dist*, *rvs=None*, *crvs=None*, *rv_names=None*)

Returns the Gacs-Korner common information K[X1:X2...] over the random variables in *rvs*.

Parameters

- **dist** (*Distribution*) – The distribution from which the common information is calculated.
- **rvs** (*list, None*) – The indexes of the random variables for which the Gacs-Korner common information is to be computed. If None, then the common information is calculated over all random variables.
- **crvs** (*list, None*) – The indexes of the random variables to condition the common information by. If none, than there is no conditioning.
- **rv_names** (*bool, None*) – If *True*, then the elements of *rvs* are treated as random variable names. If *False*, then the elements of *rvs* are treated as random variable indexes. If *None*, then the value *True* is used if the distribution has specified names for its random variables.

Returns **K** (*float*) – The Gacs-Korner common information of the distribution.

This next group of measures can not be represented on information diagrams, and can not really be directly compared to the measures above:

4.9 Perplexity

The perplexity is a trivial measure to make the entropy more intuitive.

$$\mathrm{P}[X] = 2^{\mathrm{H}[X]}$$

The perplexity of a random variable is the size of a uniform distribution that would have the same entropy. For example, a distribution with 2 bits of entropy has a perplexity of 4, and so could be said to be "as random" as a four-sided die.

The conditional perplexity is defined in the natural way:

$$\mathrm{P}[X|Y] = 2^{\mathrm{H}[X|Y]}$$

`perplexity`(*dist*, *rvs=None*, *crvs=None*, *rv_names=None*)

Parameters

- **dist** (*Distribution*) – The distribution from which the perplexity is calculated.
- **rvs** (*list, None*) – The indexes of the random variable used to calculate the perplexity. If None, then the perpelxity is calculated over all random variables.
- **crvs** (*list, None*) – The indexes of the random variables to condition on. If None, then no variables are condition on.
- **rv_names** (*bool*) – If *True*, then the elements of *rvs* are treated as random variable names. If *False*, then the elements of *rvs* are treated as random variable indexes. If *None*, then the value *True* is used if the distribution has specified names for its random variables.

Returns **P** (*float*) – The perplexity.

4.10 Extropy

The extropy [7] is a dual to the entropy.

`extropy`(*dist*, *rvs=None*, *rv_names=None*)

Returns the extropy J[X] over the random variables in *rvs*.

If the distribution represents linear probabilities, then the extropy is calculated with units of 'bits' (base-2).

Parameters

- **dist** (*Distribution or float*) – The distribution from which the extropy is calculated. If a float, then we calculate the binary extropy.
- **rvs** (*list, None*) – The indexes of the random variable used to calculate the extropy. If None, then the extropy is calculated over all random variables. This should remain *None* for ScalarDistributions.
- **rv_names** (*bool*) – If *True*, then the elements of *rvs* are treated as random variable names. If *False*, then the elements of *rvs* are treated as random variable indexes. If *None*, then the value *True* is used if the distribution has specified names for its random variables.

Returns **J** (*float*) – The extropy of the distribution.

There are also measures of "distance" or divergence:

4.11 Jensen-Shannon Divergence

The Jensen-Shannon divergence is a principled divergence measure which is always finite.

`jensen_shannon_divergence`(*dists*, *weights=None*)

The Jensen-Shannon Divergence: H(sum(w_i*P_i)) - sum(w_i*H(P_i)).

Parameters

- **dists** (*[Distribution]*) – The distributions, P_i, to take the Jensen-Shannon Divergence of.
- **weights** (*[float], None*) – The weights, w_i, to give the distributions. If None, the weights are assumed to be uniform.

Returns **jsd** (*float*) – The Jensen-Shannon Divergence

Raises

- `ditException` – Raised if there *dists* and *weights* have unequal lengths.
- `InvalidNormalization` – Raised if the weights do not sum to unity.
- `InvalidProbability` – Raised if the weights are not valid probabilities.

CHAPTER 5

References

Bibliography

[1] Samer A Abdallah and Mark D Plumbley. A measure of statistical complexity based on predictive information with application to finite spin systems. *Physics Letters A*, 376(4):275–281, January 2012.

[2] Anthony J Bell. *The Co-Information Lattice*. Springer, New York, ica 2003 edition, 2003.

[3] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, New York, second edition, 2006.

[4] Virgil Griffith, Edwin KP Chong, Ryan G James, Christopher J Ellison, and James P Crutchfield. Intersection information based on common randomness. *arXiv preprint arXiv:1310.1538*, 2013.

[5] Peter Gács and János Körner. Common information is far less than mutual information. *Problems of Control and Information Theory*, 2(2):149–162, 1973.

[6] Te Sun Han. Multiple mutual informations and multiple interactions in frequency data. *Information and Control*, 46(1):26–45, July 1980.

[7] Frank Lad, Giuseppe Sanfilippo, and Gianna Agrò. Extropy: a complementary dual of entropy. *arXiv preprint arXiv:1109.6440*, 2011.

[8] Satosi Watanabe. Information Theoretical Analysis of Multivariate Correlation. *IBM Journal of Research and Development*, 4(1):66–82, January 1960.

Python Module Index