10-704: Information Processing and Learning **Spring 2015**

Lecture 5: January 27

Lecturer: Aarti Singh *Scribes: Kyle Soska*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

5.1 Summary of Entropy Estimators

5.1.1 Discrete variables

We have a distribution P supported on a finite alphabet $\{1, ..., d\}$ with $P(X = j) = p_j$ where $\sum_{j=1}^{d} p_j = 1$. We observe independent samples $\{X_i\}_{i=1}^n \sim P$ and would like to estimate some functional of P, say the entropy:

$$H(P) = -\sum_{j=1}^{d} p_j \log_2(p_j)$$

For this problem, we discussed two estimators:

- **Plugin Estimator**

  The **plugin** estimator uses empirical estimates of the frequencies $\hat{p_j} = \frac{1}{n}\sum_{i=1}^{n} 1[X_i = j]$ to obtain an estimate of the entropy as follows:

  $$\hat{H}_n = -\sum_{j=1}^{d} \hat{p_j} \log_2(\hat{p_j})$$

- **LP Estimator**

  The **LP Estimator** works by transforming the samples $\{X_i\}_{i=1}^n$ into a **fingerprint**, which is the vector $f = (f_1, f_2, ...)$ for which f_i is the number of elements in the domain that occurred i times in the sample, i.e. with empirical frequency i/n. The fingerprint is also known as the "type" of a sample.

  The fingerprint serves as an estimate of the histogram of the distribution which is defined as the mapping $h_p : (0, 1] \to \mathbb{N} \cup \{0\}$ for which $h_p(x)$ is the number of domain elements that occur with probability exactly x. More formally, $h_p(x) = |\{\alpha : p_\alpha = x\}|$. The entropy can be computed from the histogram as follows:

  $$H(x) = -\sum_{x:h_p(x)\neq 0} h_p(x) x \log_2(x)$$

  We can then quantize $(0, 1]$ to a grid $Q = \{q_1, ..., q_k\}$ and let h_j be the variable associated with $h_p(q_j)$ to obtain the approximation:

$$\bar{H}(x) = -\sum_{q_j \epsilon \mathbb{Q}} h_j q_j \log_2(q_j)$$

The final estimate $\hat{H}_n(x)$ is obtained using a linear program that matches the h_js to the observed fingerprint f. For details, see previous lecture. The key take away is that using fingerprint (or type) instead of empirical frequencies yields an estimator for entropy with optimal sample complexity.

5.1.2 Continuous variables

Here we assume the distribution P is continuous with density $p = dP/d\mu$. As before we obtain independent samples $\{X_i\}_{i=1}^n \sim P$. We would like to estimate the differential entropy:

$$H(p) = \int p(x) \log p(x) d\mu(x)$$

5.1.2.1 Plugin Estimator

There are three different ways to obtain a plug-in estimator using a density estimate $\hat{p}(x)$:

1. **Integral Estimate**: $\hat{H}(x) = -\int \hat{p}(x) \log \hat{p}(x) dx$
2. **Re-substitution Estimate**: $\hat{H}(x) = -\frac{1}{n} \sum_{i=1}^n \log \hat{p}(X_i)$ where $\hat{p}$ is obtained using the samples $\{X_1, ..., X_n\}$.
3. **Splitting Data Estimate**: $\hat{H}(x) = -\frac{1}{m} \sum_{i=1}^m \log \hat{p}(X_i)$ where $\hat{p}$ is obtained using the samples $\{X_{m+1}, ..., X_n\}$. A cross-validation estimate can be defined similarly, e.g. the leave-one-out estimate is given as $\hat{H}(x) = -\frac{1}{n} \sum_{i=1}^n \log \hat{p}_i(X_i)$ where $\hat{p}_i$ is obtained using all samples except X_i.

The key difference between the Re-substitution estimate and the Splitting Data Estimate is that the splitting estimate sums over different samples than the ones used for estimating the density $\hat{p}$.

5.1.2.2 Density estimators

The plug-in estimators defined above can be obtained by using any density estimator. Some popular examples are the following:

1. **Kernel Density Estimator (KDE)**

   Kernel density estimation takes the approach of estimating density at a given point using a kernel K with bandwidth parameter h to form a weighted average using other points from the sample. Intuitively, the points that are closer to the point whose density is being estimated will have a higher contribution to the density than points that are further away. The selection of the kernel and the bandwidth parameter adjust the characteristics of this relationship.

$$\hat{p}_h = \frac{1}{nh^d} \sum_{i=1}^n K\left(\frac{||x - x_i||}{h}\right)$$

2. **KNN Estimator**

   KNN density estimation at a point x follows by obtaining n samples from a distribution and computing the volume Vol_{KNN} needed to encapsulate k nearest points to x and then taking the ratio:

$$\hat{p}(x) = \frac{k}{nVol_{KNN}(x)} \equiv \frac{k}{n(\text{dist}_{KNN}(x))^d}$$

5.1.2.3 Von Mises Expansion

The Von Mises expansion (essentially a functional Taylor's expansion) of second-order for the entropy is as follows:

$$H(p) = H(q) + \int (\log(q(x)+1)(q(x)-p(x))dx + O(||q-p||_2^2) = -\int p(x)\log(q(x)) + O(||q-p||_2^2)$$

This allows us to analyze the error of a data-splitting entropy estimate as follows (let $q = \hat{p}$):

$$\hat{H}(p) - H(p) = -\frac{1}{n}\sum_{i=1}^{n}\log(\hat{p}(X_i)) + \int p(x)\log(\hat{p}(x)) + O(||\hat{p}-p||_2^2)$$

The first two terms on the right hand side can be bounded using standard concentration inequalities that control deviation of empirical mean from true mean, and the third term is simply the mean square error of the density estimate which is typically known e.g. for kernel or k-NN density estimate. We will discuss the mean square of a kernel density estimator next, and the resulting error guarantee on the corresponding integral estimate of the entropy.

5.2 Kernel Density Estimation Mean Square Error

We will consider Hölder-β smooth density functions.

Holder Smoothness: a function f is β-Holder if $\forall r \leq \lfloor\beta\rfloor,\ |f^{(r)}(x+\mu) - f^{(r)}(x)| \leq C|\mu|^{\beta-r}$

The mean square error of the Kernel Density Estimator decomposes into a Bias and a Variance term: $\mathbb{E}[||\hat{p}_n - p||^2] = Bias^2 + Variance$. Here the Bias^2 scales as $\mathcal{O}(h^{2\beta})$ and Variance scales as $\mathcal{O}(\frac{1}{nh^d})$. .

The bandwidth parameter h may be selected to balance the bias-variance tradeoff. One strategy would be to set the derivative of the sum equal to 0 and solve:

$$\frac{\partial}{\partial h}\left(h^{2\beta} + \frac{1}{nh^d}\right) = 2\beta h^{2\beta-1} - \frac{d}{nh^{d+1}} = 0 \ \Rightarrow\ h \asymp n^{\frac{-1}{2\beta+d}}$$

assuming β, d are constants. The resulting mean square error is $n^{\frac{-2\beta}{2\beta+d}}$. Thus, Kernel density has poor performance in higher dimensions (as d becomes large) and is often impractical for dimensions higher than 5. Smoother function (β higher) have better convergence, but for any finite β, the density estimator cannot achieve parametric rate of mean square error convergence (which is n^{-1}).

5.3 Integral Entropy Estimate Error

The entropy estimation error can be decomposed as:

$$|\hat{H}(p) - H(p)| \leq |\mathbb{E}\hat{H}(p) - H(p)| + |\hat{H}(p) - \mathbb{E}\hat{H}(p)|$$

The first term can be regarded as Bias and the second as standard deviation for entropy estimation.

If p is Hölder-β smooth, then it can be shown that for the integral entropy estimator, we have:

$$Bias \asymp h^{\beta} + \frac{1}{nh^d} \qquad St.dev : \frac{1}{\sqrt{n}}$$

We can pick h to optimize the bias-st.dev tradeoff for entropy estimation:

$$\frac{d}{dh}\left(h^{\beta} + \frac{1}{nh^d}\right) \to \beta h^{\beta-1} - \frac{d}{nh^{d+1}} = 0 \qquad h \asymp n^{\frac{-1}{\beta+d}}$$

With this setting (note its different than the optimal bandwidth for kernel density estimation), we have:

$$Bias \asymp n^{\frac{-\beta}{\beta+d}},\ st.dev. \asymp \frac{1}{\sqrt{n}},\ \text{and hence}\ \ |\hat{H} - H| \asymp \max\{n^{\frac{-\beta}{\beta+d}}, n^{-1/2}\} \asymp \frac{1}{\sqrt{n}}\ \text{if}\ \beta \geq d$$

Thus, for smooth functions with $\beta \geq d$, the integral entropy estimator achieves parametric rate (which is $n^{-1/2}$).

The Von-Mises estimator (data-splitting estimator with Von-Mises expansion error analysis) can achieve parametric rate when $\beta \geq \frac{d}{2}$, and hence outperforms the integral entropy estimator.
It can be shown that the optimal value is $\beta \geq \frac{d}{4}$ to achieve parametric rate and this can be achieved by using higher-order expansion in the Von-Mises expansion.

5.4 Estimating Mutual Information

Estimating quantities such as mutual information can be reduced to the problem of estimating entropy or density by expressing mutual information in terms of these quantities:

$$\hat{I}(X,Y) = \hat{H}(X) + \hat{H}(Y) - \hat{H}(X,Y)$$
$$\hat{I}(X,Y) = D(\hat{p}(x,y)||\hat{p}(x)\hat{p}(y)) = \int \hat{p}(x,y)\log\frac{\hat{p}(x,y)}{\hat{p}(x)\hat{p}(y)}dxdy$$

5.5 Application to machine learning: Estimating Structure of Graphical Models

The joint distribution of a collection of random variables can be expressed using chain rule as:

$$p(X_1, ..., X_p) = \prod_{i=1}^{p} p(X_i|X_{i-1}...X_1)$$

.

It may be the case that the distribution of random variables admits a more restricted form which can be characterized by a graphical structure in which case we define $pa(X_i)$ to be the set of parents of X_i and the joint probability can be expressed as:

$$p(X_1, ..., X_p) = \prod_{i=1}^{p} p(X_i|pa(X_i))$$

Graphical models capture the relationship of conditional independence among variables. For a variable X_i, $X_i \perp non-descendants|pa(X_i)$. Other conditional independence relations represented by the graphical model can be read off using d-separation. Tests for the conditional independence of random variables becomes essential to the estimation of graphical models. A test for conditional independence can be based on mutual information since the latter is zero for independent variables.

We may observe the p random variables n times $[X_1^{(i)}, ..., X_p^{(i)}]_{i=1}^n$ and want to estimate the structure of the underlying graphical model. Without making any simplifying assumptions this problem is NP-Hard as we need to test independence relations between all possible pairs of subsets of nodes conditioned on another subset of nodes. However, the problem is feasible for specific graphs such as trees.

Tree Assumption: each node has only one parent, that is $\forall X_i, |pa(X_i)| = 1$.

Under the tree assumption, the factorization of the joint distribution can be written as:

$$p(X_1, ..., X_p) = \prod_{i=1}^{p} \frac{p(X_i, pa(X_i))}{p(pa(X_i))} = \prod_{i=1}^{p} p(X_i) \prod_{(i,j)\epsilon\mathcal{E}} \frac{p(X_i, X_j)}{p(X_i)p(X_j)}$$

where $\mathcal{E}$ be the set of edges belonging to the graph. This follows since each node appears as many times in the second product term as its degree, but is a parent only degree-1 times (hence the need for the first product term).

We want to maximize the likelihood of the data (or equivalently minimize the negative log likelihood) over the edge set.

$$\max_{\mathcal{E}} \prod_{i=1}^{n} p\left(X_1^{(i)}, ..., X_p^{(i)}\right) \equiv \min_{\mathcal{E}} \sum_{i=1}^{n} \left[\log \frac{1}{p(X_1^{(i)}, ..., X_p^{(i)})}\right]$$

Lets look at the edge set which minimizes expected negative log likelihood. Using the factorized form of the distribution, the problem is equivalent to:

$$\min_{\mathcal{E}} \mathbb{E}\left[\sum_{i=1}^{p} \log\left(\frac{1}{p(X_i)}\right) - \sum_{i,j\epsilon\mathcal{E}} \log\left(\frac{p(X_i, X_j)}{p(X_i)p(X_j)}\right)\right]$$

$$= \min_{\mathcal{E}} \sum_{i=1}^{n} H(X_i) - \sum_{(i,j)\epsilon\mathcal{E}} I(X_i, X_j)$$

This is equivalent to finding edge set that maximizes mutual information of pairs $\max_{\mathcal{E}} \sum_{(i,j)\epsilon\mathcal{E}} I(X_i, X_j)$ which is the maximum weight spanning tree problem where the weights correspond to pairwise mutual information. The maximum weight spanning tree can be found very efficiently using Chow-Liu Algorithm

(also known as Kruskal's algorithm). The algorithm consists of picking pairs with max mutual information greedily (largest to smallest) and joining them provided no loop is formed.

If the true distribution does not correspond to a tree graphical model, it can be shown that this method (run with mutual information computed using bi-variate and uni-variate marginals of the distrribution) still finds the best tree approximation to any distribution.

Since the true mutual information is unknown, an estimate (discussed above) will need to be used. Further reading on estimating trees and sample complexity guarantees can be found at [CL68], [LH11].

References

[CL68] Chow, C. and Liu, C., "Approximating discrete probability distributions with dependence trees," *Information Theory, IEEE Transactions on*, 1968, pp. 462-467.

[LH11] Liu, H., Xu, M., Gu, H., Gupta, A., Lafferty, J., Wasserman, L., "Forest density estimation," *The Journal of Machine Learning Research 12*, 2011, pp. 907-951.