Applied Information Theory 650

http://www.cs.umass.edu/∼elm/Teaching/650_F14/

Assignment: Estimating Information Theoretic Quantities from Data

In this assignment, we will explore various methods for estimating entropy, KL-divergence, and mutual information.

1 Estimating differential entropy

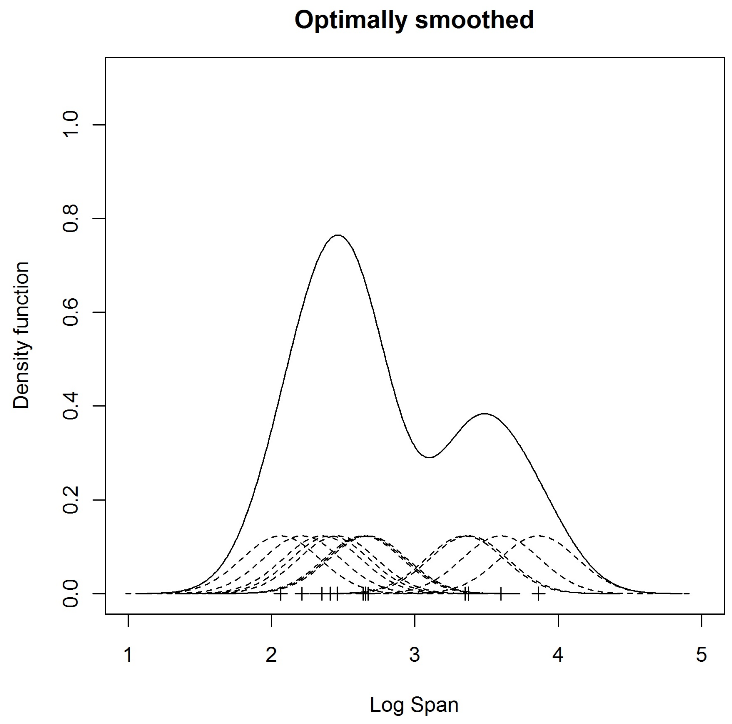
We will start by estimating the entropy of a continuous one-dimensional distribution from which we have been given a sample. Suppose that the distribution has a density $f(x)$. In this part of the assignment, we will explore the difference between estimating entropy using a plug-in estimator based on a non-parametric density estimate and a more direct entropy estimator, the m-spacings estimate of entropy proposed by Vasicek.

1.1 Plug-in estimates of entropy

The idea of a plug-in entropy estimator is simple. First estimate the density $\hat{p}(x)$ of the data. Then estimate the entropy of the density estimate. We will use a non-parametric density estimator to estimate the density. This estimator is also called a *Parzen window density estimator*.

1.1.1 Non-parametric density estimator

The idea of a non-parametric density estimate is illustrated in the diagram below. Given a sample from a one-dimensional distribution (shown as tick marks on the x-axis), a Gaussian kernel is placed on each sample point. The Gaussian kernels are shown as dotted lines, with one Gaussian over each sample point. The kernels are summed up at each point, leading to the top solid line, which is the non-parametric density estimate.



(figure copied from http://www.mvstat.net/tduong/research/seminars/seminar-2001-05/)

Let the sample of size n of the unknown distribution be denoted by the set $\{x_1, x_2, ..., x_n\}$. Furthermore, assume that we use a Gaussian kernel with standard deviation σ at each sample location. Note that this implies that the mean of the ith Gaussian is just x_i. The value of the non-parametric density estimate at a new location x is given by

$$\hat{p}(x) = \frac{1}{n}\sum_{i=1}^{n} N(x; x_i, \sigma).$$

where $N(x; x_i, \sigma)$ is the density of the point x under a Gaussian distribution with mean x_i and standard deviation σ.

Expanding our notation with the formula for a Gaussian distribution, we obtain:

$$\hat{p}(x) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-x_i)^2}{2\sigma^2}}.$$

In words the idea is this. The density at a new point is simply the average of the contributions to that point by all of the kernels. If the new sample point x is near to a sample point x_i, it will get a big contribution from that x_i's kernel. If it is far from another point x_j, then it will get a very small contribution from x_j's kernel.

`Function 1.` Write a function

`p = densEst(xs, x, sigma).`

that takes 3 arguments, the original sample points `xs` organized as a column vector, the point `x` at which the density is to be evaluated, and the standard deviation `sigma` of the kernel, and returns `p`, the density of the kernel density estimate at the point `x`.

1.1.2 Leave-one-out point estimates

A leave-one-out estimate is a way of evaluating the bandwidth (or standard deviation) σ of a non-parametric density estimate. Given a sample $\mathbf{x} = \{x_1, x_2, ..., x_n\}$, the leave-one-out estimate of a point x_j that is one of the original sample points is the density estimate of x_j using all of the points in $\mathbf{x}$ *except* for x_j itself:

$$looPoint(\mathbf{x}, j, \sigma) = \texttt{densEst}(\mathbf{x} \setminus x_j, x_j, \sigma),$$

where j is the index of the point to be evaluated, and $\mathbf{x} \setminus x_j$ means all of the elements of the set $\mathbf{x}$ except for x_j.

`Function 2.` Write the function looPoint described above. The function prototype should be:

```
p = looPoint(xs, j, sigma)
```

1.1.3 Leave-one-out sample likelihood

We will generate the *leave-one-out-sample-likelihood* by finding the average log likelihood of a sample using the leave-one-out point estimates. In other words, we will take each point of a sample, get its likelihood using the leave-one-out point estimate, and take the average of the log of these leave-one-out estimates. This will give us a sense of how "good" a given σ is for a particular sample.

`Function 3.` Write the function looAvgLogLike described above. The function prototype should be:

```
avgLogLike = looAvgLogLike(xs, sigma).
```

1.1.4 Optimizing the kernel bandwidth

Finally, we would like to find the value of σ that gives us the highest average log likelihood of the leave-one-out estimate. It is usually impractical to optimize over all possible values of σ. Instead, you should try a fixed set of sigmas, starting from something small and going to something big. For example, you could start at the shortest distance between any two samples (a reasonable lower bound on σ) and go up to the largest distance between samples (a reasonable upper bound on σ), stepping by factors of 1.1 (or more if you like) along the way. I will leave it to you on how to decide on a set of σ's to try.

`Function 4.` Write a function `findBestSigma` to find the best sigma for a given set of samples, using the looAvgLogLike function. The function prototype should be:

```
bestSigma = findBestSigma(xs, S)
```

Here S is a set of values of σ that you want to try. (Try at least 10 values of σ.)

1.2 Estimating the entropy of a given non-parametric density estimate.

Now that you have a way of finding the best value of σ over some set, you can turn a given sample and the optimal σ into a reasonably good non-parametric density estimate. The next goal is to estimate the entropy of this density.

1.2.1 Monte Carlo estimate of entropy from a density

Since differential entropy is an expectation (of the negative log probability density), we can estimate it by sampling from a distribution and forming an empirical average of the negative log probability density. This empirical average will converge to the true differential entropy by the law of large numbers.

Writing a function that draws a single sample point from a non-parametric density estimate is simple. All you have to do is choose one of the original sample points with uniform probability. Then choose a random Gaussian deviation from this sample point, scaling by the appropriate σ. For example, suppose my original sample has 10 points. I choose one of these points randomly, perhaps point 7. I then add a random Gaussian offset to this point by using the function `randn` in matlab and scaling by the appropriate factor for σ. The original point x_7 plus the random Gaussian offset gives me a valid sample from the non-parametric density estimate. I use my non-parametric density estimate function (function 1 in this homework) to get the density of this point, take the log, and add it to my cumulative sum.

`Function 5.` Write a function to generate a single sample point from a non-parametric density estimate. It takes as arguments the original list of points and the kernel standard deviation σ. It returns a single sample.

```
x = sampleNPDE(xs, sigma)
```

`Function 6.` Now write a function to estimate the entropy of a non-parametric density estimate by sampling as described above:

```
ent = samplingEntropyEst(xs, N, sigma)
```

Here N is the number of times to sample in the Monte Carlo procedure.

1.3 The $m-$spacings estimate of entropy

Write a function to compute the $m-$spacings estimate of entropy. Use the formula for the $m-$spacings estimate given in class. For the value of m, just use the nearest integer value to the square root of n, the number of sample points. (There are other ways to choose m, but this one should work fine.)

`Function 7.` The function prototype should be:

```
ent = mspacingsEntropyEst(xs)
```

2 Experiments with entropy estimators

Functions 6 and 7 above are two different entropy estimators. You will do some experiments to look at bias, variance, and speed of computation for some standard distributions.

For each of the entropy estimates above, try estimating the entropy of the following distributions:

1. Uniform distribution on the interval $[0, 1]$.
2. Uniform distribution on the interval $[0, 8]$.
3. Uniform distribution on the interval $[0, 0.5]$.
4. Gaussian distribution with mean 0 and standard deviation 1.
5. Gaussian distribution with mean 0 and standard deviation 100.
6. Exponential distribution with mean 1.
7. Exponential distribution with mean 100.

You will need to figure out how to draw samples from each of these distributions. (Note, this is easy in matlab, as it supports sampling from all of these distributions.)

For each of these distributions, you should do the following:

- Give the true entropy in bits. (You can look up the formula for these in the book or on the web.)
- Run experiments for both entropy estimators.
- For each entropy estimator, estimate the entropy based on samples of 4 different sizes: 10,100,1000,10000.

- For each estimator, distribution, and sample size, run the entropy estimator 10 different times. This will allow you to get the variance of the entropy estimate.
- Calculate the mean and standard deviation of the entropy estimate.
- Record running times for all experiments.

Summarize and analyze your results. What are the pros and cons of each estimator?