10-704: Information Processing and Learning **Fall 2016**

Lecture 6: Sept 19

Lecturer: Aarti Singh

Note: *These notes are based on scribed notes from Spring15 offering of this course. LaTeX template courtesy of UC Berkeley EECS dept.*

6.1 Summary of Plug-in Entropy Estimators

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.

Unlike the integral estimate, the re-substitution and splitting data estimators are easy to compute.

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 $\hat{p}$.

We will analyze the Splitting data estimator using Von-Mises analysis next.

6.1.1 Splitting data Estimator Analysis via Von-Mises expansion

Consider the following estimator. Split the data into half, use the first half to construct a density estimate $\hat{p}_h$ and then replace the integral (which is just an expectation) with a sample mean:

$$\hat{H}(p) = \frac{2}{n} \sum_{i=n/2+1}^{n} -\log(\hat{p}_h(X_i))$$

The analysis approach again uses the taylor expansion but is based on the observation that the linear, quadratic, and cubic terms involve simple functionals of the unknown density p (See Krishnamurthy et al. (2014) and references therein for details). There are some technical details here (you have to define derivatives on functions appropriately) but, for densities that are bounded from below, you can write:

$$\begin{aligned} H(p) &= H(q) + \int (\log(q(x)+1)(q(x)-p(x))dx + O(\|q-p\|_2^2) \\ &= -\int p(x)\log(q(x)) + O(\|q-p\|_2^2) \end{aligned}$$

This is just a functional taylor expansion, but it known as the Von Mises Expansion (since the functionals are defined over the space of distributions). The expansion immediately gives:

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

The first two terms are just the difference between a sample average and its mean, so it will have absolute error of order $O(1/\sqrt{n})$. The second term is just the MSE of the density estimator, which we know is $O(n^{\frac{-2\beta}{2\beta+d}})$ for the kernel density estimator with optimal bandwidth choice $O(n^{\frac{-1}{2\beta+d}})$. If the density smoothness $\beta > d/2$, this first-order corrected estimator has a $n^{-1/2}$ parametric rate. Notice that we set the bandwidth $h \asymp n^{\frac{-1}{2\beta+d}}$ which is the optimal bandwidth for density estimation. This contrasts with the plugin method.

It turns out that the minimax rate for these sorts of problems is typically $n^{-\min\{1/2, \frac{4\beta}{4\beta+d}\}}$. This means that the critical smoothness is $d/4$; if $\beta > d/4$ you can achieve the parametric convergence rate. The estimators that achieve this require higher order corrections. If you make a second order expansion with a third order remainder, you achieve the critical smoothness index, but are suboptimal in the $\beta < d/4$ regime. To be minimax optimal everywhere you have to make a third order expansion with fourth order remainder.

Some other comments:

1. This framework is particularly appealing since you can use whatever density estimator you like. You can use kNN-based, histogram, KDE etc. We have focused on analyzing the kernel density estimator, however a similar analysis follows for other estimators by plugging-in their mean square density estimation error. Specifically, the kNN-based density estimator is given as follows:

   $$\hat{p}_k(x) = \frac{k}{n \text{vol}_{kNN}(x)} \propto \frac{k}{n(\text{dist}_{kNN}(x))^d}$$

   where vol_{kNN} and dist_{kNN} are the volume of the smallest ball that contains the kth nearest neighbor of x and the distance to it, respectively. Since k/n is an estimate of the probability mass in that ball, dividing by its volume gives the density estimate. The corresponding entropy estimator is Kozachenko et al. (1987)

   $$\hat{H}_k = \frac{d}{n/2} \sum_{i=n/2+1}^{n} \log \text{dist}_{kNN}(x) + c_{k,n}$$

   The constant depends on n, k and the volume of the unit ball in d-dimensions. This estimator is consistent for all k (even fixed k, not increasing with n while the optimal choice of k for density estimation needs to scale with n) under some appropriate assumptions on density - boundedness, continuous on support and well-behaved boundary of support. Hence, often the mean-NN entropy estimator

   $$\hat{H} = \frac{1}{n-1} \sum_{k=1}^{n-1} \hat{H}_k = \frac{d}{n(n-1)} \sum_{i \neq j} \log \|x_i - x_j\| + const$$

   is used e.g. in the clustering paper Faivishevsky et al. (2010) we discussed earlier.

2. This is also great because you can tune the hyperparameter (in this case the bandwidth) by cross-validating the density estimator.

3. You can also show other properties for these first-order estimators in some special cases. For example, provided the first order term at the true distribution is not identically zero (for entropy this means p is uniform) you can show asymptotic normality. In this case you can also build confidence intervals by using a plugin estimator for the asymptotic variance Krishnamurthy et al. (2014).

6.2 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:

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

Developing such estimates in high dimensions and analyzing their properties is an open research direction.

6.3 More applications to Machine Learning

6.3.1 Feature selection

Mutual information can be used for feature selection in prediction problems (classification, regression, etc.) with high-dimensional inputs. For example, ideally one would like to select k features out of d that have maximum mutual information with the output Y:

$$\arg \max_{\Omega \subset [1,\ldots,d]; |\Omega|=k} I(X_\Omega, Y)$$

However, this objective (also known as information gain) is combinatorial requiring search over all possible subset of k out of d and is also not sub-modular, as you saw in QNA1 assignment. Nevertheless, in practice (such as in decision tree algorithms like ID3), often a greedy solution is employed that either selects the k variables with largest information gain $I(X_i, Y)$, aka mutual information score for each variable. However, this can pick redundant features and often better solutions can be obtained by selecting the variable with largest information gain, then condition on it, and select the next variable with largest information gain conditioned on the ones that are already selected, and so on. In applications, mutual information estimated from finite samples under parametric or non-parametric assumptions is used, as we discussed in previously.

6.3.2 Independence testing

Since $I(X,Y) = 0$ if X is independent of Y, estimators of mutual information can be used as test statistics for independence testing. To control the false-alarm or size of such tests i.e. $P_{X \perp Y}(\hat{I}(X,Y) > \text{threshold})$, we need to know or estimate the distribution of the test statistic under the null i.e. $X \perp Y$ (similarly, to control detection probability or power, we need to know or estimate the distribution of the test statistic under the alternate i.e. X not dependent on Y). The analysis we show argues that $\hat{I}(X,Y)$ converges to $I(X,Y)$ in mean, but as we discussed in von-Mises analysis, for some estimators we can also characterize the distribution of $\hat{I}(X,Y) - I(X,Y)$ (e.g. it is asymptotically gaussian), which enables bounding the false-alarm or size etc.

Similarly, conditional mutual information estimates can be used for conditional independence testing.

6.3.3 Machine learning on distributions

There has been a recent line of work referred to as "Machine Learning on Distributions." Here we treat the data points as distributions, and the goal is to do clustering, classification, or regression where the data

points are distributions, and we only access them through their samples. For example, each image can be represented by a collection of image patches which are samples from the distribution characterizing the image. See http://videolectures.net/nipsworkshops2012_poczos_learning/ for more examples.

Many ML algorithms can be written just with distances or similarities (via the kernel trick) and so we can use a divergence to specify a distance or a similarity. Two examples are kernel-SVM and spectral clustering, where we just need a measure of similarity between each example (which is a distribution). We can plug an estimate of relative entropy (KL-divergence, L_2-divergence, or any other distance between distributions/densities), into the kernel to define a measure of similarity. Then we can simply run these algorithms.

One caveat is that we may not be preserving symmetry and positive semidefiniteness of the Gram matrix. One workaround is to symmetrize the relative entropy e.g. as $(D(p||q) + D(q||p)/2$ and simply project the similarity matrix we have onto the PSD (positive semi-definite) cone - by taking eigendecomposition of the similarity matrix and setting negative eigenvalues to zero.

6.3.4 Estimating Structure of Graphical Models

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

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

.

In machine learning, graphical models are used as a graphical representation of conditional dependencies between random variables. When the distribution of random variables admits a more restricted form which can be characterized by a graphical structure. For directed graphical models, 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. There may be more than one graphical model that represents the same set of conditional independence relations. These are referred to as belonging to the same "equivalence class". We focus on the problem of learning the (undirected) structure of a graphical model. The structure can then be directed to recover one of the graphical models from an equivalence class of graphical models.

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).

One way to learn the structure of the graphical model is 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, and we are seeking a tree that spans all the vertices but has maximum sum of edge weights. 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 how the error in mutual information estimation propagates into error in learning the structure of the tree, and sample complexity guarantees for estimating trees can be found at Chow et al. (1968), Liu et al. (2011).

In next class, we will talk about an algorithm for learning the structure of general (non-tree) graphical models.

References

Krishnamurthy, A., Kandasamy, K., Poczos, B, Wasserman, L. (2014). Nonparametric Estimation of Renyi Divergence and Friends. *International Conference on Machine Learning (ICML).*

Kozachenko, L. F. , Leonenko, N. N. (1987). Sample estimate of the entropy of a random vector. *Problems Inform. Transmission, 23, pp. 95-101.*

Faivishevsky, L. & Goldberger, J. (2010). A Nonparametric Information Theoretic Clustering Algorithm. *International Conference on Machine Learning (ICML)*, 2010.

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

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