Non-parametric Entropy Estimation Toolbox (NPEET)

Greg Ver Steeg

July 6, 2013. Version 1.1

1 Introduction

This document describes a package of Python code for implementing various non-parametric continuous entropy estimators (and some discrete ones for convenience). After describing installation, Sec. 3 provides a wide-ranging discussion of technical, theoretical, and numerical issues surrounding entropy estimation. Sec. 4 provides references to the relevant literature for each estimator implemented. If you use these estimators in your research, please cite the appropriate authors. Sec. 5 describes the functionality and options in details.

2 Installation notes

This package contains Python code implementing several entropy estimation functions. The implementation is very simple. It only requires that numpy/scipy be installed. It requires a fairly recent version of scipy (≥ 0.12).

```
$curl -O http://www.isi.edu/~gregv/npeet.tgz
$tar -xvf npeet.tgz; cd npeet
$python
>>> import entropy_estimators as ee
>>> x = [[1.3],[3.7],[5.1],[2.4],[3.4]]
>>> y = [[1.5],[3.32],[5.3],[2.3],[3.3]]
>>> ee.mi(x,y)
Out: 0.168
```

You can run "python test.py" and see code inside for example usage.

3 Notes about entropy estimation

3.1 Theoretical

The basic idea behind non-parametric entropy estimators is that you locally estimate the log probability density at each sample point, then you average these to get an estimate. In spirit, this is very similar to kernel density estimation, except we do not have to go through the intermediate step of constructing a probability density and then integrating over it. This local probability density estimate at each point is subject to caveats, however. In particular, it is assumed that the probability density around a point is uniform within the (max-norm) ball containing its k-nearest neighbors. It is easy to imagine that this can fail; we will discuss some potential problems below. However, as N, the number of samples, gets big, this procedure should be a consistent estimator, unlike kernel density estimates [8].

Manifolds What if your high-dimensional vectors lie on a low-dimensional manifold? It depends on what is on the manifold, and what quantity you are looking for. You can define the entropy restricted to the manifold and Alfred Hero has some work investigating that. If you are estimating mutual information $MI(x : y)$ and the points x or y or both are restricted to low dimensional manifolds, that is no problem. The dimension does not enter into the estimators, only nearest neighbors in the joint and marginal subspaces. We investigate this nice property of mutual information estimators [11].

On the other hand if the *joint* distribution of x, y is on a manifold, or even nearly so, there is a problem. As a concrete example, imagine $y = x + noise$. Nearly functional relationships like this badly violates our assumption that the probability density is nearly uniform in a max-norm ball around each sample point. As the relationship between x, y gets closer to functional, the estimator will converge to $\log(N)$, where N is the number of samples, (as opposed to diverging as it should). This makes sense in a way. In principle you cannot tell if x gives more than $\log N$ bits of information about y from only N samples. (But in practice, you can often do this if you make any reasonable assumptions at all about your data.)

Entropy decomposition In principle, you do not need fancy estimators for every entropy. You just need the basic estimator $H(X)$, then you can combine it in clever ways to produce higher order entropies. Take the conditional mutual information as an example $I(X : Y|Z) = H(X, Z) + H(Y, Z) - H(X, Y, Z) - H(Z)$. If this quantity is nearly zero, but each of those individual entropies is big, you will be adding errors together for each term and you will get a very unreliable estimate. This is what is done for the mixed continuous-discrete mutual information estimator and the discrete estimators, so beware.

Mixed estimator In particular, if we have $I(X:Y)$ where X is a continuous variable and Y is discrete, taking values $Y = 1, \dots, k$, we can write mutual information as,

$$I(X:Y) = H(X) + \sum_{i=1}^{k} p(Y=i) H(X|Y=i).$$

Our estimator replaces $H(X)$ and $H(X|Y=i)$ with the standard k-nearest neighbor entropy estimator. After conditioning on some value of $Y = i$, there may not be enough points remaining to estimate entropy (fewer than $k+1$ points). In that case, we assume the entropy is maximal, i.e., $\hat{H}(X|Y=i) \equiv \hat{H}(X)$. That way, insufficient samples produces a mutual information of zero. This represents a philosophical choice that we should prefer to err on the side of under-estimating mutual information given insufficient data.

3.2 Implementation

In the Kraskov paper, they use the max-norm, as we do here. They claim that any norm would be fine. Anecdotally, I have had problems with convergence when trying to use the Euclidean norm. I did not try very hard though, so possible it was due to some mistake.

As mentioned, we use $k = 3$ nearest neighbors by default [5]. Smaller k should lead to less bias (because we are only assuming constant density in a smaller neighborhood), but can lead to more noise. Larger k reduce variance. $k = 3$ seems to strike a good balance.

Kraskov et. al. in the "Implementation details" section describe adding low intensity noise to break degeneracies in calculating nearest neighbors. I have done that as well. The noise is of order 10^{-10}, so be sure this is a small perturbation compared to your data.

3.3 Numerical

Although the estimators are *asymptotically* unbiased, for small N, there may still be bias (see e.g. [12]). You can even see this in the example in "test.py". If you are comparing multiple entropies, it may be better to use the same number of samples for each estimate. If you want you can average the estimated entropy over multiple random subsamples of fixed size.

Negative MI Mutual and conditional mutual information (even for continuous variables) should never be negative. However your *estimate* may be negative, due to error in the estimator. If you want to get a handle on the error in your estimator, use something like the shuffle_test routine on your data.

Other Limitations The shuffle test gives an estimate of the error that is especially useful for small mutual information. However, your resolution is still fundamentally limited by N, and this can cause problems if you are not careful [4].

Optimization In general, the slowest part of entropy estimation is finding the nearest neighbors. This is implemented in C using scipy's cKDTree method, so it should be pretty fast ($O(N \log N)$). On the other hand, in high-dimensions, KDTree's are not very good, and you might be reduced to the speed of brute force N^2 search for nearest neighbors. If N is too big, I recommend using multiple, random, small subsets of points ($N' << N$) to estimate entropies, then average over them.

4 References

If you make use of any estimators, please cite the appropriate literature described below. Feel free to cite this document if you find the software to be a useful tool.

The mutual information estimator is from Kraskov et. al. [7]. Note that it corresponds to the estimator they call $I^{(1)}$ in Eq. 8. The continuous entropy estimator is based on Kozachenko and Leonenko [6], but, as a non-Russian speaker, I implemented it based on the Kraskov paper. The conditional mutual information estimator comes from Palus et. al. [9]. There were other choices [8], but this one was particularly simple. The KL Divergence estimator comes from Wang et. al. [13], Eq. 5. They suggest many nice variations as well. And, of course, mutual information can be written as a KL divergence, so this can provide some alternate estimators. The estimators from Wang et. al. can be particularly useful if you know something about relevant length scales. I have not seen the mixed continuous-discrete mutual information estimator before, but my implementation described above is fairly obvious. Another version has been considered [3]. We also introduce an elegant version of this estimator [4], which will be implemented in future versions.

4.1 Discrete entropy estimation

The discrete estimators are simply the "naive" or "plug-in" estimators, which can be badly biased in the absence of sufficient data. There is a well-developed literature on estimating discrete entropies in the case that the number of samples not much larger than the number of discrete states. We discuss a few of the simple approaches taken in the neuroscience literature along with references in [10]. See DeDeo et. al.'s paper for more sophisticated methods and references [2]. They have also made a package available implementing these bias correction procedures for discrete entropies at `http://thoth-python.org/`.

5 Functions

5.1 Continuous

entropy(x) Estimates the (differential) entropy of $x \in \mathbb{R}^{d_x}$ from samples $x^{(i)}, i = 1, \ldots, N$. The input should be a list of "vectors", that is a list of lists, as in the example above. Remember, differential entropy, unlike discrete entropy, can be negative! [1]

mi(x,y) Estimates the mutual information between $x \in \mathbb{R}^{d_x}$ and $y \in \mathbb{R}^{d_y}$ from samples $x^{(i)}, y^{(i)}, i = 1, \ldots, N$.

cmi(x,y,z) Estimates the conditional mutual information between $x \in \mathbb{R}^{d_x}$ and $y \in \mathbb{R}^{d_y}$ conditioned on $z \in \mathbb{R}^{d_z}$ from samples $x^{(i)}, y^{(i)}, z^{(i)}, i = 1, \ldots, N$.

kldiv(x,xp) Estimates the KL divergence between two distributions $p(x), q(x)$ from samples x, drawn from $p(x)$ and samples x' drawn from $q(x)$. The number of samples do not have to be the same. KL Divergence is not symmetric. This function is untested.

5.2 Discrete

entropyd(x) Estimates the *discrete* entropy given a list of samples of discrete variable x.

midd(x,y) Estimates the mutual information between discrete variables x and y.

cmidd(x,y,z) Estimates the conditional mutual information between discrete variables x and y conditioned on discrete variable z.

5.3 Mixed

micd(x,y) Estimates the mutual information between a *continuous* variable $x \in \mathbb{R}^{d_x}$ and a *discrete* variable y. Note that mutual information is symmetric, but you must pass the continuous variable first. This routine has not been tested. Additional caveats below apply.

5.4 Utilities

shuffle_test(measure,$x,y,[z]$**)** Shuffle the x's so that they are uncorrelated with y, then estimates whichever information measure you specify with "measure". E.g., mutual information with **mi** would return the average mutual information (which should be near zero, because of the shuffling) along with the confidence interval (95% by default, set with the keyword $ci = 0.95$). This gives a good sense of numerical error and, particular, if your measured correlations are stronger than would occur by chance.

vectorize(x) Turns a list of scalars into a list of one-dimensional vectors, as required by the estimators. E.g. $[1, 2, 3]$ to $[[1], [2], [3]]$.

5.5 Options

For continuous estimators, you can set the number of nearest neighbors to use by passing the argument $k =?$. The default is $k = 3$. The reason for this choice is that it has worked well for me, and for Kraskov, and our intuitions were confirmed by a more extensive study [5]. All information is reported in bits, but this can be changed by passing the argument $base =?$.

Acknowledgements

The research that led to development of this code was done with Aram Galstyan. Thanks also to Nathan Hodas and Shuyang Gaos for use, comments, and improvements. The research that resulted in this code was supported by DARPA grant No. W911NF–12–1–0034, AFOSR award FA9550-11-1-0417, and AFOSR MURI grant No. FA9550-10-1-0569.

References

[1] Thomas M Cover and Joy A Thomas. *Elements of information theory.* Wiley-Interscience, 2006.

[2] S. DeDeo, R. Hawkins, S. Klingenstein, and T. Hitchcock. Bootstrap methods for the empirical study of decision-making and information flows in social systems. *eprint arXiv:1302.0907*, December 2013. `http://arxiv.org/abs/1302.0907`. *Entropy*, in press.

[3] Gauthier Doquire and Michel Verleysen. Mutual information based feature selection for mixed data. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2011.

[4] Fei Sha Simon DeDeo Greg Ver Steeg, Aram Galstyan. Demystifying information-theoretic clustering. In *Under review*, 2013.

[5] Shiraj Khan, Sharba Bandyopadhyay, Auroop R Ganguly, Sunil Saigal, David J Erickson III, Vladimir Protopopescu, and George Ostrouchov. Relative performance of mutual information estimation methods for quantifying the dependence among short and noisy data. *Physical Review E*, 76(2):026209, 2007.

[6] L. F. Kozachenko and N. N. Leonenko. Sample estimate of the entropy of a random vector. *Probl. Peredachi Inf.*, 23(2):95–101, November 1987.

[7] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Phys. Rev. E*, 69:066138, Jun 2004.

[8] B. Póczos and J. Schneider. Nonparametric estimation of conditional information and divergences. 2012.

[9] M. Vejmelka and M. Paluš. Inferring the directionality of coupling with conditional mutual information. *Physical Review E*, 77(2):026214, 2008.

[10] Greg Ver Steeg and Aram Galstyan. Information transfer in social media. In *Proc of. World Wide Web Conference*, 2012.

[11] Greg Ver Steeg and Aram Galstyan. Information-theoretic measures of influence based on content dynamics. In *Proceedings of the 6th International Conference on Web Search and Data Mining*. ACM, 2013.

[12] Jonathan D. Victor. Approaches to information-theoretic analysis of neural activity. *Biological Theory*, 1(3):302–316, 2006.

[13] Qing Wang, Sanjeev R. Kulkarni, and Sergio Verdú. Divergence estimation for multidimensional densities via k-nearest-neighbor distances. *IEEE Trans. Inf. Theor.*, 55:2392–2405, May 2009.