Enlitic

Work Sample Report

Kernel density estimation with Mixture of Gaussians

Sudhir Sornapudi
Department of Computer Engineering
Missouri University of Science and Technology
Email: ssbw5@mst.edu

March 31, 2017

Contents

1 Introduction

Kernel density estimation is a method to estimate probability density function, pdf, of a random variable. This work uses a set of Gaussian kernels to make the density estimation. Hence, the kernel density estimator is formulated with a mixture of Gaussian distributions.

The given problem is an unsupervised learning technique, that is the algorithm needs to fit the model from dataset consisting of only inputs without ground truth labels.

The model is trained on given train dataset, so that the model learns from data and fits the parameters. The validation dataset is required to tune the parameters to minimize over-fitting. Finally the overall performance is evaluated on a separate test dataset by predicting the trained model.

2 Model

The given dataset contains two splits $\mathcal{D}_\mathcal{A} \in R^{k \times d}$ and $\mathcal{D}_\mathcal{B} \in R^{m \times d}$, the log-likelihood of $\mathcal{D}_\mathcal{B}$ under $\mathcal{D}_\mathcal{A}$ is computed with following probability density function

$$\log p(x) = \log \sum_{i=1}^{k} p(z_i) p(x|z_i) \tag{1}$$

where $x \in R^d$ and z_i is discrete.

This is a conditional probability where $p(z_i)$ is the probability of i-th Gaussian mixing component and $p(x|z_i)$ is the probability of x under i-th Gaussian mixing component. With the considered assumption,

$$p(z_i) = \frac{1}{k} \tag{2}$$

and

$$p(x|z_i) = \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp(-\frac{(x_j - \mu_{i,j})^2}{2\sigma_i^2}) \tag{3}$$

and making σ constant, and $\mu \in \mathcal{R}^{k\times d}$, Equ. (1) can be written as

$$\log p(x) = \log \sum_{i=1}^{k} \exp\{\log \frac{1}{k} + \sum_{j=1}^{d} [-\frac{(x_j - \mu_{i,j})^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma^2)]\} \tag{4}$$

the equation is further simplified as

$$\log p(x) = -\frac{d}{2}\log(2\pi\sigma^2) - \log(k) + \log \sum_{i=1}^{k} \exp\{\sum_{j=1}^{d} [-\frac{(x_j - \mu_{i,j})^2}{2\sigma^2}]\} \tag{5}$$

The Equ. (4) is simplified to eliminate the problem with exponential of larger negative values because of the $-\frac{1}{2}\log(2\pi\sigma^2)$ term.

The log-probability of each sample in $\mathcal{D}_B$ is computed using Equ. (5) with given condition of $\mu_{i,j} \equiv x_{i,j}^A$ where $x_A \equiv \mathcal{D}_A \in \mathcal{R}^{k\times d}$. Finally the mean of the log-probability on $\mathcal{D}_\mathcal{B}$ is calculated using

$$\mathcal{L}_{\mathcal{D}_\mathcal{B}} = \frac{1}{m}\log \prod_{i=1}^{m} p(x_i^B) = \frac{1}{m}\sum_{i=1}^{m} \log p(x_i^B) \tag{6}$$

3 Data Visualization

The datasets provided for the modeling are MNIST and CIFAR100. MNIST are a set of 28×28 grayscale images as shown in Figure 1 while CIFAR100 contains $32 \times 32 \times 3$ RGB images as shown in Figure 2.

The visualization of images from both the datasets are created by reshaping a set of 400 randomly chosen flattened numpy array training image dataset.

The figures shown below verify correctness of preprocessing for both datasets. The visualization code is optimized by eliminating for loops to represent the images in a 20×20 grid structure.

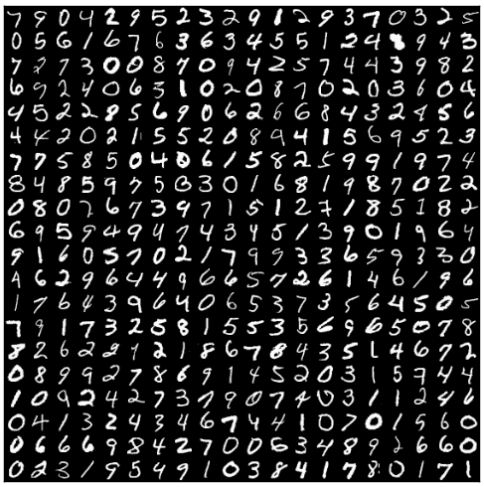

Figure 1: MNIST

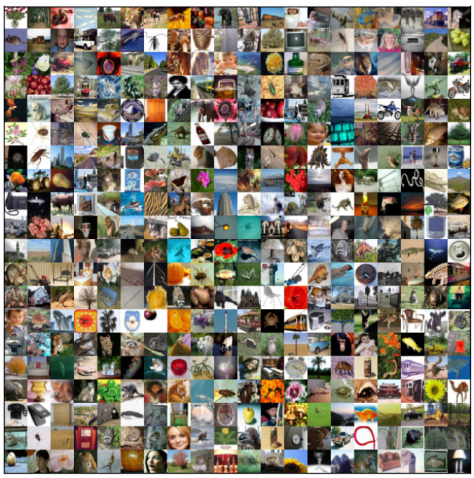

Figure 2: CIFAR100

4 Experimental Results

The MNIST and CIFAR100 datasets are preprocessed according to the problem statement. A grid search is implemented to find the optimal σ for the model. The experiment was performed with $\sigma = \{0.05, 0.08, 0.1, 0.2, 0.5, 1.0, 1.5, 2.0\}$ on both MNIST and CIFAR100 datatsets with ($\mathcal{D}_A = MNIST_{train}, \mathcal{D}_B = MNIST_{valid}$) and ($\mathcal{D}_A = CIFAR_{train}, \mathcal{D}_B = CIFAR_{valid}$). The results are represented in a tabular form as shown in Table 1.

Increase in σ has lead to an increase in the log-likelihood value. After reaching a peak value, a decrease in value of log-likelihood is observed by further increasing the σ, forming a bell-shaped curve. $\sigma = 0.2$ is considered to be the optimal value, because the objective is to maximize log-likelihood, so the higher log-likelihood value is better. The maximum value of log-likelihood for MNIST, $\mathcal{L}_{\mathcal{D}^{MNIST}_{valid}} = 233.217$ and for CIFAR, $\mathcal{L}_{\mathcal{D}^{MNIST}_{valid}} = 862.514$ is observed when $\sigma = 0.20$.

Table 1: Results on MNIST and CIFAR100 validation datasets

σ	$\mathcal{D}^{MNIST}_{valid}$	$\mathcal{D}^{CIFAR}_{valid}$
0.05	-3164.25	-13621.160
0.08	-617.928	-2882.506
0.10	-120.228	-756.633
0.20	**233.217**	**862.514**
0.50	-233.895	-902.701
1.00	-740.899	-2881.853
1.50	-1051.083	-4099.274
2.00	-1272.574	-4972.661

The optimal value of σ is considered to compute log-likelihood, $\mathcal{L}_{\mathcal{D}_\mathcal{B}}$ on test data where ($\mathcal{D}_A = MNIST_{train}, \mathcal{D}_B = MNIST_{test}$) and ($\mathcal{D}_A = CIFAR_{train}, \mathcal{D}_B = CIFAR_{test}$). The results show that for MNIST, $\mathcal{L}_{\mathcal{D}^{MNIST}_{test}} = 234.269$, and for CIFAR, $\mathcal{L}_{\mathcal{D}^{CIFAR}_{test}} = 850.468$ when computed on the optimal σ that is 0.20. The running time is also benchmarked as shown in Table 2. The run time is tabulated by calculating the average run-time on five test runs for each dataset. Its is observed that running the model on MNIST dataset is 1.56x times faster than CIFAR100 dataset. This obvious because MNIST data contains 28×28 grayscale images, where as CIFAR100 data contains $32 \times 32 \times 3$ RGB images.

Table 2: Results on MNIST and CIFAR100 test datasets with optimal σ

	$\sigma_{optimal} = 0.20$	Average run-time
$\mathcal{D}_{test}^{MNIST}$	234.269	10152.91 seconds
$\mathcal{D}_{test}^{CIFAR}$	850.468	15903.13 seconds

The code to compute mean of log-probability is written in python v2.7 and ran on CPU with Intel Core i5 2400 processor, 8GB DDR3 RAM and 512MB AMD RADEON HD 6350 graphics card for analyzing the results.

5 Conclusion

The mean of log-likelihood is computed on given datasets using kernel density estimation based on mixture of Gaussians distributions. The standard deviation (σ) corresponding to a Gaussian component is considered optimal when there is a peak value observed in the computation of log-likelihood. This work is a maximum likelihood estimation on given datasets. The realized model has successfully found out the optimal standard deviation of Gaussian kernel on the given MNIST and CIFAR100 datasets.

The model is coded in CPU-based Python and Numpy environments using their standard libraries (math, decimal, cPickel, csv, time, argparse and sys modules). Matplotlib is used for visualizing the datasets provided. The 'decimal' module with a precision of 7 was helpful and avoided the problem with using 'float64' for handling high precision floating point arithmetics.

The model was optimized by vectorizing the data in order to minimize the "for loops" to make the code run faster. To further optimize the code one can use multiprocessing module to make use of all the processors on-board for parallel processing.

The kernel density estimation with mixture of Gaussians is one of the simplest model to construct a probabilistic model, but in general the model is said to be biased when the data is bounded.