UNIVERSITÀ
DI SIENA 1240

Gaussian Mixture Neural Network

Meconcelli Duccio
Pindali Mana Aparna

Project for ARTIFICIAL INTELLIGENCE exam (2022-2023)
Professor Trentin Edmondo

2023/07

Abstract

The aim of our experiment is to determine the efficacy of combining the Gaussian mixture model with a neural network in estimating a probability distribution function, building upon the framework of the Parzen Neural Network. In this investigation, we use this model to predict the nature of the probability density distribution using data samples from an exponential distribution.

Our approach leverages the statistical power of Gaussian Mixture Models (GMM) in modeling complex distributions and the ability of Multilayer Perceptrons (MLPs) to capture intricate nonlinear relationships. We present a detailed analysis of the differences between the base GMM and our hybrid algorithm, highlighting the advantages of our proposed method.

The best results were obtained using the new algorithm with a Gaussian mixture model with 32 components. We also perform the experiment to compare biased and unbiased versions of the model and in general the unbiased version produced overall more accurate results than the biased ones.

All the code of this project can be found on this GitHub Page [1].

Contents

1 Introduction

The estimation of probability density functions (PDFs) from unlabeled data is a fundamental problem in machine learning and plays a crucial role in various domains, including pattern recognition, anomaly detection, and clustering.

Traditional statistical approaches, such as Gaussian Mixture Models which combines multiple Gaussian distributions to estimate pdf, have been widely used for this purpose.

However, these models often struggle to accurately capture complex and multimodal distributions. On the other hand, neural networks, specifically Multilayer Perceptrons, excel at learning intricate nonlinear relationships but may lack the statistical robustness of traditional models and are not suited for unsupervised learning.

In this report, we take a look on a hybrid algorithm, inspired by the Parzen Neural Network (PNN), that combines the strengths of both approaches to improve the accuracy and flexibility of PDF estimation from unlabeled data.

The algorithm follows what happens in the PNN that uses a variant of the Parzen Window (PW) method. In every iteration of the feed-forward algorithm the leave-one-out rule is used to take out the pattern over which the PW is applied, and a new training set is created. Since a selected sample is not included the estimation of its own target, the PNN is an unbiased model.

In our experiment, we replace the PW with a GMM to estimate the PDF. The GMM is a mixture density where the PDF is estimated as a linear combination of many weighted component pdfs. The experiment is carried out with different component values for the GMM.

2 The Algorithms

In this section, we provide a description of the Gaussian Mixture Model (GMM) and the GMM + NN algorithm. We present the mathematical formulations and the steps involved in each algorithm.

2.1 Gaussian Mixture Model

A Gaussian Mixture Model represents (GMM) a probability density function (PDF) as a weighted sum of Gaussian component densities. A GMM is defined as follows:

$$p(\vec{x}|\vec{\mu_k}, \boldsymbol{\Sigma_k}) = \sum_{k=1}^{K} P(w_k) \cdot \mathcal{N}(\vec{x}|\vec{\mu_k}, \boldsymbol{\Sigma_k}) \tag{1}$$

where $p(\vec{x}|\vec{\mu_k}, \boldsymbol{\Sigma_k})$ is the PDF, K is the number of mixture components, $P(w_k)$ is the probability associated with each component and it is the k-th mixing parameter (satisfying $\sum_{k=1}^{K} P(w_k) = 1$), μ_k is the mean vector, and Σ_k is the covariance matrix for the k-th component. The Gaussian density function $\mathcal{N}(x|\mu_k, \boldsymbol{\Sigma_k})$ is given by:

$$\mathcal{N}(x|\vec{\mu_k}, \boldsymbol{\Sigma_k}) = \frac{1}{(2\pi)^{\frac{D}{2}} \cdot |\boldsymbol{\Sigma_k}|^{\frac{1}{2}}} \cdot \exp\left(-\frac{1}{2}(x - \vec{\mu_k})^T \boldsymbol{\Sigma_k}^{-1}(x - \vec{\mu_k})\right) \tag{2}$$

where D is the dimensionality of the data, and in our case equal to 1.

The parameters of the GMM, including $P(w_k)$, $\vec{\mu_k}$, and $\boldsymbol{\Sigma_k}$, are estimated using the Maximum likelihood (ML) estimation algorithm. We use following iterative algorithm (gradient ascent):

- Initialize the parameters ($P(w_k)$, $\vec{\mu_k}$, and $\boldsymbol{\Sigma_k}$) using random initialization methods or Kmeans approaches [2].
- Update the mixing parameter ($P(w_k)$):

$$P(w_k)^{(t+1)} = \frac{\sum_{i=1}^{N} P(w_i|\vec{\mu_k}, \boldsymbol{\Sigma_k})^{(t)}}{N} \tag{3}$$

- Update the mean vector ($\vec{\mu_k}$):

$$\vec{\mu_k}^{(t+1)} = \frac{\sum_{i=1}^{N} P(w_i|\vec{\mu_k}, \boldsymbol{\Sigma_k})^{(t)} \cdot \mathbf{x}_i}{\sum_{i=1}^{N} P(w_i|\vec{\mu_k}, \boldsymbol{\Sigma_k})^{(t)}} \tag{4}$$

- Update the covariance matrix ($\boldsymbol{\Sigma}$):

$$\boldsymbol{\Sigma}_k^{(t+1)} = \frac{\sum_{i=1}^{N} P(w_i|\vec{\mu_k}, \boldsymbol{\Sigma_k})^{(t)} (\mathbf{x}_i - \vec{\mu_k}^{(t+1)})(\mathbf{x}_i - \vec{\mu_k}^{(t+1)})^T}{\sum_{i=1}^{N} P(w_i|\vec{\mu_k}, \boldsymbol{\Sigma_k})^{(t)}} \tag{5}$$

where:

- t represents the iteration step
- N is the total number of data points
- k denotes the mixture component
- $P(w_i|\vec{\mu_k}, \boldsymbol{\Sigma_k})^{(t)}$ is the responsibility (target probability) for component k at iteration t

Note that these formulas are used iteratively until convergence, updating the parameters in each iteration.

2.2 GMM + NN Algorithm

The GMM + NN algorithm combines the GMM with a Neural Network (NN) to estimate the PDF of unlabeled data.

Our algorithm follows a two-step process. First, we utilize a Gaussian Mixture Model to approximate the underlying distribution of the unlabeled data. This step allows us to generate synthetic targets for supervised training of a Multilayer Perceptron. The synthetic targets are created by sampling from the GMM and assigning labels to each data point. In the second step, we train the MLP using the synthetic targets and the original unlabeled data as inputs. The MLP learns to map the input data to the synthetic target labels, effectively estimating the PDF of the unlabeled data.

Given a K components for the GMM and N training samples from a PDF, the GMM + NN algorithm involves the following steps:

Step 0: Generate Training set for GMM

For each data x_i present in the starting dataset (i ranging from 1 to N) a new training set (with size $(N-1)$ elements) is extracted which includes all the N elements except the i-th element (*unbiased dataset*).

Step 1: GMM Training

- For each new taining set, generated in Step 0, initialize the GMM parameters: $P(w_k)$, $\vec{\mu_k}$, and $\boldsymbol{\Sigma_k}$.
- Estimate the parameters using the Maximum likelihood (ML) estimation algorithm.
- For each data point x_i, compute the PDF (target y_i) using the proper i-th dataset as follows:

$$y_i = p(\vec{x}|\vec{\mu_k}, \boldsymbol{\Sigma_k}) = \sum_{k=1}^{K} P(w_k) \cdot \mathcal{N}(\vec{x}|\vec{\mu_k}, \boldsymbol{\Sigma_k}) \tag{6}$$

At the end of Step 1 a supervised training set for the Neural Network set is composed using x_i as input and y_i as target, with $i = [1, ..., N]$.

Step 2: NN Training

- Train a Multilayer Perceptron (MLP) using the data points x_i as input and the corresponding target y_i as the desired output.
- The MLP learns to map the input data to the assigned targets using backpropagation and gradient-based optimization techniques for update the MLP's weights and biases on a supervised learning approach.

To estimate the PDF for a new data point x, pass it through the trained MLP.

3 Experiments and Results

In this section, we describe how the experiments was conducted to evaluate the performance of the proposed GMM + NN algorithm for estimating the PDF of unlabeled data. We also present the results obtained from these experiments, comparing different variations of the algorithm and using various evaluation metrics.

We use four different values for the number of components (4, 8, 16 and 32) and for each setting, the pdf is estimated using the GMM and the GMM + NN.

Furthermore, in the end a comparison was also made with the Biased version which, unlike the unbiased version seen so far, does not eliminate the i-th element of the GMM training set to generate the targets.

3.1 Datasets

First the *Training set* for the experiment was generated with known PDF. The dataset is composed of 100 points which were selected randomly from an exponential distribution with rate fixed to 0.6.

For the *Test set* sample points were selected ranging form 0 to 10 with a constant step of 0.001 and then each point was evaluated on the exponential PDF.

3.2 Gridsearch Procedure

We performed a grid search on the Multilayer Perceptron (MLP) architecture to find the optimal configuration for the neural network. The grid search involved varying parameters such as the number of hidden layers, the number of neurons per layer, and the activation functions. The search was guided by maximizing the R2-score, which is a commonly used metric for regression tasks.

The R2-score measures the proportion of the variance in the dependent variable that is predictable from the independent variables. It is calculated as:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{7}$$

where N is the number of samples, y_i is the true value, $\hat{y}_i$ is the predicted value, and $\bar{y}$ is the mean of the true values. The Gridsearch parameters are as follows:

```
    mlp_params = {
        "criterion": [MSELoss],
```

```
        "max_epochs": [50, 80],
        "batch_size": [4, 8, 16],
        "lr": [0.005, 0.01],
        "optimizer": [Adam],
        "module__n_layer": [1, 2, 3],
        "module__last_activation": ["lambda", ReLU],
        "module__num_units": [80, 50, 10],
        "module__activation": [ReLU],
        "module__type_layer": ["increase", "decrease"],
        "module__dropout": [0.3, 0.5, 0.0],
    }
```

Listing 1: parameters for the gridsearch from `main.py` file

- `module__n_layer` : set the number of hidden layers.
- `module__last_activation` : set the last activation function to be applied to the model, if *lambda* is specified than it will be applied a sigmoid function with adaptive amplitude parameter *lambda* [3].
- `module__num_units` : set the number of neurons for the first layer of the model.
- `module__activation` : set the activation function for all the layers.
- `module__type_layer` : set how the layers should behave. With *"increase"* at each layer the number of neurons increases by a factor of 2. With *"decrease"* at each layer the number of neurons decrease by a factor of 2.

To assess the generalization performance of the algorithm, we employed 5-fold cross-validation. The dataset was split into five subsets, with each subset serving as a validation set once while the rest of the data was used for training. This process was repeated five times to obtain reliable performance estimates.

3.3 Evaluation Metrics

To compare the different variations of the algorithm, we analyzed the results using both graphical and quantitative methods. Graphs were plotted to visualize the predicted PDFs and compare them to the true PDFs, providing an intuitive understanding of the performance.

Additionally, a set of evaluation metrics was used to compare the results of our hybrid algorithm with those obtained using the base Gaussian Mixture Model and the true exponential PDF, allowing for a comprehensive assessment. These metrics included:

- R2 Score: As described earlier, the R2 score measures the goodness of fit between the predicted PDF and the true PDF.
- Mean Squared Error (MSE) Score: The MSE score measures the average squared difference between the predicted and true PDFs. It is calculated

as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{8}$$

where y_i represents the true values and $\hat{y}_i$ represents the predicted values.

- Kullback-Leibler (KL) Divergence Score: The KL divergence score quantifies the difference between two probability distributions, in this case, the predicted and true PDFs. It quantifies the information lost when one distribution is used to approximate the other.

  KL divergence is calculated as:

$$KL(P||Q) = \sum_{i} P(i) \log\left(\frac{P(i)}{Q(i)}\right) \tag{9}$$

  where P and Q represent the true and estimated PDFs, respectively.
- Maximum Error (Max Error) Score: The Max Error score measures the maximum absolute difference between the predicted and true PDFs.
- Integrated Square Error (ISE) Score: The ISE score computes the integrated squared difference between the predicted and true PDFs. It is calculated as:

$$ISE = \int (p(x) - \hat{p}(x))^2 dx \tag{10}$$

  where $p(x)$ represents the true PDF and $\hat{p}(x)$ represents the estimated PDF.

3.4 Results

The following results show some figures for a visual comparison. Each image consists of:

- in *gray* the histogram relating to data sampling.
- the real PDF of the exponential distribution is in *dotted green*
- in *blue* the PDF predicted by the basic GMM
- in *red* the PDF predicted by the new GMM + MLP algorithm

Each image is also associated with a unique *id* such as to be able to associate the images with the results shown in Table 1 and in the `.csv` file present in the repository [1].

Table 1 presents some results of the comparative analysis using the evaluation metrics. The GMM + NN (labeled as **MLP** in the Model Type column) algorithm consistently achieves higher R2 scores, lower MSE scores, smaller

ISE scores, and lower Max Error scores compared to the regular GMM approach. These metrics indicate the superior accuracy and precision of the GMM + NN algorithm in estimating the PDF of unlabeled data.

3.4.1 Different Number of Components

In general 4 and 32 components (respectively figure 1 and 6) for the GMM + MLP perform better than 8 and 16 components (figure 2). On the other hand the basic GMM models perform better with few components (4 and 8) as it was expected (It can be seen from the *blue lines* from figures 1, 2a compared to figures 2b). The performance difference between the base Gaussian Mixture Model (GMM) and the GMM + Multilayer Perceptron (MLP) algorithm with different numbers of components can be attributed to a trade-off between model complexity and the amount of available data.

When using a large number of components in the base GMM, the model becomes more complex and has a higher capacity to capture intricate data distributions. However, this increased complexity can also lead to overfitting, especially when the amount of available data is limited. Overfitting occurs when the model becomes too specialized to the training data and fails to generalize well to unseen data.

In contrast, the GMM + MLP algorithm combines the statistical modeling capabilities of the GMM with the learning abilities of the MLP, enabling the model to capture complex data distributions effectively. However, the MLP component of the algorithm introduces additional learning capacity, allowing the model to adapt and learn from the available data. In the case of the GMM + MLP algorithm, using 32 and 4 components may strike a balance between model complexity and the amount of data available. With 32 components, the GMM can still capture complex data distributions effectively, while the MLP can leverage the synthetic targets generated by the GMM to learn the underlying patterns in the data. With 4 components, the model's complexity is reduced, enabling the GMM + MLP algorithm to focus on learning and generalizing from the available data, resulting in improved performance.

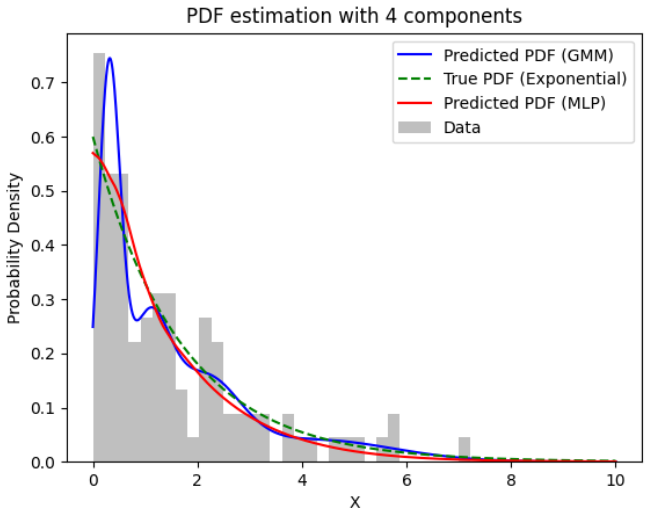
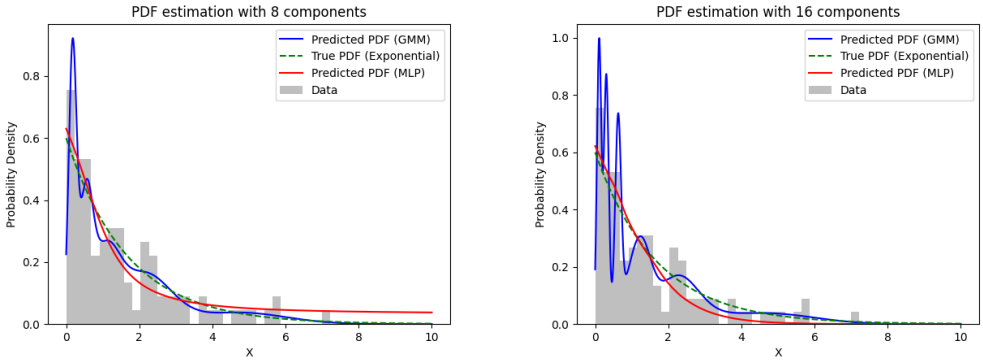


Figure 1: best result with 4 components *(red line)*, random initialization (id = f9c)

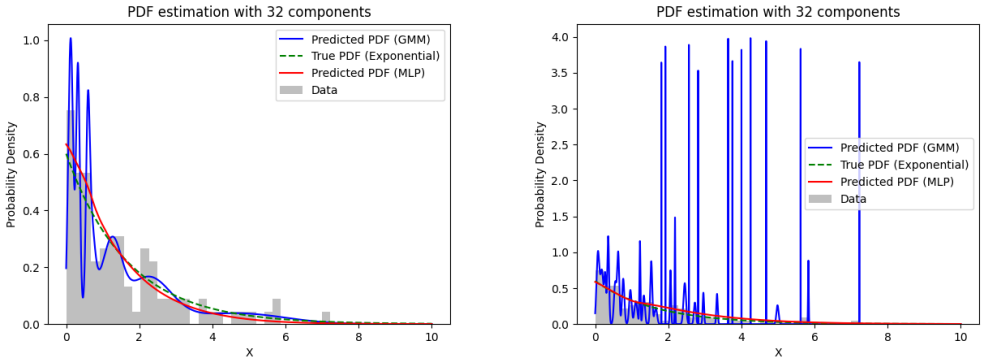


(a) 8 components, random init. (id = 7fb) **(b)** 16 components, kmeans init. (id = fe5)

Figure 2: Best result with 8 components (**a**) and 16 components (**b**)

3.4.2 GMM initialization

Another interesting result concerns the initialization of the parameters of the GMM: if on one hand the variation of the method of initialization of these parameters does not have conspicuous results as regards the algorithm GMM + NN, this cannot be said of the basic GMM where initialization plays a fundamental role as you can see from the figure 3 *(blue lines)* where random initialization performs much better than kmeans. On the other hand, the generalizing factor of the MLP manages to attenuate this performance gap of the GMM.



(a) 32 components, random init. (id = 470) **(b)** 32 components, kmeans init. (id = 3f4)

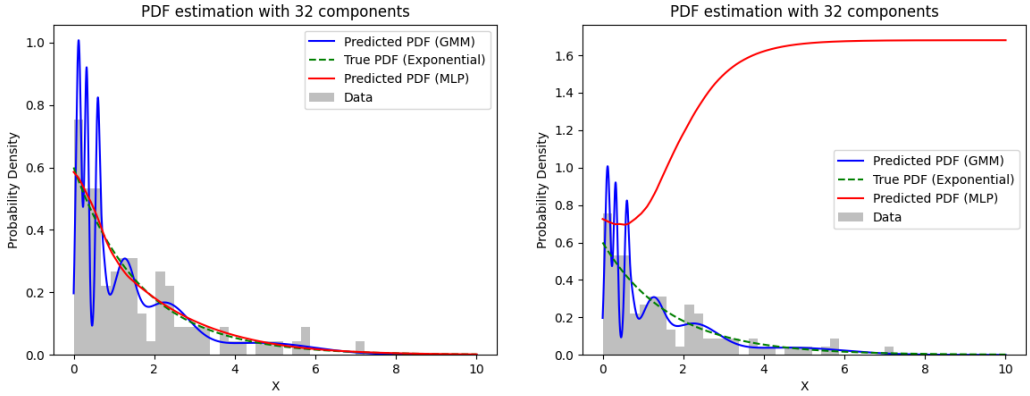
Figure 3: Best result with 8 components (**a**) and 16 components (**b**)

3.4.3 Biased vs Unbiased

The biased version, particularly when using k-means initialization for the GMM parameters, yielded inferior results compared to the unbiased version. The biased results exhibited significant deviations from the true PDF, emphasizing the importance of avoiding biased target generation for the MLP component.

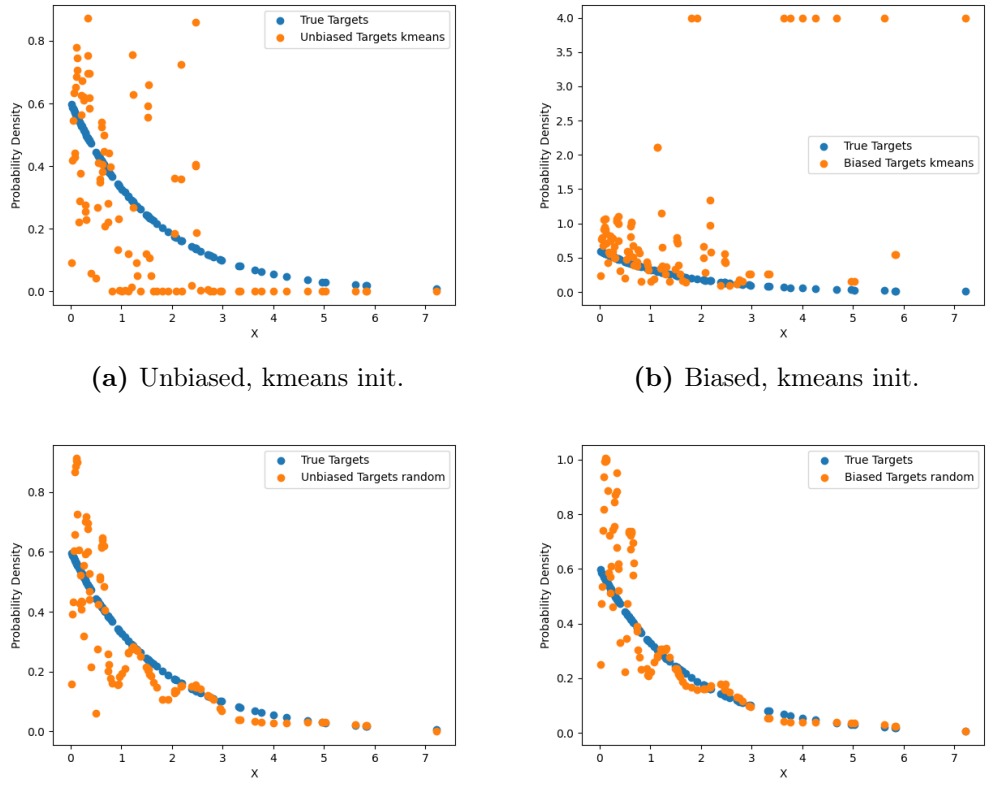
Unbiased version has better results than the biased version as expected. The biased results are particularly inaccurate in this scenario of Fig. 4 and the predicted graph is vastly different from the true pdf.

For a visual comperason we also plotted the GMM outputs used as the MLP target in the GMM + NN algorithm (figure 5). From the graphs it can be seen that as regards the random initialization there is not much difference between the biased and the unbiased version, it is possible to notice only an increase in the values of the predicted PDF as regards the biased version, which is understandable because the single value more than the unbiased version in the training set approximates the mean at the point of trying to predict pdf. On the other hand, initialization with kmens leads to an evident sharp increase in the values of the biased version compared to the unbiased one, especially for values far from the peak of the distribution. This is probably due to the initialization with kmeans which clustering the parameters of the Gaussians in a few single points then fails to move these parameters from the clustered points when applied the iterative algorithm.



(a) unbiased *red line* **(b)** biased red line

Figure 4: Differences between the bias and ubiased version of GMM + NN with 32 components with kmeans initialization, id = 926



(a) Unbiased, kmeans init. **(b)** Biased, kmeans init.

(c) Unbiased, random init. **(d)** Biased, random init.

Figure 5: Target Output from the GMM before start the MLP training, 32 components

3.4.4 Best Result

Fig 6 shows the graph with 32 components, initialized with kmeans. The predicted PDF for this MLP is the closest to the true pdf in these results scoring 0.997 with R2-score. As expected [3], the winning architecture resulting from the grid search uses a trainable amplitude of activation functions as the last activation function, this type of function is always the winner across all the experiments. Counterintuitively the architecture uses a lot of neurons as the starting point of the first hidden layer and these increase in the second and last hidden layer by a multiplicative factor of 2 creating a very large network for this type of problem.

The architecture for the best result is show with the following algorithm snippet:

,

```
    mlp_params = {
        'batch_size': 4,
        'criterion': 'MSELoss',
        'lr': 0.005,
        'max_epochs': 50,
        'module__activation': ReLU,
        'module__dropout': 0.3,
        'module__last_activation': 'lambda',
        'module__n_layer': 2,
        'module__num_units': 80,
        'module__type_layer': 'increase',
        'optimizer': Adam
    }
```

Listing 2: parameters for the best model

Further experiments and analyses can be conducted to validate these findings on a wider range of datasets and problem domains.

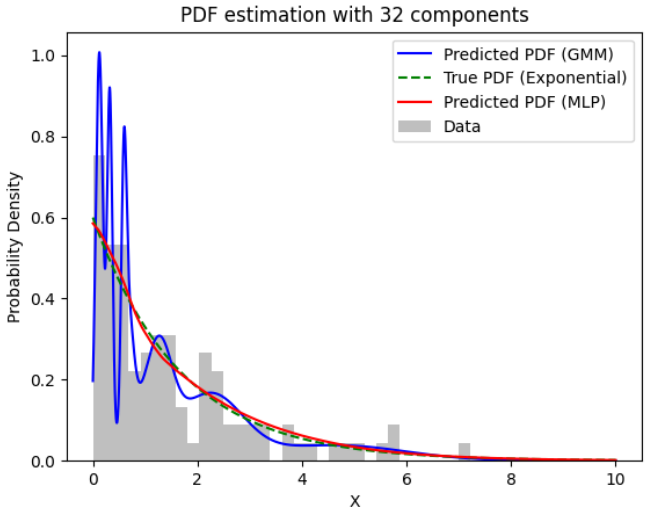


Figure 6: best result (red line), initialized with kmeans (id = 926)

Table 1: top 9 and worse 9 results (R.F.D. = Random From Data)

Id	**# Comp.**	**Model Type**	**R2**	**MSE**	**Max Error**	**ISE**
926	32	MLP kmeans	0.997	0.007	0.025	0.005
9ac	32	MLP random	0.992	0.012	0.025	0.016
d1f	32	MLP R.F.D.	0.988	0.015	0.048	0.023
d42	32	MLP kmeans++	0.984	0.017	0.071	0.0301
751	32	MLP random	0.983	0.018	0.051	0.0334
d77	4	MLP kmeans	0.982	0.018	0.065	0.0342
470	32	MLP random	0.981	0.019	0.067	0.0375
3bb	4	MLP kmeans++	0.980	0.019	0.061	0.0384
3f4	32	MLP kmeans	0.978	0.020	0.044	0.0426
...	...	...	...	...	...	...
5fe	4	GMM random	0.886	0.048	0.351	42986
808	16	GMM random	0.742	0.071	0.433	48267
470	32	GMM random	0.705	0.076	0.45	48906
930	8	GMM kmeans	0.631	0.085	3.61	49776
3f4	32	GMM kmeans	-1.526	0.22	3.93	94053
d1f	32	MLP R.F.D. Biased	-4.51	0.33	1.036	11.05
926	32	MLP kmeans Biased	-102	1.44	1.679	208.4
823	32	MLP kmeans Biased	-115	1.53	1.93	234.4
d17	32	MLP kmeans++ Biased	-135	1.65	1.94	273.3

4 Conclusion

In our experiments, we presented an algorithm for estimating the PDF of unlabeled data by combining Gaussian Mixture Models and Multilayer Perceptrons. Our hybrid approach takes advantage of the statistical robustness of GMMs and the nonlinear modeling capabilities of MLPs. Experimental results showcased the superior performance of our algorithm compared to the base GMM in terms of PDF estimation accuracy. The GMM + NN algorithm leverages the strengths of both statistical modeling and neural networks, resulting in improved performance. This finding supports the idea that combining statistical approaches with powerful learning capabilities can lead to better modeling and estimation outcomes.

Among the variations of the GMM + NN algorithm tested, the best results were obtained when using 32 components for the GMM used to generate the targets for the MLP. This configuration allowed for a more fine-grained modeling of the data distribution, capturing its inherent complexity effectively.

Additionally, we observed that using a biased version for generating the target for the MLP yielded worse results compared to the unbiased version. This observation was particularly evident when initializing the GMM parameters using the k-means algorithm. The biased version resulted in inaccurate PDF estimations, with predicted graphs significantly deviating from the true PDF.

It is worth noting that the results presented in this report are specific to the datasets and experimental setups used. Further investigation on different PDF types and problem domains is necessary to validate the generalizability and scalability of the GMM + NN algorithm.

References

[1] D. Meconcelli and A. Pindali Mana, "Gaussian-mixture-neural-network," GitHub, 07 2023. [Online]. Available: https://github.com/Duccioo/Gaussian-Mixture-Neural-Network

[2] "Gmm initialization methods — scikit-learn 1.3.0 documentation," https://scikit-learn.org/stable/auto_examples/mixture/plot_gmm_init.html, (Accessed on 07/10/2023).

[3] E. Trentin, "Networks with trainable amplitude of activation functions," *Neural Networks*, vol. 14, no. 4-5, pp. 471–493, 2001.