Information Theory, Graphical Models, and Decision Trees

EE376A: Information Theory

Tsachy Weissman

Mutual information

Fundamental Limits of:

Data Transmission (Channel coding theory)

Data Compression (Rate distortion theory)

This lecture:

Investigate the role mutual information plays in machine learning through selected examples

Prediction problem

We observe training data $(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n), X_i \in R^d$ is the feature, and Y_i is the label. We need to construct a prediction algorithm for future feature vectors.

Two general approaches towards prediction:

Decision theoretic approach: first fit a probabilistic model of the joint distribution P_{XY}, and then compute the Bayes rule

Learning theoretic approach: directly construct a prediction $f(X)$ with the aim that $E[L(f(X), Y)]$ is small.

Two approaches

We discuss two approaches that are decision theoretic and learning theoretic, respectively. They both rely on mutual information.

Decision theoretic approach: Tree-Augmented Naïve Bayes (TAN): we assume that the joint distribution $P_{X|Y}$ factorizes as a tree graphical model

Learning theoretic approach: we assume that the function $f(X)$ recursively partitions the feature space using a tree, and we try to find a tree that has small test error

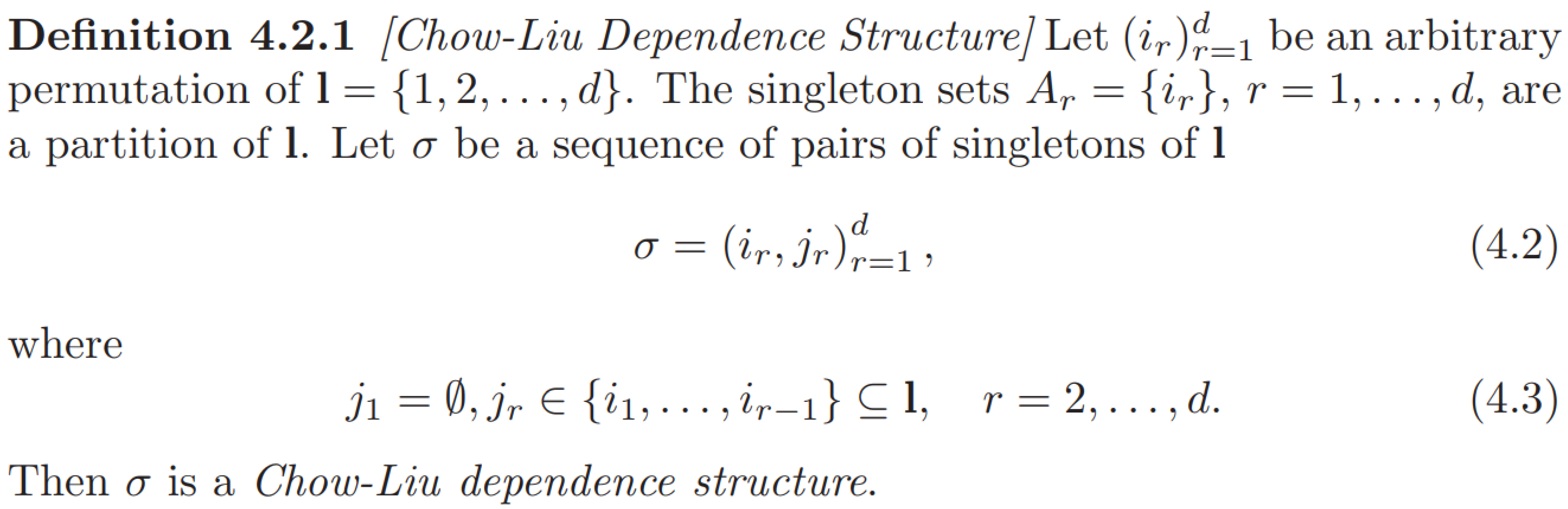
Decision theoretic approach

We reduce the problem to an unsupervised learning problem first:

Say we observe $X_1, X_2, \dots, X_n$ such that the distribution P_X factorizes as a tree. How we can learn P_X?

We do this unsupervised learning algorithm for each label Y.

Tree graphical model structure



Definition 4.2.1 *[Chow-Liu Dependence Structure]* Let $(i_r)_{r=1}^{d}$ be an arbitrary permutation of $\mathbf{I} = \{1, 2, \ldots, d\}$. The singleton sets $A_r = \{i_r\}$, $r = 1, \ldots, d$, are a partition of $\mathbf{I}$. Let σ be a sequence of pairs of singletons of $\mathbf{I}$

$$\sigma = (i_r, j_r)_{r=1}^{d}\,, \tag{4.2}$$

where

$$j_1 = \emptyset, j_r \in \{i_1, \ldots, i_{r-1}\} \subseteq \mathbf{I}, \quad r = 2, \ldots, d. \tag{4.3}$$

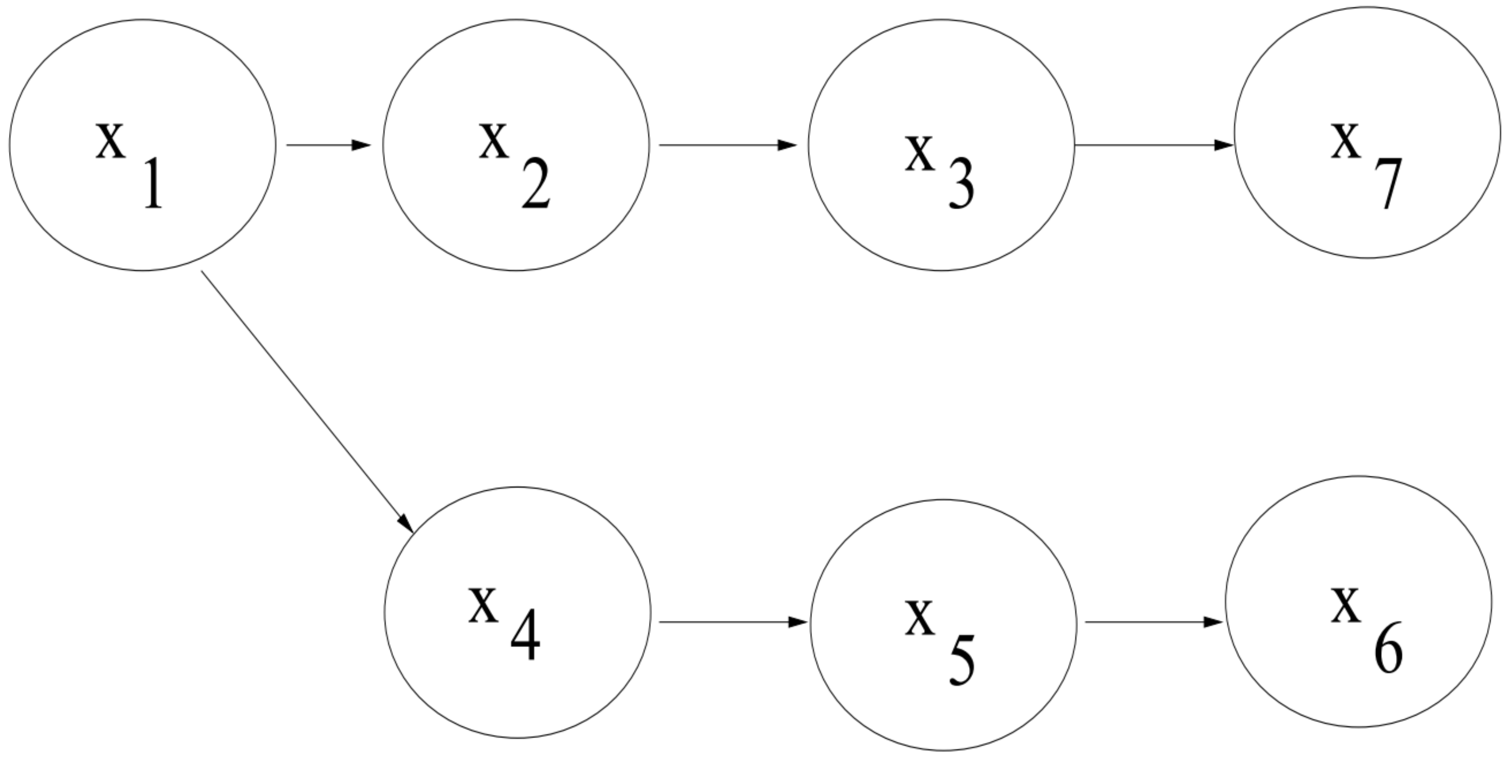
Then σ is a *Chow-Liu dependence structure.*

A comprehensive introduction: https://people.kth.se/~tjtkoski/chowliulect.pdf

Tree graphical model structure

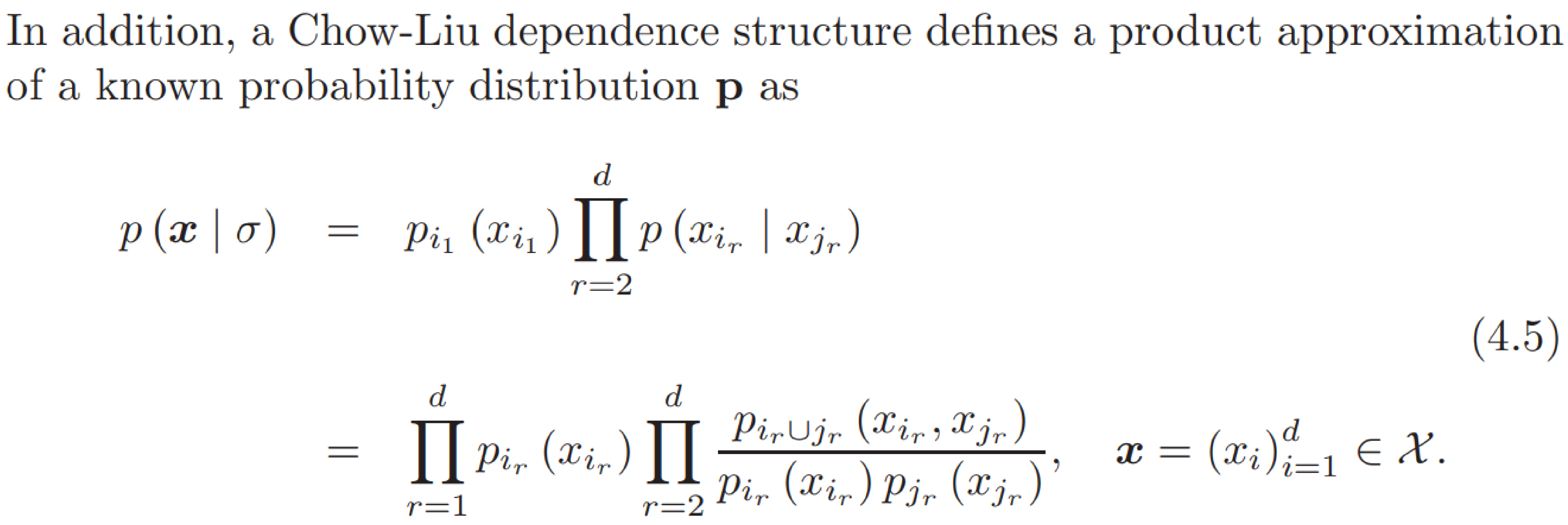
An example of a Chow-Liu dependence structure

$$(j_1, j_2, j_3, j_4, j_5, j_6, j_7) = (\emptyset, 1, 2, 1, 4, 5, 3).$$



A comprehensive introduction: https://people.kth.se/~tjtkoski/chowliulect.pdf

Tree graphical model distribution



In addition, a Chow-Liu dependence structure defines a product approximation of a known probability distribution $\mathbf{p}$ as

$$\begin{aligned} p(\boldsymbol{x} \mid \sigma) &= p_{i_1}(x_{i_1}) \prod_{r=2}^{d} p(x_{i_r} \mid x_{j_r}) \\ &= \prod_{r=1}^{d} p_{i_r}(x_{i_r}) \prod_{r=2}^{d} \frac{p_{i_r \cup j_r}(x_{i_r}, x_{j_r})}{p_{i_r}(x_{i_r})\, p_{j_r}(x_{j_r})}, \quad \boldsymbol{x} = (x_i)_{i=1}^{d} \in \mathcal{X}. \end{aligned} \tag{4.5}$$

Important: this shows that tree can be represented as either directed or undirected graphical models

A comprehensive introduction: https://people.kth.se/~tjtkoski/chowliulect.pdf

Maximum likelihood estimation

Parameter space: $\mathcal{P} = \{p_{i \cup j}(x_i, x_j); (i,j) \in \mathbf{1} \times \mathbf{1}, i \neq j\}$

Likelihood function: $L(\sigma, \mathcal{P}) = \prod_{n=1}^{N} p\left(\boldsymbol{x}^{(n)} \mid \sigma, \mathcal{P}\right)$

For any $i \in \mathbf{1}$ and $\xi \in \mathcal{X}_i$ we introduce

$$I_{\xi,i}\left(\boldsymbol{x}^{(n)}\right) = \begin{cases} 1 & \text{if } \xi = x_i^{(n)} \\ 0 & \text{otherwise.} \end{cases}$$

$$L(\sigma, \mathcal{P}) = \prod_{n=1}^{N} \prod_{\xi \in \mathcal{X}_{i_1}} p_{i_1}\left(x_{i_1}^{(n)}\right)^{I_{\xi,i_1}\left(\boldsymbol{x}^{(n)}\right)} \prod_{r=2}^{d} \prod_{\xi \in \mathcal{X}_{i_r}} \prod_{\eta \in \mathcal{X}_{j_r}} \left[p\left(x_{i_r}^{(n)} \mid x_{j_r}^{(n)}\right)\right]^{I_{\xi,i_r}\left(\boldsymbol{x}^{(n)}\right) I_{\eta,j_r}\left(\boldsymbol{x}^{(n)}\right)}$$

$$= \prod_{\xi \in \mathcal{X}_{i_1}} \prod_{n=1}^{N} p_{i_1}\left(x_{i_1}^{(n)}\right)^{I_{\xi,i_1}\left(\boldsymbol{x}^{(n)}\right)} \prod_{r=2}^{d} \prod_{\xi \in \mathcal{X}_{i_r}} \prod_{\eta \in \mathcal{X}_{j_r}} \left[\prod_{n=1}^{N} p\left(x_{i_r}^{(n)} \mid x_{j_r}^{(n)}\right)\right]^{I_{\xi,i_r}\left(\boldsymbol{x}^{(n)}\right) I_{\eta,j_r}\left(\boldsymbol{x}^{(n)}\right)} .$$

Log likelihood

$$l(\sigma, \mathcal{P}) = \ln L(\sigma, \mathcal{P}) = \frac{1}{N}\sum_{n=1}^{N} \ln p\left(\boldsymbol{x}^{(n)} \mid \sigma, \mathcal{P}\right)$$

$$= \sum_{\xi \in \mathcal{X}_{i_1}} \frac{1}{N}\sum_{n=1}^{N} I_{\xi, i_1}\left(\boldsymbol{x}^{(n)}\right) \ln p_{i_1}\left(x_{i_1}^{(n)}\right)$$

$$+ \sum_{r=2}^{d} \sum_{\xi \in \mathcal{X}_{i_r}} \sum_{\eta \in \mathcal{X}_{j_r}} \frac{1}{N}\sum_{n=1}^{N} I_{\xi, i_r}\left(\boldsymbol{x}^{(n)}\right) I_{\eta, j_r}\left(\boldsymbol{x}^{(n)}\right) \ln p\left(x_{i_r}^{(n)} \mid x_{j_r}^{(n)}\right)$$

$$= \sum_{\xi \in \mathcal{X}_{i_1}} \widehat{p}_{i_1}(\xi) \ln p_{i_1}(\xi) + \sum_{r=2}^{d} \sum_{\eta \in \mathcal{X}_{j_r}} \widehat{p}_{j_r}(\eta) \sum_{\xi \in \mathcal{X}_{i_r}} \frac{\widehat{p}_{i_r j_r}(\xi, \eta)}{\widehat{p}_{j_r}(\eta)} \ln p_{i_r \mid j_r}(\xi \mid \eta)$$

Log likelihood

Non-negativity of KL divergence gives:

$$p_{i_1}^{\mathrm{ML}}(\xi) = \widehat{p}_{i_1}(\xi), \quad \xi \in \mathcal{X}_{i_1},$$

$$p_{i_r|j_r}^{\mathrm{ML}}(\xi \mid \eta) = \frac{\widehat{p}_{i_r j_r}(\xi, \eta)}{\widehat{p}_{j_r}(\eta)}, \quad \xi \in \mathcal{X}_{i_r}, \eta \in \mathcal{X}_{j_r}, r = 2, \ldots, d.$$

Plugging-in the log likelihood:

$$\begin{aligned} l\left(\sigma, \mathcal{P}^{\mathrm{ML}}\right) &= \sum_{\xi \in \mathcal{X}_{i_1}} \widehat{p}_{i_1}(\xi) \ln \widehat{p}_{i_1}(\xi) + \sum_{r=2}^{d} \sum_{\eta \in \mathcal{X}_{j_r}} \sum_{\xi \in \mathcal{X}_{i_r}} \widehat{p}_{i_r j_r}(\xi, \eta) \ln \frac{\widehat{p}_{i_r j_r}(\xi, \eta)}{\widehat{p}_{j_r}(\eta)} \\ &= \sum_{r=1}^{d} \sum_{\xi \in \mathcal{X}_{i_r}} \widehat{p}_{i_r}(\xi) \ln \widehat{p}_{i_r}(\xi) + \sum_{r=2}^{d} \sum_{\eta \in \mathcal{X}_{j_r}} \sum_{\xi \in \mathcal{X}_{i_r}} \widehat{p}_{i_r j_r}(\xi, \eta) \ln \frac{\widehat{p}_{i_r j_r}(\xi, \eta)}{\widehat{p}_{i_r}(\xi)\, \widehat{p}_{j_r}(\eta)} \end{aligned}$$

Log likelihood

Non-negativity of KL divergence gives:

$$p_{i_1}^{\mathrm{ML}}(\xi) = \widehat{p}_{i_1}(\xi), \quad \xi \in \mathcal{X}_{i_1},$$

$$p_{i_r|j_r}^{\mathrm{ML}}(\xi \mid \eta) = \frac{\widehat{p}_{i_r j_r}(\xi, \eta)}{\widehat{p}_{j_r}(\eta)}, \quad \xi \in \mathcal{X}_{i_r}, \eta \in \mathcal{X}_{j_r}, r = 2, \ldots, d.$$

Plugging-in the log likelihood:

$$\begin{aligned}
l\left(\sigma, \mathcal{P}^{\mathrm{ML}}\right) &= \sum_{\xi \in \mathcal{X}_{i_1}} \widehat{p}_{i_1}(\xi) \ln \widehat{p}_{i_1}(\xi) + \sum_{r=2}^{d} \sum_{\eta \in \mathcal{X}_{j_r}} \sum_{\xi \in \mathcal{X}_{i_r}} \widehat{p}_{i_r j_r}(\xi, \eta) \ln \frac{\widehat{p}_{i_r j_r}(\xi, \eta)}{\widehat{p}_{j_r}(\eta)} \\
&= \sum_{r=1}^{d} \sum_{\xi \in \mathcal{X}_{i_r}} \widehat{p}_{i_r}(\xi) \ln \widehat{p}_{i_r}(\xi) + \sum_{r=2}^{d} \sum_{\eta \in \mathcal{X}_{j_r}} \sum_{\xi \in \mathcal{X}_{i_r}} \widehat{p}_{i_r j_r}(\xi, \eta) \ln \frac{\widehat{p}_{i_r j_r}(\xi, \eta)}{\widehat{p}_{i_r}(\xi)\, \widehat{p}_{j_r}(\eta)} \\
&= \sum_{r=1}^{d} \sum_{\xi \in \mathcal{X}_{i_r}} \widehat{p}_{i_r}(\xi) \ln \widehat{p}_{i_r}(\xi) + \sum_{r=2}^{d} \widehat{\mathbf{I}}(i_r, j_r)
\end{aligned}$$

Chow-Liu algorithm (1968)

Clearly, the loglikelihood function in (4.22) is the empirical version, or plug-in estimate, of (4.8). The first term $\sum_{r=1}^{d}\sum_{\xi\in\mathcal{X}_{i_r}}\widehat{p}_{i_r}(\xi)\ln\widehat{p}_{i_r}(\xi)$ does not depend on σ. Hence, we find the maximum likelihood estimate σ^{ML} of the structure σ by

$$\sigma^{\mathrm{ML}} = \operatorname{argmax}_{\sigma}\left\{\sum_{r=2}^{d}\widehat{\mathbf{I}}(i_r, j_r)\right\}. \tag{4.24}$$

The number of trees with d nodes is finite. Hence in principle we could find σ^{ML} by exhaustive search and evaluation of $l\left(\sigma,\mathcal{P}^{\mathrm{ML}}\right)$. Nevertheless, since the number of spanning trees with d nodes is d^{d-2} [85, Cayley′s formula p.82], exhaustive search is infeasible in practice. Hence the second main result of [16] is the observation that there exists a computationally effective way of finding σ^{ML}.

There are well known standard algorithms for finding the maximum weight spanning tree, e.g., the *Kruskal algorithm* or the *Prim algorithm* [1], independently discovered by several others, too, c.f., [48, 79]. The algorithm finds the maximum weight spanning tree in $\mathcal{O}\left(d^2\ln d\right)$ time.

Zooming out

Why should we use maximum likelihood?

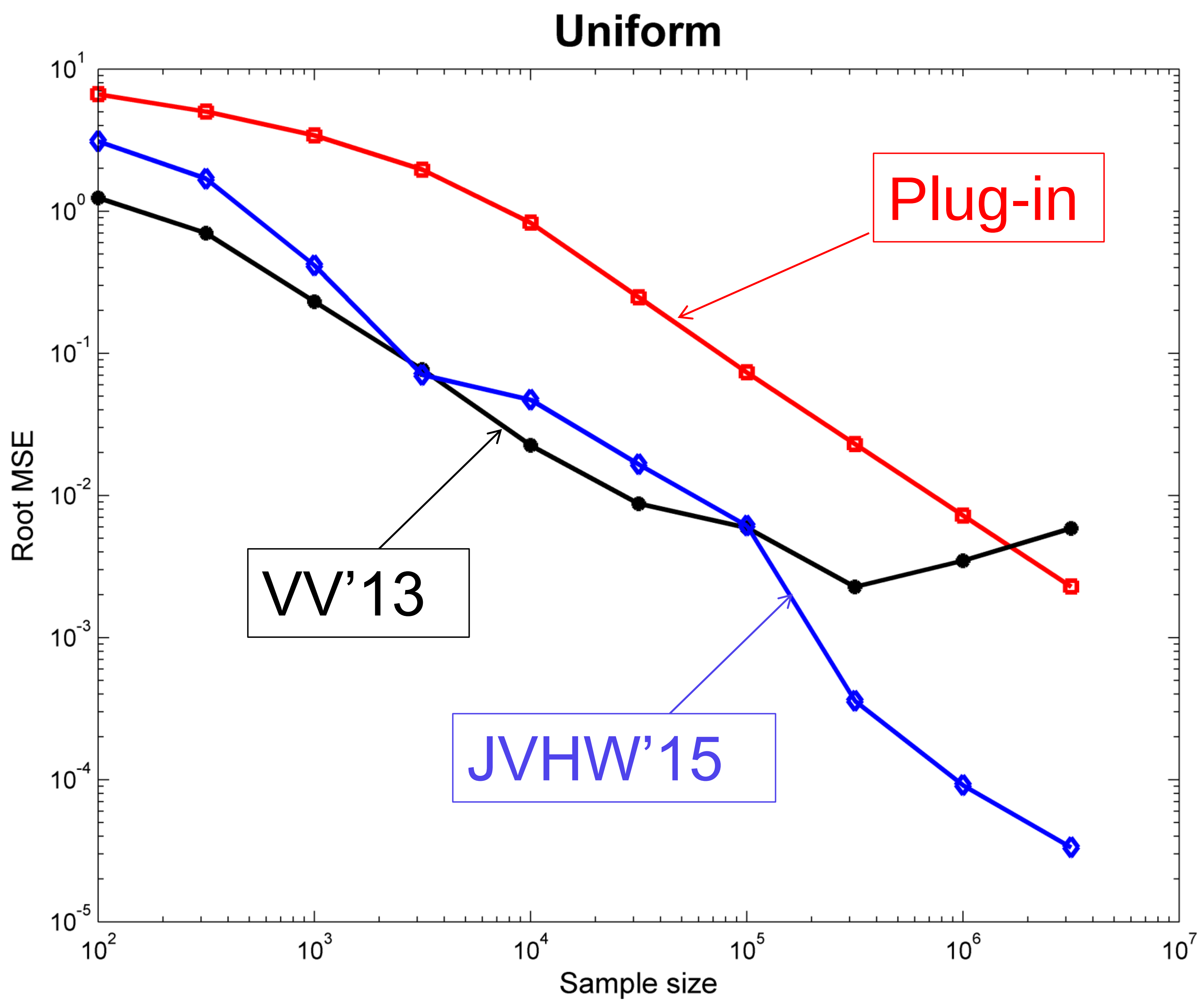
A variational formula

$$\sigma^{\mathrm{True}} = \arg\max_{\sigma} \left\{ \sum_{r=2}^{d} \mathbf{I}(i_r, j_r) \right\}$$

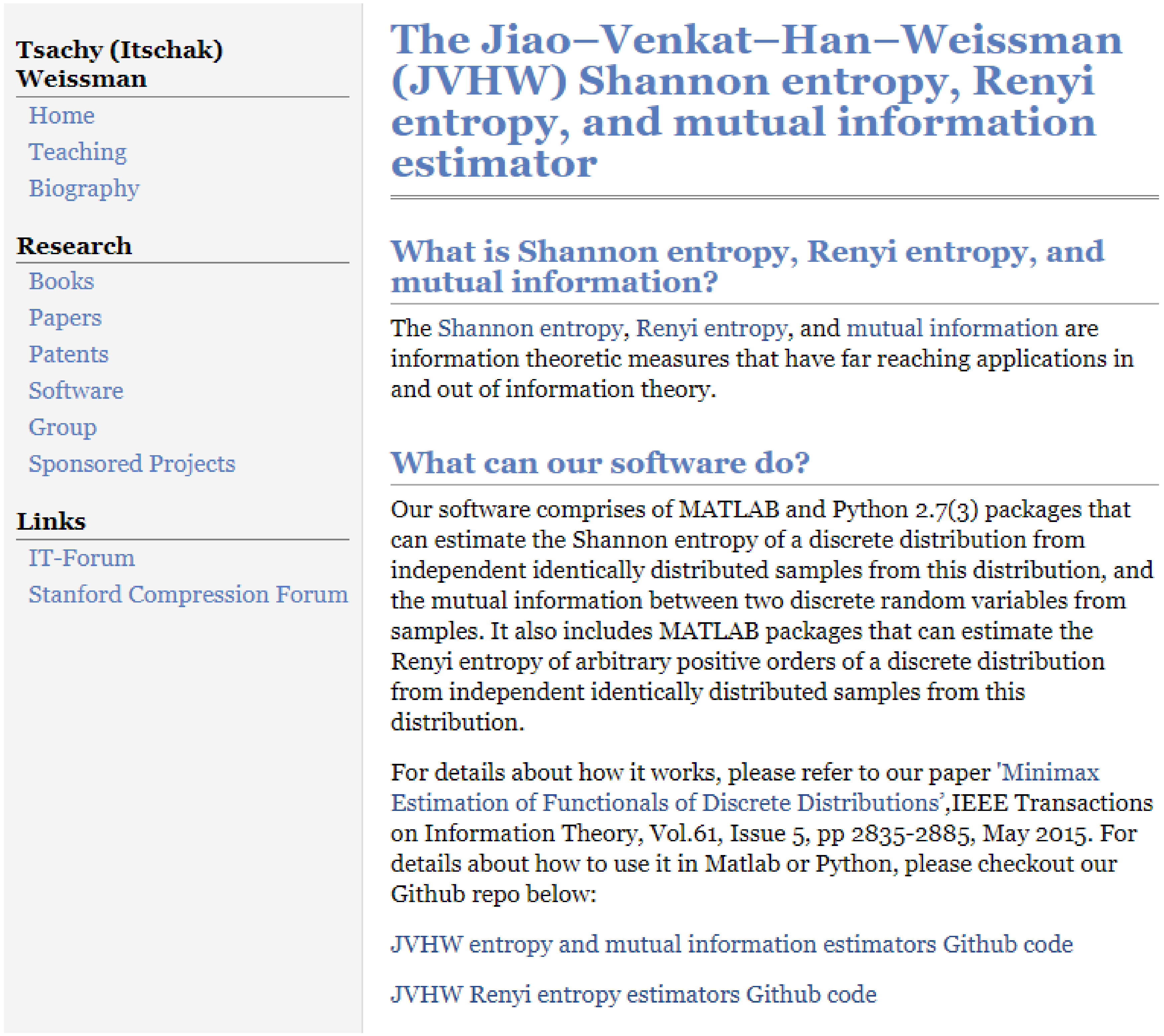
The Chow-Liu algorithm replaces the true mutual information by the empirical mutual information.

Conceptually, what if there exists a much better estimator for mutual information than the empirical one?

Numerical results (alphabet=10^4)



Code online



Tsachy (Itschak) Weissman

- Home
- Teaching
- Biography

Research

- Books
- Papers
- Patents
- Software
- Group
- Sponsored Projects

Links

- IT-Forum
- Stanford Compression Forum

The Jiao–Venkat–Han–Weissman (JVHW) Shannon entropy, Renyi entropy, and mutual information estimator

What is Shannon entropy, Renyi entropy, and mutual information?

The Shannon entropy, Renyi entropy, and mutual information are information theoretic measures that have far reaching applications in and out of information theory.

What can our software do?

Our software comprises of MATLAB and Python 2.7(3) packages that can estimate the Shannon entropy of a discrete distribution from independent identically distributed samples from this distribution, and the mutual information between two discrete random variables from samples. It also includes MATLAB packages that can estimate the Renyi entropy of arbitrary positive orders of a discrete distribution from independent identically distributed samples from this distribution.

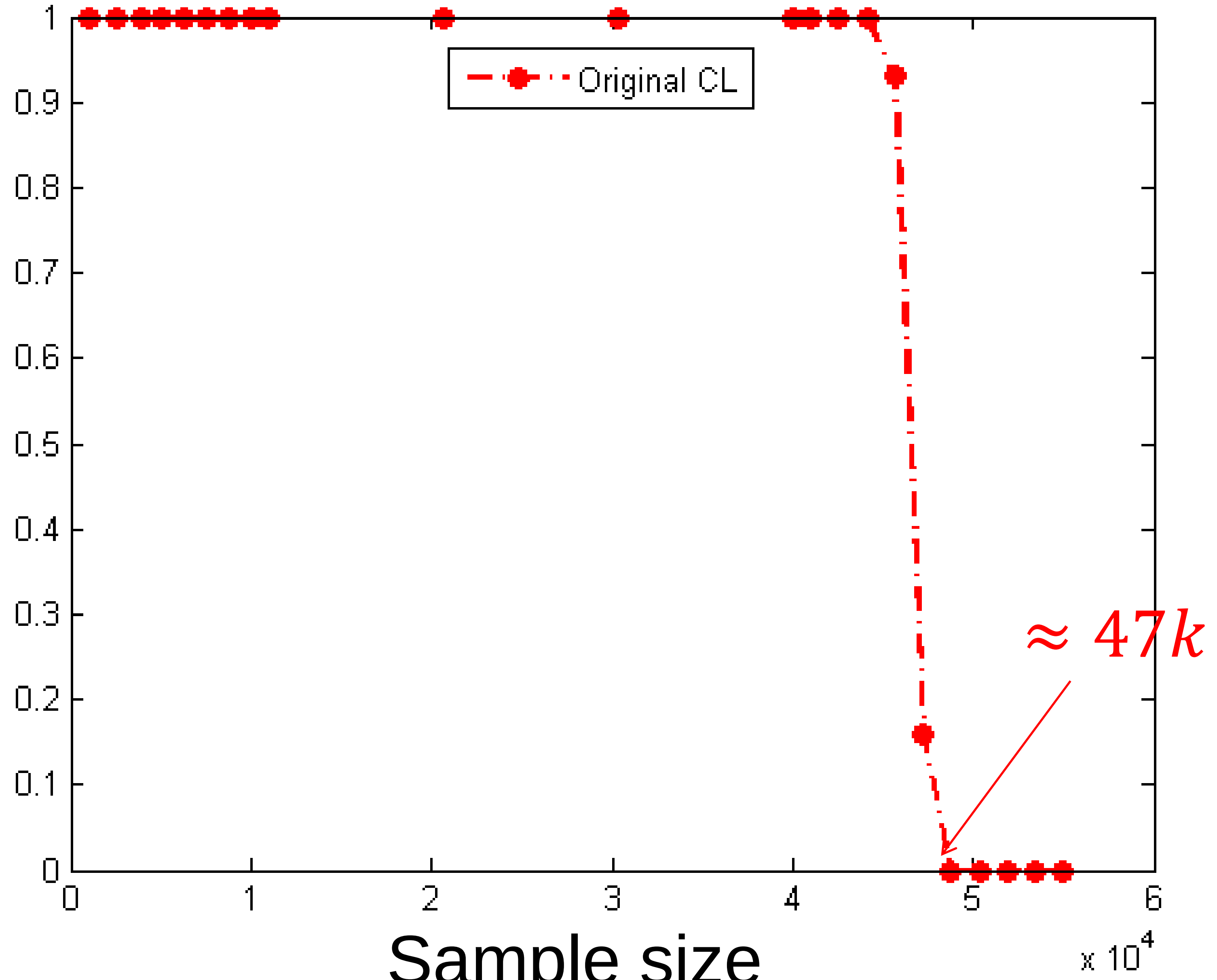
For details about how it works, please refer to our paper 'Minimax Estimation of Functionals of Discrete Distributions',IEEE Transactions on Information Theory, Vol.61, Issue 5, pp 2835-2885, May 2015. For details about how to use it in Matlab or Python, please checkout our Github repo below:

JVHW entropy and mutual information estimators Github code

JVHW Renyi entropy estimators Github code

Chow-Liu algorithm (1968)

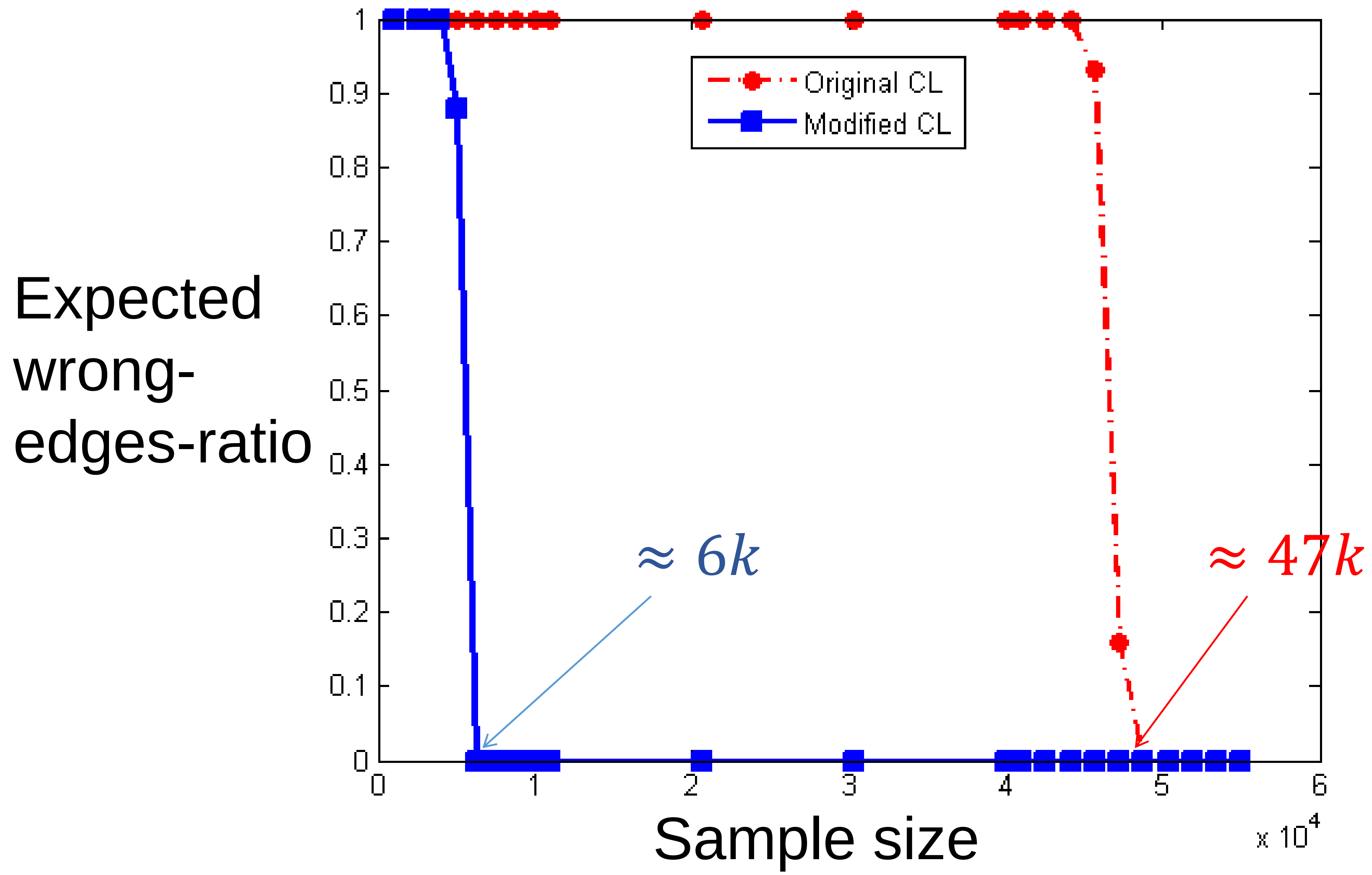
Expected wrong-edges-ratio



Sample size

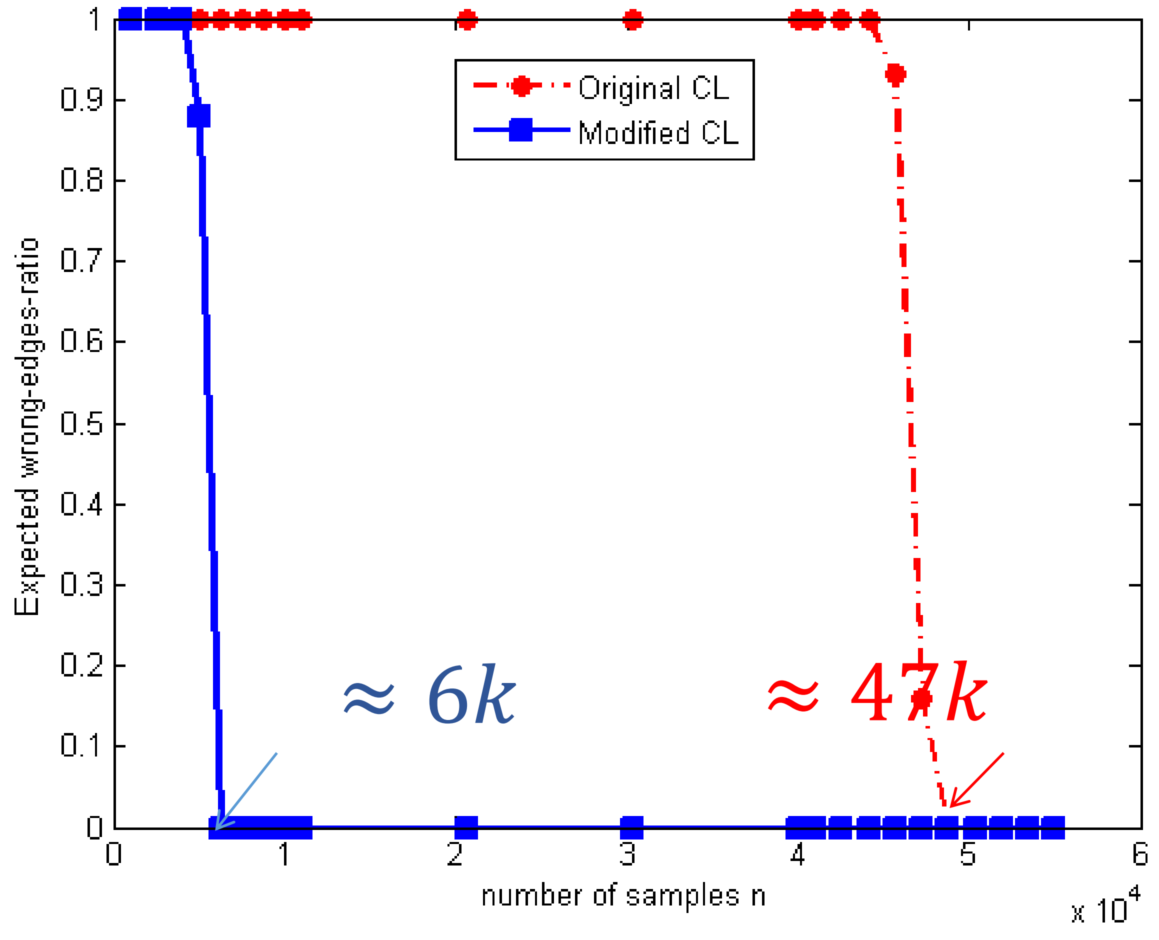
Setting: star graph, 7 nodes, each node alphabet size 300

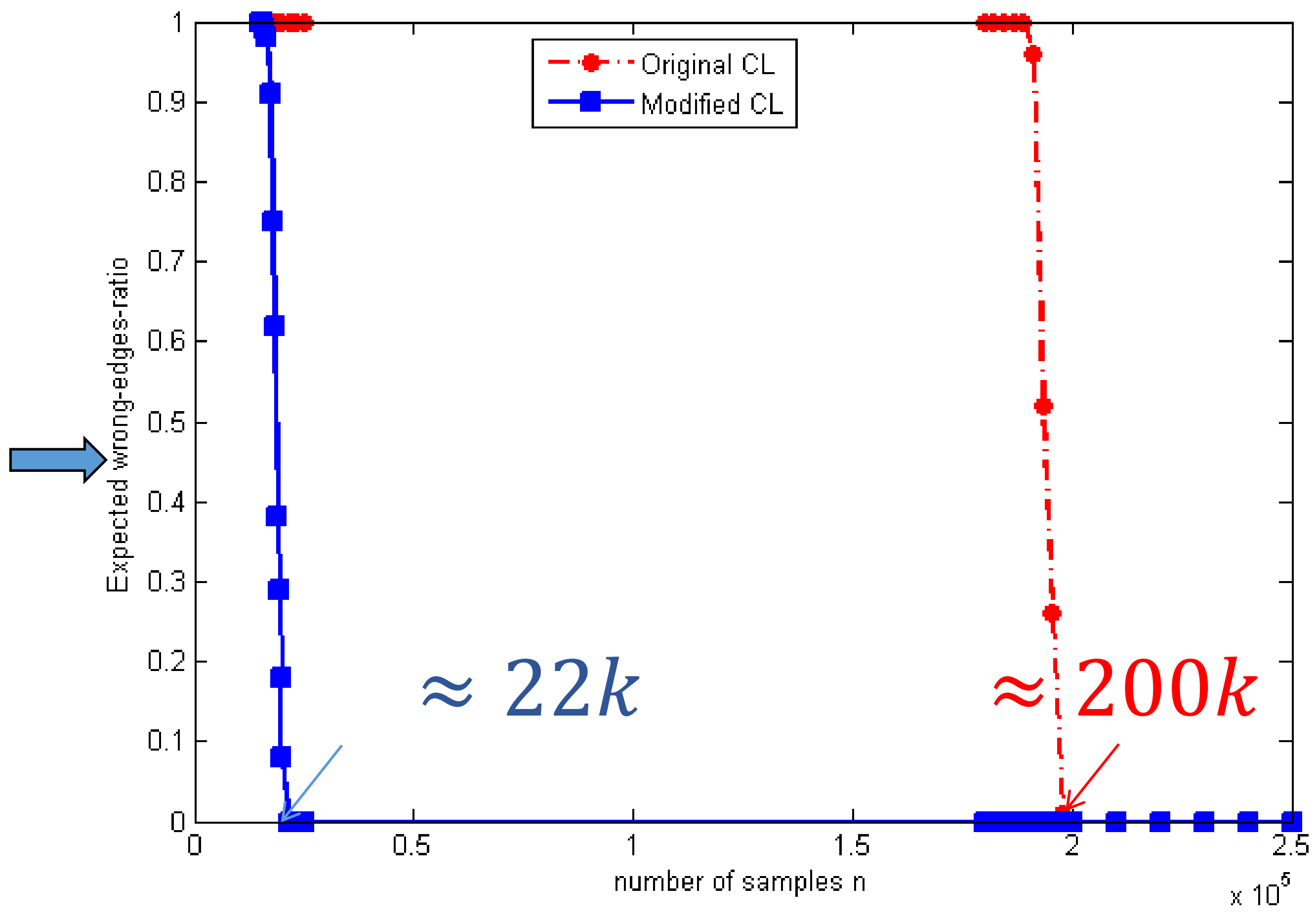
Our modified CL (JVHW'15)



Setting: star graph, 7 nodes, each node alphabet size 300

Our modified CL (JVHW'15)

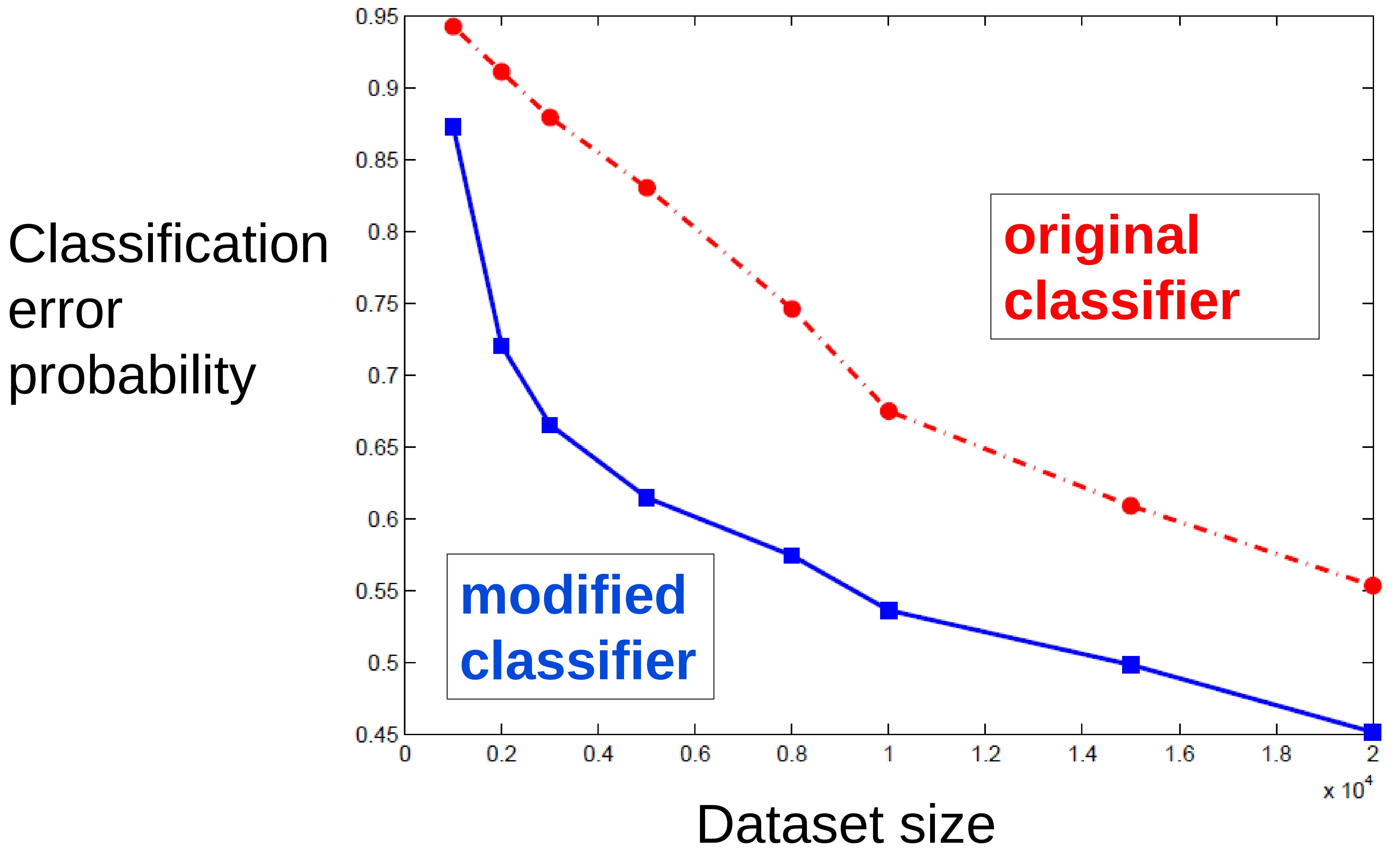




$$\boxed{\text{Alphabet size 300}} \longrightarrow \boxed{\text{Alphabet size 600}}$$

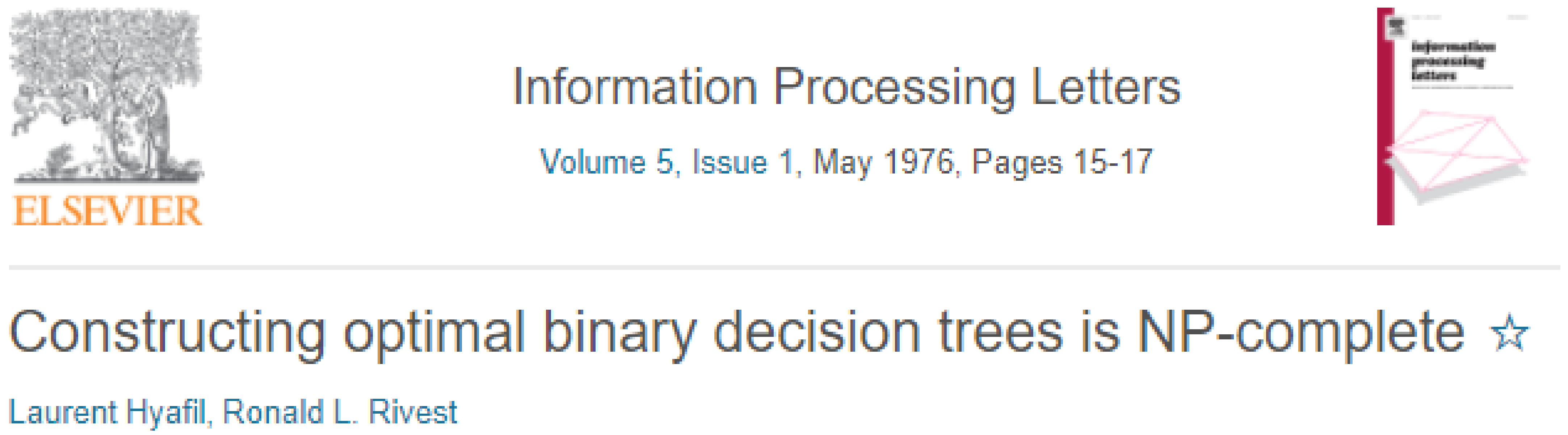
Tree-Augmented Naïve Bayes (JHW'16)

Dataset "**letter**" in UCI machine learning repository



Learning theoretic approach

How can we find a function $f(X)$ that estimates Y well?



ELSEVIER

Information Processing Letters

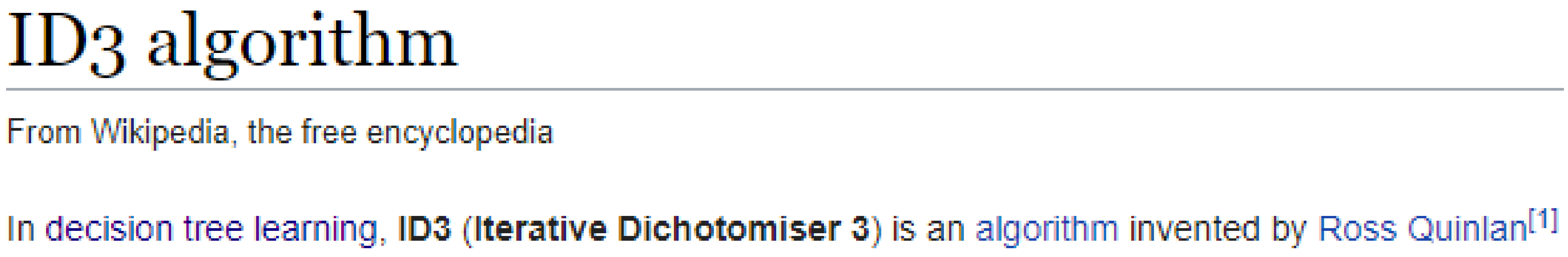
Volume 5, Issue 1, May 1976, Pages 15-17

Constructing optimal binary decision trees is NP-complete ☆

Laurent Hyafil, Ronald L. Rivest

Decision tree learning literature: use heuristics

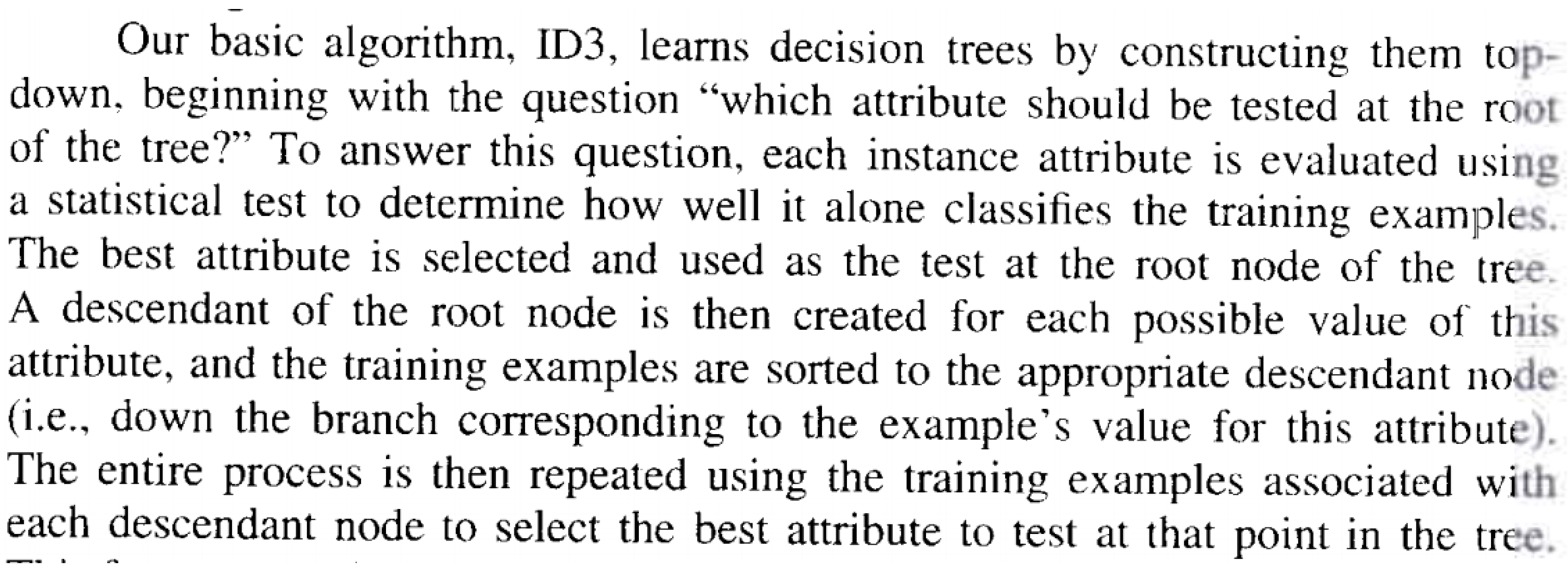
An early example: ID3



ID3 algorithm

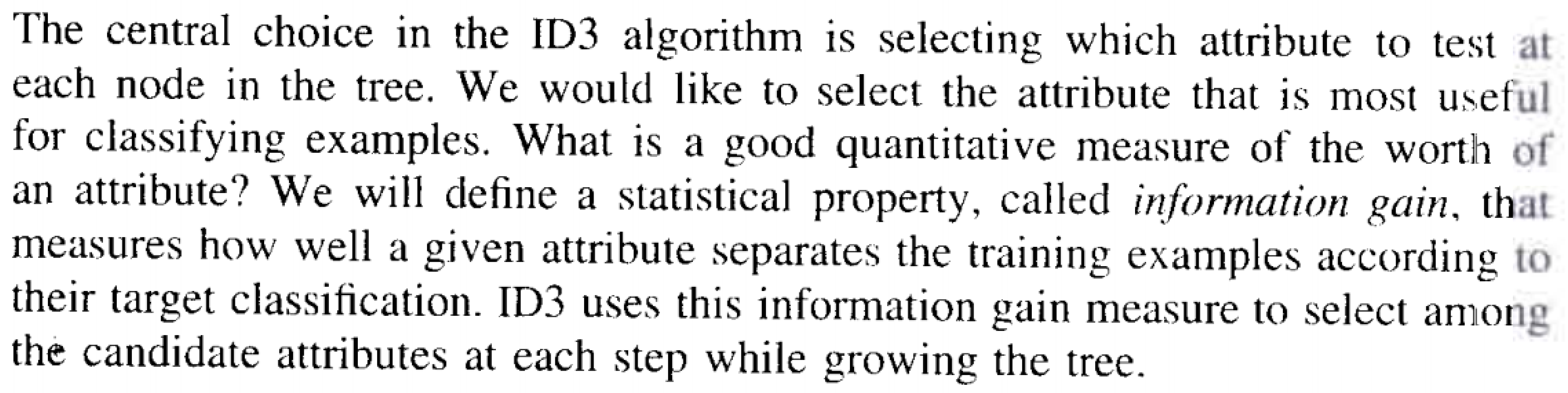
From Wikipedia, the free encyclopedia

In decision tree learning, **ID3** (**Iterative Dichotomiser 3**) is an algorithm invented by Ross Quinlan[1]



Our basic algorithm, ID3, learns decision trees by constructing them top-down, beginning with the question "which attribute should be tested at the root of the tree?" To answer this question, each instance attribute is evaluated using a statistical test to determine how well it alone classifies the training examples. The best attribute is selected and used as the test at the root node of the tree. A descendant of the root node is then created for each possible value of this attribute, and the training examples are sorted to the appropriate descendant node (i.e., down the branch corresponding to the example's value for this attribute). The entire process is then repeated using the training examples associated with each descendant node to select the best attribute to test at that point in the tree.

Measuring the quality of attributes



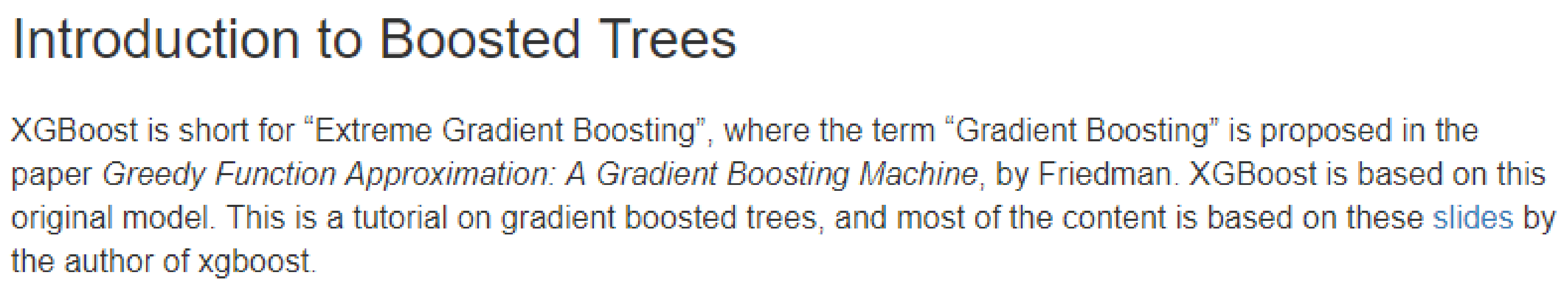
The central choice in the ID3 algorithm is selecting which attribute to test at each node in the tree. We would like to select the attribute that is most useful for classifying examples. What is a good quantitative measure of the worth of an attribute? We will define a statistical property, called *information gain*, that measures how well a given attribute separates the training examples according to their target classification. ID3 uses this information gain measure to select among the candidate attributes at each step while growing the tree.

Information Gain: conditional mutual information

$$I(Y; X_i | \{X_j : j \text{ selected}\})$$

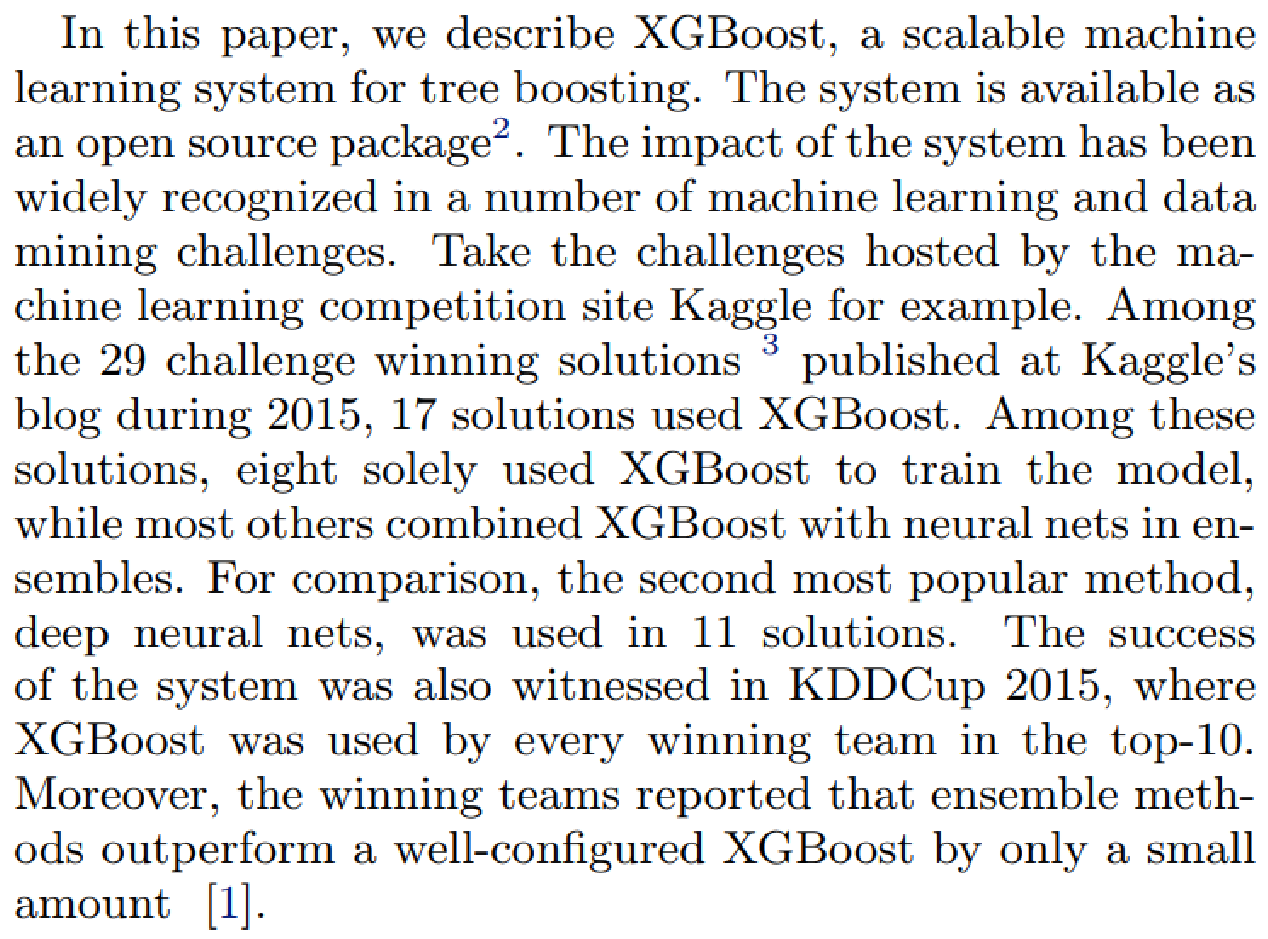
Many variants of decision trees

C4.5, CART, Adaboost, Xgboost…



Introduction to Boosted Trees

XGBoost is short for "Extreme Gradient Boosting", where the term "Gradient Boosting" is proposed in the paper *Greedy Function Approximation: A Gradient Boosting Machine*, by Friedman. XGBoost is based on this original model. This is a tutorial on gradient boosted trees, and most of the content is based on these slides by the author of xgboost.



In this paper, we describe XGBoost, a scalable machine learning system for tree boosting. The system is available as an open source package[2]. The impact of the system has been widely recognized in a number of machine learning and data mining challenges. Take the challenges hosted by the machine learning competition site Kaggle for example. Among the 29 challenge winning solutions [3] published at Kaggle's blog during 2015, 17 solutions used XGBoost. Among these solutions, eight solely used XGBoost to train the model, while most others combined XGBoost with neural nets in ensembles. For comparison, the second most popular method, deep neural nets, was used in 11 solutions. The success of the system was also witnessed in KDDCup 2015, where XGBoost was used by every winning team in the top-10. Moreover, the winning teams reported that ensemble methods outperform a well-configured XGBoost by only a small amount [1].