CONDITIONAL DENSITY ESTIMATION THROUGH NEURAL NETWORKS WITH AN APPLICATION TO DNA SEQUENCING-BASED MICROSCOPY

A PREPRINT

Ameya Anjarlekar
Department of Electrical Engineering
Indian Institute of Technology, Bombay
ameyanjarlekar@gmail.com

Manoj Gopalkrishnan
Department of Electrical Engineering
Indian Institute of Technology, Bombay
manoj.gopalkrishnan@gmail.com

April 13, 2020

ABSTRACT

This paper presents an approach for the estimation of conditional probability, P(x|y) given a set of observed samples. We estimated the conditional probability using an Artificial Neural Network where the expected output is calculated using the Kernel Density Estimation method. Further, we correctly predicted the condition on the distribution for a new test set of observed samples. Finally, we also tested our model wherein it was possible to correctly predict the micro-scale spatial information like the relative positions of bio-molecules on a surface without the need of conventional optics. Thus, also expanding our application to efficiently performing DNA microscopy.

Keywords Conditional Density Estimation · Kernel Density Estimation · Polymer Microscopy

1 Introduction

Almost all problems associated with random phenomena can be interpreted in terms of Probability Density Estimation (PDE) and in general in terms of Conditional Probability Density Estimation (when there are more than one random phenomena affecting a certain output). For example, in our problem of DNA microscopy we need to find the coordinates of the bio-molecules on a surface. This has been mainly achieved using small-scale imaging techniques as well as modern techniques [cite] like super-resolution microscopy which have largely relied on conventional optical methods. Also, superior methods like atomic force microscopy and transmission electron microscopy have achieved resolutions which are superior than optical methods. These basically utilize smaller probe sizes to interact with the sample more accurately.

Figure 1: Polymer growth between two molecules

We propose a method for conditional probability density estimation to find the spatial location of bio-molecules. We first grow polymer chains between sufficient pairs of molecules. Everytime a polymer chain is grown between two molecules, different number of monomer units get used up wherein the probability of the number of monomer units being (n) is related to the distance between the two molecules (λ) by some governing law. This is in fact is P(n|λ). We divide our problem into two phases: Calibration and Imaging.

In Calibration phase, we take pairs of bio-molecules whose distance is known and grow polymer chains between them to observe the number of monomer units. This process is further repeated for different number of distances. Hence, This creates samples (n,λ) which is eventually used to train an Artificial Neural Network along with the help of Kernel Density Estimation (discussed in detail in section 3). The Neural Network outputs the P(n|λ) with (n,λ) as inputs to it. This is discussed in depth in section 4.

As a part of the Imaging phase, we grow polymer chains between the pairs of molecules for whom their spatial coordinates are to be obtained. Therefore, each of these pairs of molecules we have a list of 'n' i.e. the number of monomer units. Using the P(n|λ) obtained in the Calibration phase and the lists of 'n' for each pair, we perform likelihood maximization to obtain the optimal coordinates for every point. A detail description of this phase is given in section 5.

Contributions:

2 Related Work

3 Background

3.1 Kernel Density Estimation

Let $x_1, x_2, \ldots, x_n$ be independent and identically distributed samples drawn from some distribution with an unknown density . Its kernel density estimator is given by

$$f_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{n \cdot h}\sum_{i=1}^{n} K_h(\frac{x - x_i}{h}) \tag{1}$$

where K is a non-negative function — and h > 0 is called the smoothing parameter

In this paper we choose the kernel to be a Gaussian kernel.

$$K(x - x_i) = \frac{1}{\sqrt{2 \cdot \pi \cdot \sigma^2}} exp^{(}\frac{(x - x_i)^2}{2 \cdot \sigma^2}) \tag{2}$$

In simple terms, this can be thought of as smoothing a histogram containing the number of samples plotted for each data point.
For an intuitive understanding, the value of kernel starts decreasing as x goes far from the sample point and the rate of

decrease can be controlled by the smoothing parameter h. Or we can say if h is large, the probability of getting points in the neighbourhood of a particular sample point is not affected considerably by the presence of the sample point in the sampling set.

3.2 Multi Dimensional Scaling (MDS)

Multi Dimensional Scaling is used to convert "information about the pairwise distances between a set of 'n' objects" into a configuration of 'n' points mapped into the Cartesian space.
Thus, given a distance matrix (nxn) (which would be either upper or lower triangular), MDS converts it into a configuration of 'n' points in the 2-dimensional Cartesian space.
However, the configuration is not unique due to the 2 translational degrees of freedom and 1 rotational degree of freedom of the points located in 2-dimensional space.

4 Calibration Phase

4.1 Neural Network

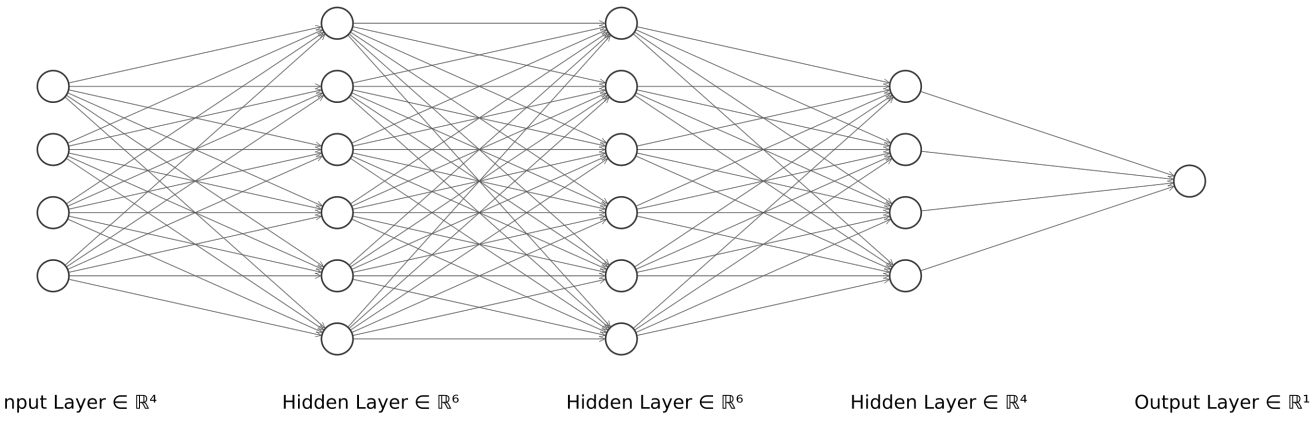

Figure 2: Proposed Neural Network

Calibration Dataset: It contains samples of tuples (n,λ) where 'n' is the number of monomer units and 'λ' is the distance between the two molecules. Let the sample size of the calibration dataset be denoted by 'N_c'. Also, we assume that for each value of 'λ' we have 'l_c' sample points.

Proposed Neural Network: For a given spatial length, we can find the approximated probability distribution for the number of segments using the Kernel Density Estimator applied to the samples obtained from the calibration dataset. Now, we need to generalize this distribution for any length (λ). This is done by training a neural network. Thus, we have the inputs as the number of segments ('n') and the spatial length between any two molecules (λ) and we get an output which is proportional to the probability of getting 'n' number of segments between the two molecules given the length between them (λ). The conditional probability density is obtained by normalizing this value. Thus, the network will learn P(n|λ).
The activation function has been used as RELU as opposed to the sigmoid activation function used widely in relevant literature. The main reason of this being as the output is reaching high values, the derivative is almost vanishing at intermediate steps leading to very low updation of weights. Also, it is preferred over widely used LeakyRELU as we are interested in obtaining positive output values. Moreover, to ensure positive output values, the weights are also clamped initially to a minimum value of 0.001.

Thus, the neural network structure is summarized as,
Input units: n, n^2, n^3, λ (the input units can be extended to higher powers of 'n' for manually introducing more non-linearity.
Output units: N(n, λ). It is proportional to the probability of getting 'n' monomer units given the length of the polymer.

4.2 Calculating the expected output of the neural network:

We first consider that the probability of polymers with n > N is very low and can be ignored. This is also a hyperparameter and can be fixed appropriately.
Now we create an array 'P' which stores the expected output of the neural network for given value of λ.
The expected output is calculated using the Kernel Density Estimation using a Gaussian Kernel with an appropriate smoothing hyperparameter.
Thus,

$$P(i|\lambda) = \frac{1}{\sqrt{(2 \cdot \pi \cdot k)}} \sum_{i,s.t.[l_i=\lambda]} e^{\frac{(n_i - i)^2}{2\sigma^2}} \tag{3}$$

Where 'k' is the total number of 'i' s.t. $[l_i = \lambda]$.
Similar arrays will be created for different values of λ

4.3 Training

The labeled data (n,λ) is provided as input to the neural network and the corresponding weights are updated by calculating dL/dw_i with the help of an appropriate loss function. We consider squared loss for our method.
After training, the conditional probability is calculated as,

$$P(n|\lambda) = \frac{(N(n, \lambda))}{(\sum_{i=0}^{i=100} N(i, \lambda))} \tag{4}$$

5 Imaging Phase

5.1 For two molecules

Imaging Dataset: We first consider our problem to estimate the length between two points and then extend it to our main problem of estimating spatial coordinates of a group of points.
Consider two points for which we are given samples of number of monomer units. Thus, our imaging dataset consists of a set of (n) where 'n' is the number of monomer units.

Estimating length between the molecules: We use the previously trained neural network which thus can correctly estimate P(N|λ). As optimum λ needs to be estimated, the input of the neural network is calculated using 'n' and an initial guess for the distance, λ.
In this phase, we calculate the loss function by taking the log-likelihood of the output conditional probability from the neural network. Finally, optimum length can be found using gradient descent or any other optimization technique.

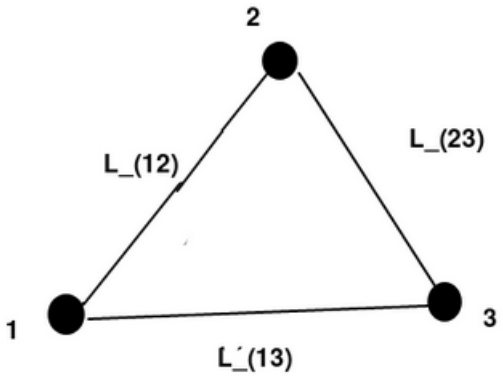

Figure 3: 2D spatial location estimate for 3 points

5.2 Extending to more molecules

For the main task of estimating the spatial coordinates, our input dataset would now consist of sets (for every possible pair of points for which data can be obtained) of samples. For 'p' points we will have maximum $\frac{p^2-p}{2}$ distances. Further, each distance can be estimated similar for the two points case.
For converting distances to coordinates we arrange the predicted distances in a (nxn) distance array (D) where $D_{i,j}$ represents the predicted distance between the 'i'th and 'j'th point. MDS is used on this matrix which provides the spatial coordinates of all the points.
Note that the spatial coordinates obtained will not be unique. This is because in the 2-dimensional case, the molecules would have 3 degrees of freedom (2 translation and 1 rotation). Similarly, we can observe different coordinate locations in the case of 3 dimensional spread of the molecules.

6 Simulations:

References

[1] Guy Hadash, Einat Kermany, Boaz Carmeli, Ofer Lavi, George Kour, and Alon Jacovi. Estimate and replace: A novel approach to integrating deep neural networks with existing applications. *arXiv preprint arXiv:1804.09028*, 2018.