Project Final Report

BY YALIN LIAO

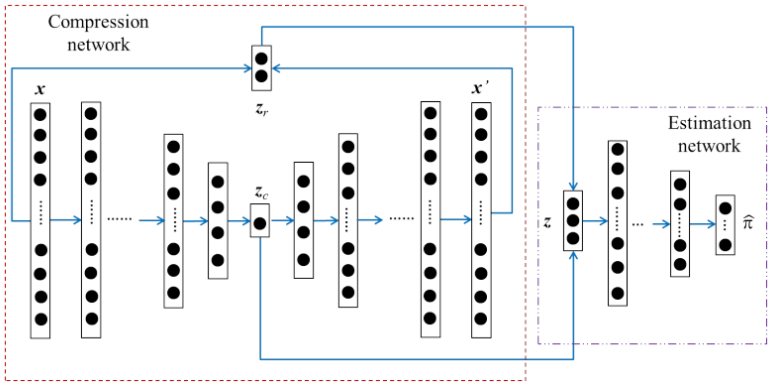
Abstract

This project is to apply the deep autoencoding Gaussian mixture model (DAGMM) in [4] to evaluate physical rehabilitation exercises. First, we will introduce DAGMM. Then we will describe the rehabilitation exercise dataset UI-PRMD [3], used to validate DAGMM. Next, we implement DAGMM on UI − PRMD and compare the results with those in [1].

1 DAGMM

Traditionally, using Gaussian Mixture model (GMM) to fit high dimensional data we need to apply the dimensionality reduction PCA or Autoencoder and then train GMM on the dimensionality-reduced dataset. Instead of using decoupled two-stage training, the paper presents an end-to-end framework DAGMM , which jointly optimizes the parameters of the deep autoencoder and the mixture model simultaneously. We will then compare results by three methods.

1.1 Overview of DAGMM



DAGMM consists of two major components: a compression network and an estimation network. As shown in the figure, the compression networks performs dimensionality reduction for input samples by an autoencoder and the estimate network takes the low dimensional representation of input data to predict the likelihood of the input data by a Gaussian mixture model.

1.2 Compression Network

Given a sample $\boldsymbol{x}$, the compression network computes its low-dimensional representation $\boldsymbol{z}$ as follows

$\boldsymbol{z}_c = h(\boldsymbol{x}; \theta_e)$	$h(\cdot; \theta_e)$ denotes the encoding function, parameterized by θ_e
$\boldsymbol{x}' = g(\boldsymbol{z}_c; \theta_d)$	$g(\cdot; \theta_d)$ denotes the decoding function, parameterized by θ_d $\boldsymbol{x}'$ is the reconstructed counterpart of $\boldsymbol{x}$
$\boldsymbol{z}_r = f(\boldsymbol{x}, \boldsymbol{x}')$	$f(\cdot)$ denotes the function of calculating reconstruction error $\boldsymbol{z}_r$ is the reconstruction error, which can be multi-dimensional
$\boldsymbol{z} = [\boldsymbol{z}_c, \boldsymbol{z}_r]$	low dimensional representation of $\boldsymbol{x}$, feed to estimation network

The compression network is trained by minimizing the average reconstruction error[1]

$$\frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{x}_i, \boldsymbol{x}_i')$$

1.3 Estimation network

Given the low-dimensional representations for the input samples, the estimation network performs density estimation under the framework of Gaussian mixture model (GMM).

To beter understand the mechanism of the estimated network, we first recall some basic knowledge on GMM. The GMM can be viewed as the marginal distribution of a joint distribution

$$\begin{aligned} p(\boldsymbol{x}) &= \sum_{k=1}^{K} p(\boldsymbol{x}, y) \\ &\quad y \text{ is latent variable} \\ &= \sum_{k=1}^{K} p(y=k)p(\boldsymbol{x}|y=k) \\ &\quad \text{Assume each condtional density function is Gaussion } p(\boldsymbol{x}|y=k) = \mathcal{N}(\boldsymbol{x};\mu_k,\Sigma_K) \\ &\quad \text{Let } \phi_k = p(y=k) \text{ denotes the mixture component distribution} \\ &= \sum_{k=1}^{K} \phi_k \mathcal{N}(\boldsymbol{x};\mu_k,\Sigma_k) \end{aligned}$$

where the integer K is the number of mixture components and $\mathcal{N}(\mu_k,\Sigma_k) = \frac{-(\boldsymbol{z}-\mu_k)^T\Sigma_k^{-1}(\boldsymbol{z}-\mu_k)/2}{\sqrt{\det(2\pi\Sigma_k)}}$.
The classical approch perfoms dimensionality reduction on the training dataset $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$ to obtain low dimensional dataset $\{\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_N\}$ and then trains GMM by maximizing the log-likelihood of the low-dimensional dataset $\{\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_N\}$

$$\max_{\{\phi_k,\mu_k,\Sigma_k\}_{k=1}^K} \frac{1}{N}\sum_{i=1}^{N} \log\left[\sum_{k=1}^{K} \phi_k \mathcal{N}(\boldsymbol{z}_i;\mu_k,\Sigma_k)\right]$$

Particularly, the model parameter $\Gamma = \{\phi_k, \mu_k, \Sigma_k\}_{k=1}^K$ are determined via the famous Expectation-Maximization (EM) algorithm.[2]

The estimation network utlizes a multi-layer neural network to predict the mixture membership $p(y=k|x;\Gamma)$ as follow

$$\boldsymbol{p} = \text{MLN}(\boldsymbol{z};\theta_m) \text{ and } \hat{\gamma} = \text{softmax}(\boldsymbol{p})$$

where $\boldsymbol{p}$ is the output of a multi-layer network parameterized by θ_m and $\hat{\gamma}$ is given by

$$\hat{\gamma} = \begin{bmatrix} \frac{e^{p_1}}{\sum_{k=1}^K e^{p_k}} \\ \vdots \\ \frac{e^{p_K}}{\sum_{k=1}^K e^{p_k}} \end{bmatrix}$$

1. In the paper later, the construction error is denoted by $L(\boldsymbol{x}_i, \boldsymbol{x}_i')$ instead of $f(\boldsymbol{x}_i, \boldsymbol{x}_i')$.
2. EM algorithm is a gradient-free iterative algorithm. But generally it cannot converge to the global optimal.

where $\hat{\gamma}_k$ approximates the mixture memership $p(y=k|\boldsymbol{z};\Gamma)$. Given a dataset of N samples and their membership prediction, the GMM's parameter $\Gamma=\{\phi_k,\,\mu_k,\,\Sigma_k\}_{k=1}^K$ can be estimated as follows (all equations are derived by Bayes' theorem)

$$\begin{aligned}\hat{\phi_k} &= \hat{p}(y=k)\\ &= \int \hat{p}(y=k,\boldsymbol{z})d\boldsymbol{z}=\int \hat{p}(y=k|\boldsymbol{z})\hat{p}(\boldsymbol{z})d\boldsymbol{z}\\ &= \sum_{i=1}^{N}\hat{p}(y=k|\boldsymbol{z}_i)\hat{p}(\boldsymbol{z}_i),\hat{p}(\boldsymbol{z}_i)=\frac{1}{N}\\ &= \frac{1}{N}\sum_{i=1}^{N}\hat{p}(y=k|\boldsymbol{z}_i)=\frac{1}{N}\sum_{i=1}^{N}\hat{\gamma}_{ik}\end{aligned}$$

$$\begin{aligned}\hat{\mu}_k &= \int \boldsymbol{z}\hat{p}(\boldsymbol{z}|y=k)d\boldsymbol{z}\\ &= \sum_{i=1}^{N}\boldsymbol{z}_i\hat{p}(\boldsymbol{z}_i|y=k)=\sum_{i=1}^{N}\boldsymbol{z}_i\frac{\hat{p}(\boldsymbol{z}_i,y=k)}{\hat{p}(y=k)}\\ &= \sum_{i=1}^{N}\boldsymbol{z}_i\frac{\hat{p}(y=k|\boldsymbol{z}_i)\hat{p}(\boldsymbol{z}_i)}{\hat{p}(y=k)}\\ &\quad \hat{p}(y=k)=\hat{\phi_k}=\frac{1}{N}\sum_{i=1}^{N}\hat{\gamma}_{ik},\hat{p}(\boldsymbol{z}_i)=\frac{1}{N},\hat{p}(y=k|\boldsymbol{z}_i)=\hat{\gamma}_{ik}\\ &= \sum_{i=1}^{N}\frac{\hat{\gamma}_{ik}}{\sum_{i=1}^{N}\hat{\gamma}_{ik}}\boldsymbol{z}_i\end{aligned}$$

and

$$\begin{aligned}\hat{\Sigma}_k &= \int \hat{p}(\boldsymbol{z}|y=k)(\boldsymbol{z}-\hat{\mu}_k)(\boldsymbol{z}-\hat{\mu}_k)^T d\boldsymbol{z}\\ &= \sum_{i=1}^{N}\hat{p}(\boldsymbol{z}_i|y=k)(\boldsymbol{z}_i-\hat{\mu}_k)(\boldsymbol{z}_i-\hat{\mu}_k)^T\\ &= \sum_{i=1}^{N}\frac{\hat{\gamma}_{ik}}{\sum_{i=1}^{N}\hat{\gamma}_{ik}}(\boldsymbol{z}_i-\hat{\mu}_k)(\boldsymbol{z}_i-\hat{\mu}_k)^T\end{aligned}$$

where $\hat{\gamma}_i=[\ \hat{\gamma}_{i1}\ \ \cdots\ \ \hat{\gamma}_{iK}\ ]^T$ is the membership prediction for $\boldsymbol{z}_i$, which represents

$$[\ p(y=1|\boldsymbol{z}_i)\ \ \cdots\ \ p(y=K|\boldsymbol{z}_i)\ ]$$

With the estimated parameters, GMM can be optimized by maximizing the log-likelihood of the low-dimensional dataset $\{\boldsymbol{z}_1,\boldsymbol{z}_2,\cdots,\boldsymbol{z}_N\}$

$$\max_{\Gamma,\theta_m}\frac{1}{N}\sum_{i=1}^{N}\log\sum_{k=1}^{K}\phi_k\mathcal{N}(\boldsymbol{z}_i;\mu_k,\Sigma_k)$$

by any gradient-based approach (Note $\Gamma=\{\phi_k,\mu_k,\Sigma_k\}_{k=1}^K$ depends on θ_m). Or,

$$\min_{\Gamma,\theta_m}\frac{1}{N}\sum_{i=1}^{N}\log E(\boldsymbol{z}_i)$$

where $E(\boldsymbol{z}_i)$ is the so-called sample energy

$$E(\boldsymbol{z}_i) = -\log\left(\sum_{k=1}^{K} \phi_k \mathcal{N}(\boldsymbol{z}_i; \mu_k, \Sigma_k)\right) = -\log\left(\sum_{k=1}^{K} \frac{-(\boldsymbol{z}_i - \mu_k)^T \Sigma_k^{-1} (\boldsymbol{z}_i - \mu_k)/2}{\sqrt{\det(2\pi\Sigma_k)}}\right)$$

1.4 Objective function

Given dataset $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$, the DAGMM training is to minimze the following objective function

$$J(\theta_e, \theta_d, \theta_m) = \frac{1}{N}\sum_{i=1}^{N} L(x, x') + \frac{\lambda_1}{N}\sum_{i=1}^{N} E(z_i) + \lambda_2 P(\hat{\Sigma})$$

The objective function includes three components.

- $L(\boldsymbol{x}_i \boldsymbol{x}_i')$ is the reconstruction error cased the compression network. In practice, L_2-norm $L = \|\boldsymbol{x}_i - \boldsymbol{x}_i'\|_2^2$ gives desirable results.
- The estimation networks that minimizes the sample energy is equivalent to maximizing the likelihood of the observed samples. This does make sense since we assume in the training dataset normal samples are the majority.
- DAGMM also has the singularity problem as in GMM. To avoid this issue, small values on the diagonal entries by $P(\hat{\Sigma}) = \sum_{k=1}^{K} (\hat{\Sigma}_k) = \sum_{k=1}^{K} \sum_{j=1}^{d} \frac{1}{\hat{\Sigma}_{kjj}}$, where d is the dimension of the low-dimensional representations, are penelized. By minimizing $P(\hat{\Sigma})$, $\Sigma_1, \Sigma_2, \Sigma_K$ tend to be diagonally dominant matrices and thus be invertible.
- Hyperparameter setting: $\lambda_1 = 0.1$ and $\lambda_2 = 0.005$.

2 Application for rehabilitation exercise assessment

2.1 Dataset UI-PRMD

The UI-PRMD dataset [3] consists of skeletal data collected from 10 healthy subjects. Each subject completed 10 repetitions of 10 rehabilitation exercises. The data were acquired with a Vicon optical tracking system, and consist of 117-dimensional sequences of angular joint displacements. The subjects performed the exercises both in a correct manner, hereafter referred to as correct movements, and in an incorrect manner, i.e., simulating performance by patients with musculoskeletal constraints, hereafter referred to as incorrect movements.

Order	Exercise
1	Deep squat
2	Hurdle step
3	Inline lunge
4	Side lunge
5	Sit to stand
6	Standing active straight leg raise
7	Standing shoulder abduction
8	Standing shoulder extension
9	Standing shoulder internal–external rotation
10	Standing shoulder scaption

Table 1. Exercises in the UI-PRMD Dataset

2.2 Method

Initially, we only apply DAGMM on the first exercise to see how much progress could be made compaired to the results in [1]. Note when training DAGMM only the correct movements should be used, as the experimental goal is not for anomaly detection. For the rehabilitation exercise assessment, the correct movements work as the prototype and new movement sequences are evaluated by the average deviation (measured by the likelihood of movement squence) from the correct movements.

2.2.1 Notation

The acquired data by the sensory system for one particular exercise performed by S healthy subjects is denoted by $\mathbb{X}$, and hereafter they are referred to as reference movements (correct movements). For the Dataset UI-PRMD, $S = 10$.

- The symbol R_s is used for the number or repetitions of the exercise by the s-th subject.
- The combined data for all R_s repetitions of the exercise by the s-th subject is denoted X_s. Thus, $\mathbb{X} = \{X_1, \cdots, X_S\}$.
- R is used for the total number of all repetitions by the S subjects, i.e., $R = \sum_{s=1}^{S} R_s$. For example, $R = 90$ for the first exercises.
- Using the notation X_{sr} for the collected data of the r-th repetition by the s-th subject. Thus, $X_s = \{X_{s1}, \cdots X_{sR_s}\}$.
- The data for each repetition X_{sr} is a temporal sequence of T measurements, therefore

$$X_{sr} = \begin{pmatrix} \boldsymbol{x}_{sr}^{(1)} & \cdots & \boldsymbol{x}_{sr}^{(T)} \end{pmatrix}$$

  In the first exercise, $T = 240$.
- The individual measurement $\boldsymbol{x}_{sr}^{(t)}$ for $t \in \{1, 2, \cdots, T\}$ is a D-dimensional vector, consisting of the values for all joint displacements in the human body, i.e

$$\boldsymbol{x}_{sr}^{(t)} = \begin{pmatrix} x_{sr}^{(t,1)} & \cdots & x_{sr}^{(t,D)} \end{pmatrix}$$

The collected data for the patient group are referred in the article as patient movements (incorrect movements), and are denoted with the symbol $\mathbb{Y}$ and analogous notations are used.

2.2.2 Metric for assessing rehabilitation exercise quality

Given a exercise sequence $\boldsymbol{m}$ (we don't know it's correct or incorrect), its derivation from the reference movements $\mathbb{X}$ is naturally used to quantify the motion quality. There are lots of metrics to meausre the deviation, such as Euclidean distance, Mahalanobis distance, and dynamic time warping (DTW) [2], and so on. For example, the movement quanlity of sequence $\boldsymbol{m}$ (we don't know it's correct or incorrect) based on Euclidean distance is defined as

$$\ell(\boldsymbol{m}, \mathbb{X}) = \frac{1}{R} \sum_{s=1}^{S} \sum_{r=1}^{R_s} \sum_{t=1}^{T} \left\| \boldsymbol{m}^{(t)} - \boldsymbol{x}_{sr}^{(t)} \right\|_2 \tag{1}$$

Following [1], we choose the likelihood induced by a GMM as the metric to evaluate exercise quality. But instead of using two-stage training of a GMM, we train the DAGMM and obtain a by-product GMM, which has the density function

$$p(\cdot; \Gamma) = \sum_{k=1}^{K} \phi_k \mathcal{N}(\cdot; \mu_k, \Sigma_k)$$

where $\Gamma = \{\phi_k, \mu_k, \Sigma_k\}_{k=1}^{K}$ is the parameter set of GMM. Thus, the negative log-likelihood for any movement $Y = \begin{pmatrix} \boldsymbol{y}^{(1)} & \cdots & \boldsymbol{y}^{(T)} \end{pmatrix}$ is calculated as

$$L(Y;\Gamma) = -\sum_{t=1}^{T} \log\left[\sum_{k=1}^{K} \phi_k \mathcal{N}(\boldsymbol{y}^{(t)}; \mu_k, \Sigma_k)\right] \tag{2}$$

and its movement quality is evaluated by its derivation from the prototype dataset $\mathbb{X}$ based on the negative likelihood, i.e.,

$$\Delta(Y, \mathbb{X}) = \frac{1}{R}\sum_{s=1}^{S}\sum_{r=1}^{R_s} \left|L(Y;\Gamma) - L(\boldsymbol{x}_{sr}^{(t)};\Gamma)\right| \tag{3}$$

If $\Delta(Y, \mathbb{X})$ is very small, we see Y is similar to $\mathbb{X}$ and thus the movement quality of Y is good. For the mixture component number K, we follows [1] to set $K = 6$.

2.2.3 Metric for assessing model performance

Let denote the performance metrics for correct movements $\mathbb{X}$ and incorrect movements $\mathbb{Y}$ by $\mathbf{x} = (x_1, x_2, \cdots, x_R)$ and $\mathbf{y} = (y_1, y_2, \cdots, y_R)$. For the first exercise, $R = 90$. Note in [1], the performance metric is the negative likelihood of the movement, i.e.,

$$x_i = L(X_i;\Gamma) \text{ and } y_j = L(X_j;\Gamma)$$

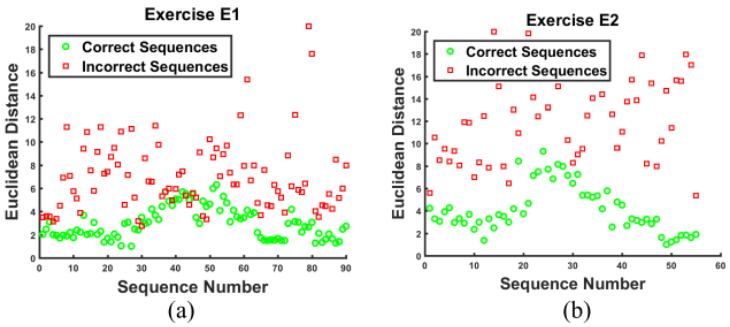
and then are linearly scaled to $[1, 20]$ by

$$x_i' = \frac{19(x_i - m)}{M - m} + 1,\ y_i' = \frac{19(y_i - m)}{M - m} + 1 \text{ for } i \in \{1, 2, \cdots, R\} \tag{4}$$

where $M = \max_{i,j\in\{1,2,\cdots R\}}\{x_i, y_i\}$ and $m = \min_{i,j\in\{1,2,\cdots R\}}\{x_i, y_i\}$. Then the so called separation degree function is proposed to compare different performance metrics. Particularly, the separation degree between two positive sequences $\mathbf{x} = (x_1, x_2, \cdots, x_m)$ and $\mathbf{y} = (y_1, y_2, \cdots, y_n)$ is defined as

$$S_D(\mathbf{x}, \mathbf{y}) = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\frac{x - y}{x + y} \tag{5}$$

Values of the seperation degree close to 1 or -1 indicate good separation between two sequences. For instance, one can observe a clearer differentiation between correct and incorrect movements in Exercise 2 in comparision to Exercise 1 from the following figure.



(a) (b)

As reported in [1], the separation degree for exercise 2, 0.479 is larger than that for exercise 1, 0.384. Also, the negative likelihood is identified as the best metric by comparing the separation degree.

In this project, we adopt the negative likelihood deviation as the the performance metric (3) simlar to what is done in Euclidean distance (1).

2.3 Some thinking and plan

1. It seems that it's more natural to use (3) as the performance metric. So we will compare the difference between performance metric (2) and (3).

2. The scale method (4) is not good. It is very sensitive to outliers since maximum and minimum are used. If possible, avoid this step.

3. Propose a better separation degree function since (5) is only valid for positive sequences. The separation degree function should be a homogeneous function.

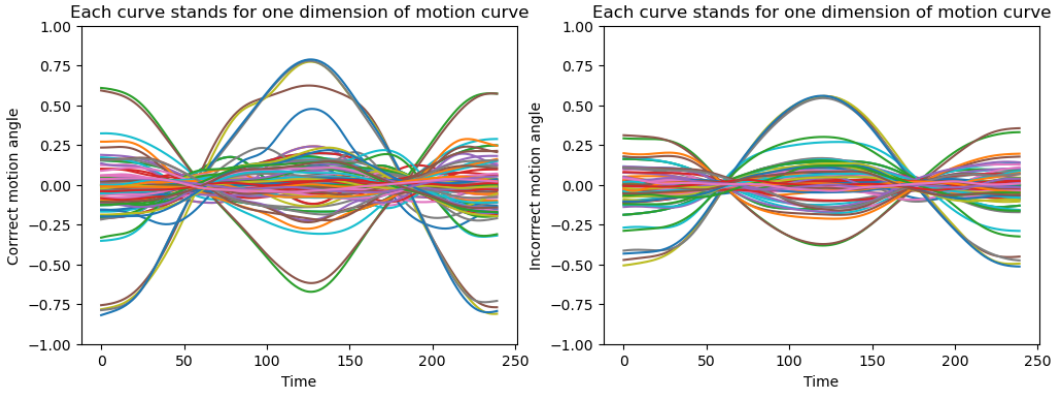
4. Obtain a GMM via training a DAGMM. Then derive the performance metric according to (3) and compare it with that based on GMM trained via EM algorithm to see how much improvement can be made.

3 Experimental results

We will conduct three method: (1) PCA plus GMM; (2) Autoencoder plus GMM; (3) DAGMM

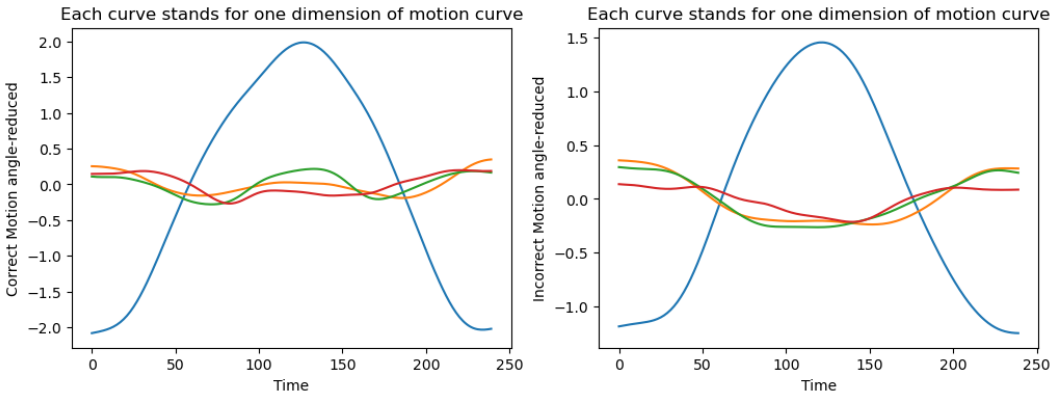
(1) PCA plus GMM

The first repetition of the first exercise

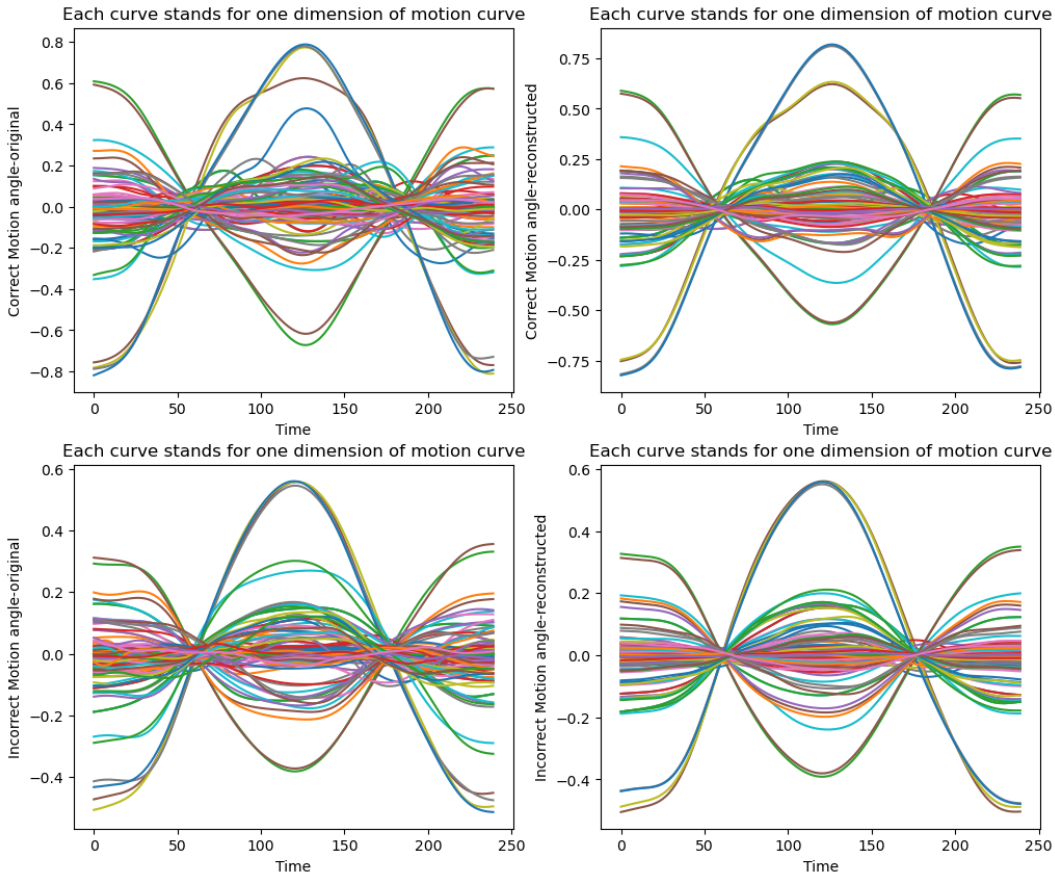


Each figure has 117 curves as the data is 117 dimensional, and the sequence length is 240 since the exercise has 240 measurements (240s). It's clearly that the data vary greatly over time in some specific dimensions. This motivates us to try PCA compressing data into a lower dimension space.

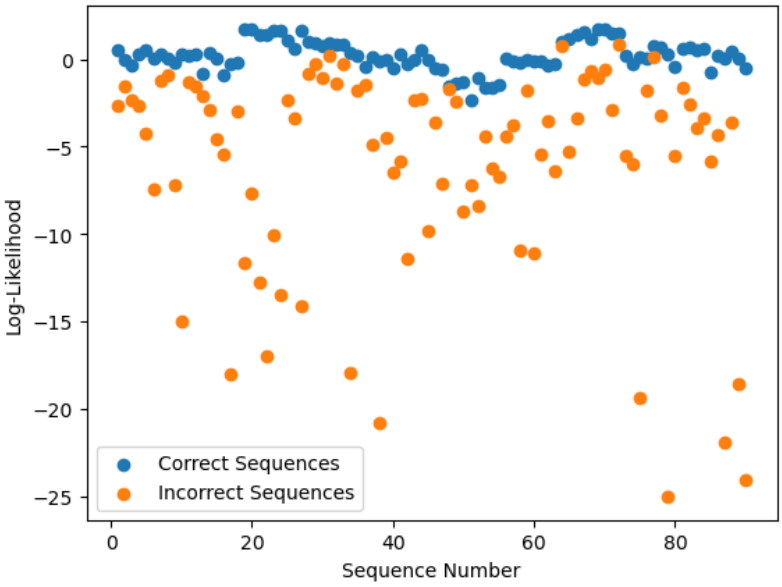
The first repetition (reduced into 4 dimensional space)



It looks not bad as the main data pattern preserves. The following figure displays the original data and reconstructed data, which further comfirms that PCA makes sense to somehow.

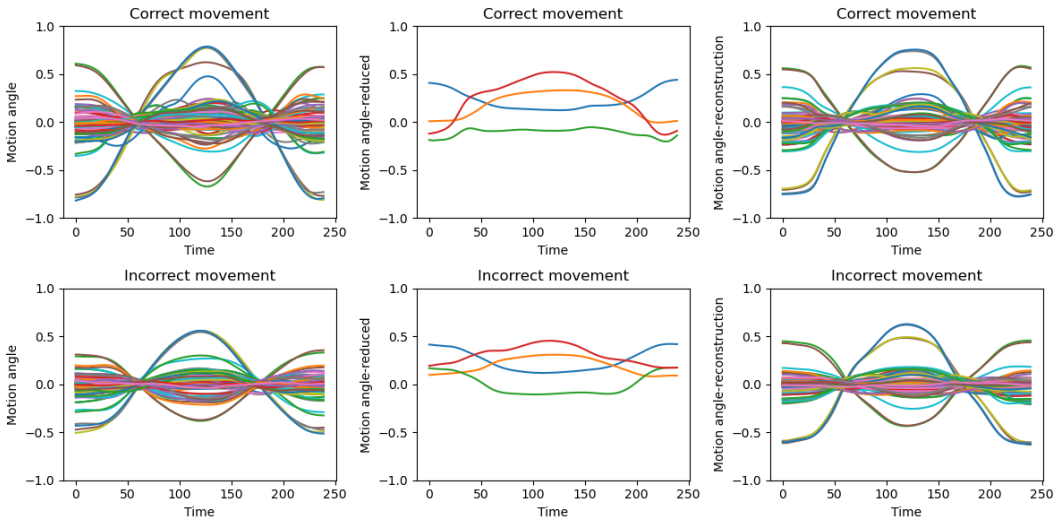


Then we train a GMM using the correct dataset and compute the average likelihood over time for each repetition (the same exercise are performed 90 times by 10 persons correctly, and performed 90 times by 10 persons incorrectly).

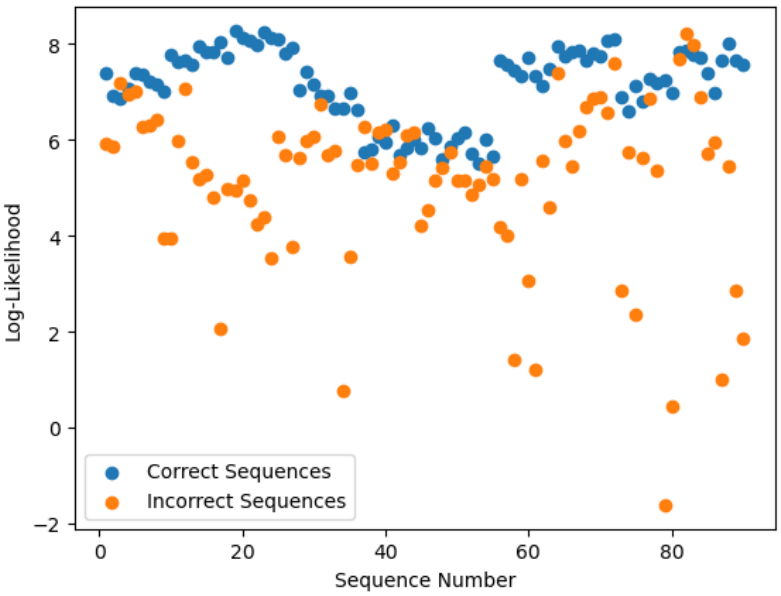


We see that some correct and incorrect motion repetitions mix together.

We want to see if autoencoder plus GMM could provide a better metric to distinguish the correct repetitions from incorrect ones. Likewise, we compress data to 4 dimensions via a deep autoencoder and train a GMM to model the correct motion. The autoencoder consists of 8 LSTM layers, where the first 4 layers encode data and the last 4 layers decode data. After 1000 epochs, we stop training the model as the reconstruction error is very small (train error: `0.00165`; valid error: `0.00157`)[3].



As shown in the figure above, the reconstruction preserves the main patterns in the original correct motion. Thus, we obtain an appropriate model and thus, reasonable low-dimensional representations. As what we have done previsouly, we train a GMM to fit the reduced correct dataset and compute the average log-likelihood over time. We show the result in the following figure.

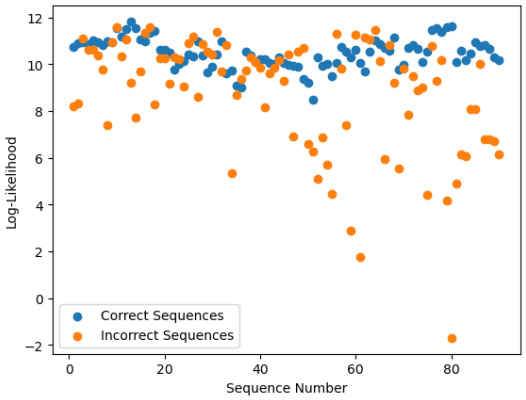


Correct and incorrect movement sequences still mix together.

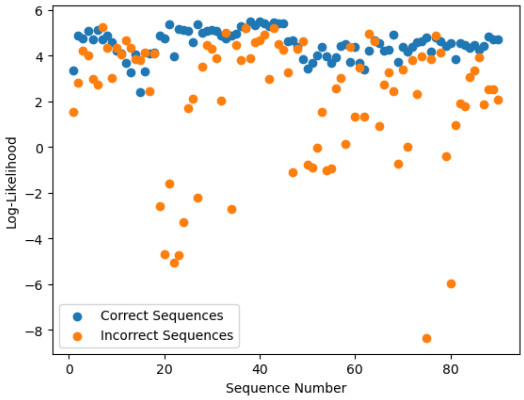
Further, how about DAGMM? Next we will implement and DAGMM. As you see, using different methods to encode data, the log-likelihood of motion data are in different scales. We need to come up with a more reasonable quantity to measure the separation degree and use it to judge which method is the best.

3. We could train the model with more epochs but it's time consuming. We could do it latter. Also, we split the correct dataset into training and validation dataset and track the training and valid loss. In this way, we could see if the model overfits training data.

In DAGMM, we compress the data to 2 dimensions, i.e, $z_c \in \mathbb{R}^2$, compute the MSE reconstruction loss and cosine loss, and concatenat them together to obtain the low-dimensional representation $z \in \mathbb{R}^4$. We follow the DAGMM paper, setting $\lambda_1 = 0.1, \lambda_2 = 0.005$, and learning rate $\text{lr} = 10^{-4}$. We explore two architectures of the compression network: (a) fully connected layers, followed by the hyper-tagent function, both encoding and decoding nets consist of three layers; (b) the same network structure by each layer is replaced by the LSTM layer.



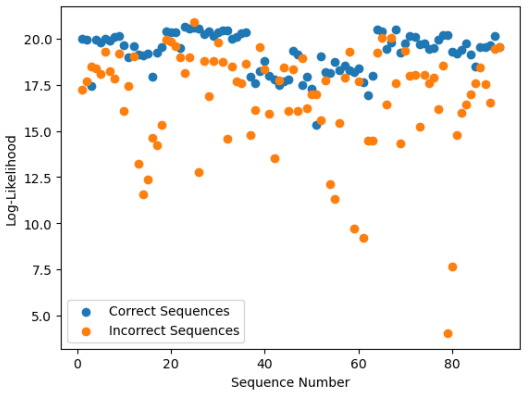
(a) fully connected layer



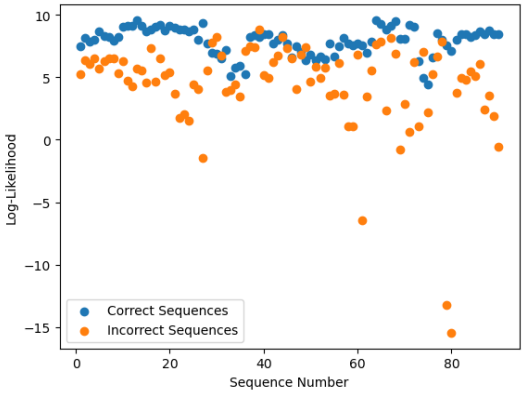
(b) LSTM

Both models are trained for 200 epochs. We could say (b) LSTM is better than (a) but obviously neither (a) or (b) is better than PCA plus GMM.

To dig out possible reasons for DAGMM's underperformance, we print all losses while training the DAGMM. We find that the energy term and the loss of penalizing small diagonal entries of covariance matrices are much larger than the reconstruction loss and think this is one reason leading to bad model performance. Thus, we reset $\lambda_1 = 0.01$ an and $\lambda_2 = 10^{-10}$. Also, we change the latent dimension from 2 to 4, which is the latent dimension in PCA. We re-run the code and show the results below.



(c) fully connected layer



(d) LSTM

Two models have better performance. Obviously, LSTM outperforms (c) and autoencoder plus GMM, and achieves slightly worse (comparable) performance than PCA plus GMM. Also, we notice that The model variance of DAGMM with LSTM layer is smaller than that of DAGMM with fully connected layer.

4 Summary

In this project, we explore the application of DAGMM to evaluate rehabilitation exercises and compare its performance with classical methods: (1) PCA plus GMM and (2) autoencoder plus GMM. The results of DAGMM are comparable to (1), but better than (2). We could design an

estimation network with LSTM layers and verify the performance of the new DAGMM. However, we believe that the main reason hindering the performance of the model is the penalty method. More importantly a new method should be proposed to penalize the singularity of the covariance matrices. Furthermore, the idea of incorporating the mechanics of Dirichlet processes into DAGMMs is natural and interesting.

Bibliography

[1] Yalin Liao, Aleksandar Vakanski, and Min Xian. A deep learning framework for assessing physical rehabilitation exercises. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 28(2):468–477, 2020.

[2] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

[3] Aleksandar Vakanski, Hyung-pil Jun, David Paul, and Russell Baker. A data set of human body movements for physical rehabilitation exercises. *Data*, 3(1):2, 2018.

[4] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International conference on learning representations*. 2018.