Imitation Learning for Skill Transfer in Human-Robot Teleoperation

Marco Schouten
Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Delft, Netherlands
m.schouten-4@student.tudelft.nl
Student Number: 5352908

Chenghao Xu
Mechanical, Maritime and Materials Engineering
Delft University of Technology
Delft, Netherlands
c.xu-7@student.tudelft.nl
Student Number: 5266068

***Abstract*—Robotic intelligence aims at performing human-like behaviour under various scenarios. To design an adaptive controller for a robot manipulator to carry out multiple complex skills is extremely expensive and time-consuming. In this research we propose a methodology for imitation learning to reproduce the actions on robot manipulators with minimal inputs, using Learning by Demonstration (LbD) techniques. Action demonstrations are acquired using a Kinect motion sensor to record the movement of the human. The experimental results confirm the effectiveness and efficiency of the proposed method when compared with the original input trajectory.**

The repository for the source code is available at https://github.com/MarcoSchouten/Imitation-Learning-for-Skill-Transfer-in-Human-Robot-Teleoperation.git

***Index Terms*—Imitation Learning, Gaussian Mixture Model (GMM), Robot Manipulation, Teleoperation, Learning by Demonstration**

I. Introduction

A. Background

Robotic intelligence is currently constructed to achieve imitation of human behaviors such as decision-making and stress response, which aims at performing human-like actions under various scenarios. This kind of strong artificial intelligence is used for performing tasks without any kind of supervision. Currently, one popular application lies in self-driving vehicles, which imitates the human driver to deal with driving process.

On the other hand, with the rapid development of 5G technology, teleoperation could perform remote control to operate the robot at a remote distance. Therefore, how to control the remote robot to imitate human behavior in real time also becomes a hot research issue.

B. Motivation

One of the main obstacles in teleoperation imitation control is that transmission data size influences imitation performance. In this work, we developed a system to learn the task-specific trajectories from a human arm. As result, we are able to generate similar trajectories by a robot using minimal representation inputs.

A common approach for imitation learning is based on Reinforcement Learning. The optimal policy is usually learned by calculating cumulative rewards. This method is simple and straightforward, and it performs better when more training data is available. However, in a teleoperation setting with sequential decisions, the learner cannot get rewards frequently, and the search space is too large. A probabilistic approach offers a solid solution to deal with multi-step decision-making problems.

C. Contribution

Our contribution is mainly divided into three folds:

- Extract the skeleton features to generate demonstrations of human actions using a Kinect sensor. The trajectory of the hand will be stored in the sequence of (t, x, y, z) from demonstrations.
- Generate the GMM method to generate a set of parameters that characterize the original trajectory and regenerate the new trajectory from characterized parameters
- Construct the simulated robot arm model to perform the new trajectories and evaluate the performance on generated trajectories

D. Ethical and Societal Implications

On the ethical account, the trend of intelligent robots requires us to rethink the definition of robotics. Due to efficient performances, the impact that robotics application has on society is of utmost relevance. Deontology is one of the three frameworks of Normative Ethics (consequentialism, Deontology, Virtue Ethics), which morally critiques actions per se is wrong or right according to rights and duties According to the normative framework of deontology, robotic's intelligence misbehaviour's (i.e. not acting in the interests of society as a whole and to sustain human life) is unethical and must be avoided.

On the other hand, if the applications are built with aims of benevolence, (e.g. constructions or aided surgery), the precision and accuracy of robotics can lead to huge benefits.

II. Methods

Learning accurate trajectories from a few demonstrations is a problem of high relevance in the field of robotics. Additionally, for a teleoperation setting, it is required to transmit low and robust data on the communication channel, which leads

to less latency (in ms) between the two endpoints. Lastly, we tested this system using retrieved trajectory from Kinect. To better deal with this problem, we decomposed it into three steps and designed a pipeline that deals with each sub-problem independently. In other words, the problem can be decomposed into three steps:

- Extract trajectory from Kinect data
- Generate the most likely trajectory according to a probabilistic model
- Compress the generated trajectory, send it over the communication channel, reconstruct the originally generated trajectory

To solve this problem we structured a pipeline Fig 1 that deals with each sub-problem independently.

A. Trajectory extraction from kinect skeleton

The objective of this part is to acquire trajectories visually using a Kinect sensor, and estimate the sequence of coordinates (x, y, z) from the action demonstration. We use some open source libraries such as OpenCV and OpenNI , which are constructed on ROS. The acquisition of demonstrations stage can be divided into 3 main steps:

- Action execution: The person will execute the target actions. For example, in this work, we acted swinging the arm 5 times. The Kinect sensor will record the action under a fixed frequency.
- Acquisition of demonstration: An artificial vision system composed of a Kinect motion sensor is implemented to identify the position of the movement of the hand, as shown in Figure 2. During each demonstration, the execution calculates the centroid of the hand to reduce the induced noise. The output from this stage is a set of coordinates (x, y, z) of the hand centre.
- Save demonstration: After obtaining the data (x, y, z) of each demonstration, the position is stored on a text file, which can be processed and visualized on MATLAB.

B. Learning from Demonstrations

Robot Learning from Demonstration (LfD) or Robot Programming by Demonstration (PbD) (also known as Imitation Learning and Apprenticeship Learning) is a paradigm for enabling robots to autonomously perform new tasks [1]. The core idea of imitation learning is that a robot can perform a certain task, just by replicating the movement given as input of a human, without being explicitly programmed. The task can be arbitrally complex and usually is decomposed into multiple subtasks. For example, opening a door involves lifting the arm, grasping the handle and moving the handle downwards. In contrast to hardcoded programming, where each subtask is carefully coded and tested, here the robot only needs to be shown how to do the task. An important note to be made is that LfD is not about replicating the movement, but about generalizing the movement from several demonstrations.

Specifically for the teleoperation settings, the goal is to generate the motion at the same time that the user is acting. This is possible because by making the robot learn from a set of demonstrations it can build a probabilistic model for the task, and then enhance its precision by considering corrections given by the user's input.

In other words, the robot's behaviour is determined before the human's input has arrived at the receiving endpoint (thus achieving zero latency).

This research bases the LfD approach through encoding segments of motion as a Gaussian Mixture Model (GMM), and subsequently using a Gaussian Mixture Regression (GMR) to retrieve the desired motion [2]. Calinon approached skill encoding as a clustering problem: Instead of modelling the regression function directly, they estimate a joint distribution over the state variables and perform regression through conditioning. During reproduction, the motion is retrieved through statistical inference using Gaussian Mixture Regression (GMR).

To summarize, first, we fit a Gaussian Mixture Model using the Expectation Maximization algorithm, that best represents the input trajectories. Secondly, we perform a Conditioning operation, to retrieve a tube of Gaussians whose mean represents a point of the inferred trajectory.

To understand the inner workings and details of the GMM it is necessary to explain how the training of the machine learning model happens. GMM training is based on the Expectation-Maximization algorithm, or EM algorithm, proposed by [3].

EM is a general technique for finding maximum likelihood solutions for probabilistic models when the observations can be viewed as incomplete data, which means they are generated through latent variables. It is an iterative process between the (E) step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters, and a maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the E step. These parameter estimates are then used to determine the distribution of the latent variables in the next E step. [4].

GMM is initialized through k-means [1] [7] to find a better convergence.

Lastly, we build a dictionary of a few tasks to test our model on different types of trajectories.

C. Encoder-Decoder Architecture for trajectory compression

In this section, given the inferred trajectory as input, we encode it with a low number of data points, that we consider the most significant. We consider use a Laplacian filer, to determine the variation of the line, and compare it against a threshold.

$$\nabla^2 = \vec{\nabla} \cdot \vec{\nabla}$$

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

In this way we can reduce drastically the number of points, hence reducing the load on the communication channel.

[1]K-means is an algorithm to partition the observations into k different clusters. It assigns each observation to the cluster with the nearest mean. As a result, it partitions the space into Voronoi Cells [6].

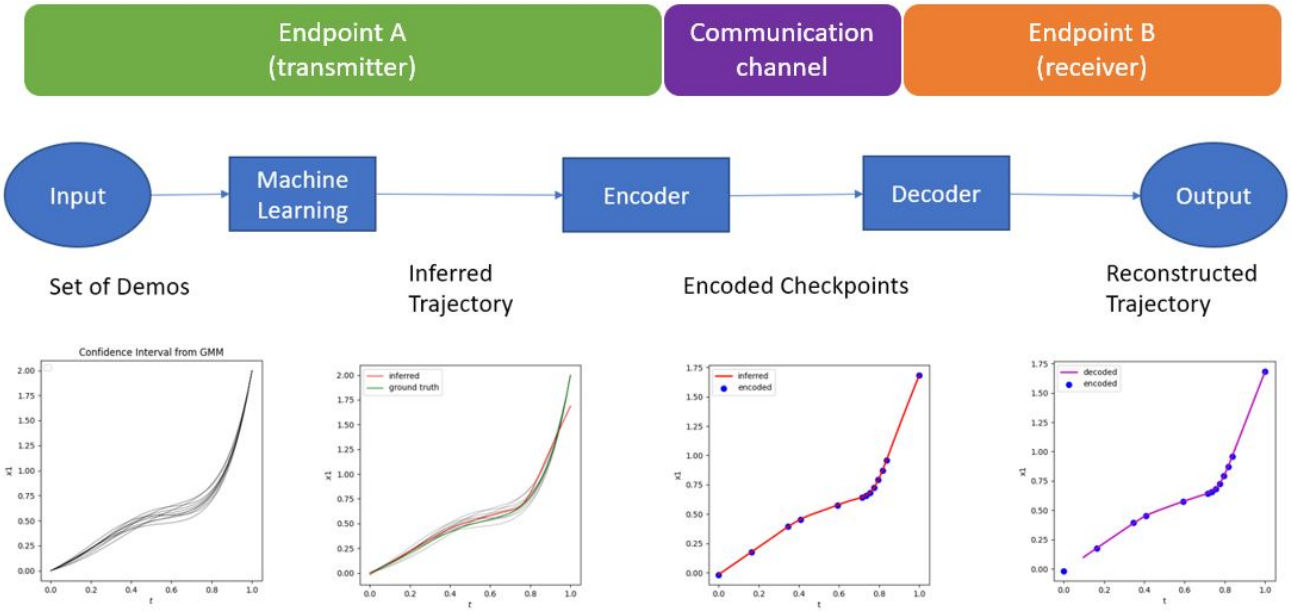

Fig. 1. Pipeline, showing up only 1 dimension (x1) for a clearer illustration.
(1) From a set of demonstrations, the GMM is trained using with EM algorithm using the demonstrations as input. (2) Next the GMR conditioning operator used on the GMM generates the most likely trajectory. (3) Thirdly the encoder, compresses the inferred trajectory by selecting critical checkpoints. (4) Finally, the given checkpoints are passed on the communication channel and an interpolated line is generated using B-spline algorithm, thus retrieving the output trajectory reconstructed on the receiving endpoint

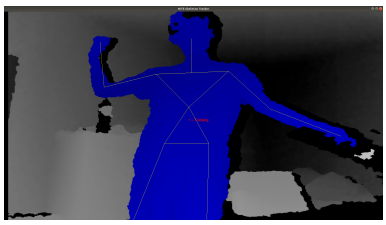

Fig. 2. Skeleton Extraction for action demonstration

As for the decoder we used a Bspline system, that gets as input the filtered checkpoints (encoded trajectory) and retrieves by interpolation the (decoded trajectory) which is ultimately the final output for our system.

III. Results

We integrated the library from [8] to compute the EM algorithm described in Fig 3

To evaluate whether our implementation was correct, we tested our system on generated functions.

Specifically, we used non-linear vectorial functions defined as:

$$\mathbf{f} : \mathbb{R} \longrightarrow \mathbb{R}^3$$

$$\mathbf{f}(t) = (x1, x2, x3)$$

We treated each variable independently to determine the centres of the Gaussians, i.e. we decomposed the 3d signal into three 1d signals, each representing its corresponding dimension.

Algorithm 2.1 Expectation Maximization for GMM

function EM_GMM($x_{1:N}$, $\{\mu_k, \Sigma_k, \pi_k\}_{k=1}^K$)
  while $\delta ll > \epsilon_{conv}$ **do** ▷ Check change of likelihood, δll, between iterations
    # E-Step:
    for $k \in \{1, \cdots, K\}$ **do**
      for $n \in \{1, \cdots, N\}$ **do**
        $\gamma_{n,k} = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{i=1}^K \pi_i \mathcal{N}(x_n|\mu_i, \Sigma_i)}$
      $N_k = \sum_n^N \gamma_{n,k}$

    # M-Step:
    for $k \in \{1, \cdots, K\}$ **do**
      $\mu_k \leftarrow \arg\max_{\mu_k} (\mathrm{L}(x_{1:N}, \mu_k, \gamma_{1:N,k}))$
      $\Sigma_k \leftarrow \frac{1}{N_k} \sum_n^N \gamma_{n,k} \mathrm{Log}_{\mu}(x_n) \mathrm{Log}_{\mu}(x_n)^\top$
      $\pi_k \leftarrow \frac{N_k}{N}$

    # Compute log-likelihood:
    $ll = \sum_{n=1}^N \ln(\sum_{k=1}^K \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k))$
  return $\{\mu_k, \Sigma_k, \pi_k\}_{k=1}^K$

Fig. 3. E-M Algorithm summary from [5]

A very recent study [9] shows that R2 [2] can be considered the best metric to determine the goodness of a regression model. Additionally, we evaluated for Mean Square Error and Mean Absolute Error. We compared the ground truth trajectory between the inferred and the reconstructed trajectory after the encoder-decoder system.

[2]R2 is a statistic that will give some information about the goodness of fit of a model. R2 is a statistical measure of how well the regression predictions approximate the real data points in regression. An R2 of 1 indicates that the regression predictions perfectly fit the data.

	x1	x2
mae	3.69664244e-06	3.86535089e-02
mse	2.37086698e-11	3.72760437e-03
r2	1.	0.98059981]

TABLE I
MACHINE LEARNING BLOCK.
GROUND TRUTH TRAJECTORY—— INFERRED TRAJECTORY

	x1	x2
mae	0.15835882	0.18605222
mse	0.03214314	0.04579591
r2	0.62513977	0.73572401

TABLE II
ENCODER BLOCK
INFERRED TRAJECTORY —— RECONSTRUCTED TRAJECTORY

	x1	x2
mae	0.15835999	0.2043527
mse	0.03214367	0.05349581
r2	0.62514137	0.72158294]

TABLE III
OVERALL SYSTEM PERFORMANCE
GROUND TRUTH TRAJECTORY—— RECONSTRUCTED TRAJECTORY

To further analyze the model's output we plotted the mean, variance, and confidence intervals for the tasks In this case, we

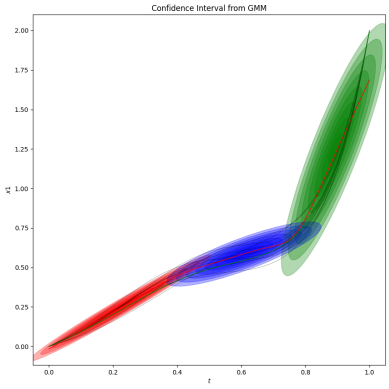

Fig. 4. Task A [10]. The three different components of the Gaussian Mixture Model are highlighted in red, blue, green

fitted the task trajectory using 3 components [3] for the GMM.

For the task trajectories that we found, we implemented the Grid Search approach to find the best hyper-parameter for our inference model (i.e. the number of Gaussian components)

IV. DISCUSSION

The EM algorithm converged for all the tasks trajectory over which we tested. We tested it only on 10 demonstrations for each task. However, the generalization capabilities for the system were quite impressive. The R2 score for the Machine Learning Block was close to 1, proving that the regression model is effective.

[3]GMM fork components mean that you have a categorical latent variable Z, that has 3 different values.

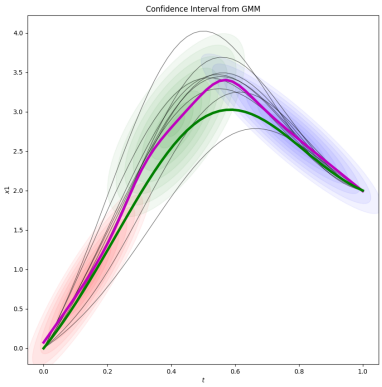

Fig. 5. Task B. The green line is the ground truth, whereas the magenta line is the inferred trajectory. The faded gray lines are the demonstration used for learning

The initialization of the GMM was quite important in determining the convergence. A Bayesian approach would most likely increase the convergence for more complex trajectories.

According to the results from the tables, though the GMR inferred trajectory was quite good, the bottleneck for the system was the Encoder-Decoder Block. The threshold for reconstruction is perhaps a too simple heuristic to decide important points. Additional Time Series analysis could provide further insights.

Lastly, the dictionary contained only 2 tasks. To further extend the work, better segmentation of complex tasks for multiple endpoints could test the generalization capabilities of this system.

V. CONCLUSIONS

Learning from Demonstration is a powerful and effective paradigm. However, a major limitation is that the robot can only become as good as the human's demonstrations. No other information is used to improve the robot's behaviour. To this end, Reinforcement Learning could allow the robot to investigate the state-action space and find an optimal policy. An idea from [11] would be to use LfD to generate an initial step of primitives.

Other approaches would be to enhance the probabilistic model (TP-GMM) by investigating new spaces. [12] presents an extension of imitation learning to Riemann manifolds. To enable this enables a more accurate input representation, that is by including joint distributions it is possible to infer the robot's orientations and pose.

REFERENCES

[1] A. Billard and D. Grollman, "Robot learning by demonstration," *Scholarpedia*, vol. 8, p. 3824, 12 2013.
[2] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent Service Robotics 2015 9:1*, vol. 9, pp. 1–29, 9 2015.
[3] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data Via the EM Algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, pp. 1–22, 9 1977.
[4] F. Dellaert, "The Expectation Maximization Algorithm," 2002.
[5] M. J. A. Zeestraten, "PROGRAMMING BY DEMONSTRATION ON RIEMANNIAN MANIFOLDS,"
[6] J. S. Ferenc and Z. Néda, "On the size distribution of Poisson Voronoi cells," *Physica A: Statistical Mechanics and its Applications*, vol. 385, pp. 518–526, 11 2007.
[7] G. Hamerly and C. Elkan, "Learning the k in k-means," *Advances in neural information processing systems*, vol. 16, pp. 281–288, 2003.
[8] A. Fabisch, "gmr: Gaussian Mixture Regression," *Journal of Open Source Software*, vol. 6, no. 62, p. 3054, 2021.
[9] D. Chicco, M. J. Warrens, and G. Jurman, "The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation," *PeerJ Computer Science*, vol. 7, p. e623, 7 2021.
[10] "Gaussian Mixture Models Clustering Algorithm Explained — by Cory Maklin — Towards Data Science."
[11] D. C. Bentivegna, C. G. Atkeson, and G. Cheng, "Learning tasks from observation and practice," *Robotics and Autonomous Systems*, vol. 47, pp. 163–169, 6 2004.
[12] M. J. Zeestraten, I. Havoutis, J. Silvério, S. Calinon, and D. G. Caldwell, "An Approach for Imitation Learning on Riemannian Manifolds," no. 1, 2017.
[13] "latex-example/GMM-EM.pdf at master · cloudygoose/latex-example."
[14] C. M. Bishop, *Pattern Recognition and Machine Learning*, vol. 4 of *Information science and statistics*. Springer, 2006.

VI. APPENDIX

Herehunder we report a mathmathical details of the model. Adapted from the slides of Prof Kai Yi. [13]

The Guassian distribution:

$$\mathcal{N}(x|\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\Sigma|^{1/2}}exp\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\} \tag{1}$$

The Guassian Mixture distribution is a linear superposition of Guassians:

$$p(x) = \Sigma_{k=1}^{K}\pi_k\mathcal{N}(x|\mu_k,\Sigma_k) \tag{2}$$

Subject to:

$$\Sigma_{k=1}^{K}\pi_k = 1 \tag{3}$$

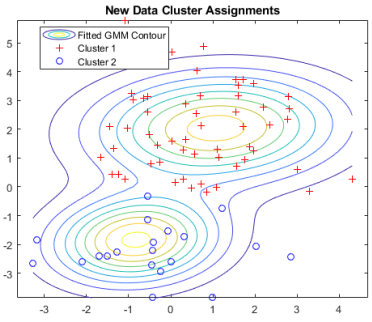

Fig. 6. Gaussian Mixture Model, 2D Example [10]

Now, for a Guassian Mixture Model, given the parameters:
k, the number of Guassian components
$\pi_1...\pi_k$, the mixture weights of the components
$\mu_1...\mu_k$, the mean of each component
$\Sigma_1...\Sigma_k$, the variance of each component
We can generate samples $s_1, s_2...s_n$ from the distribution.

Given a Guassian Mixture model, we introduce K-dimensional binary random variable z which only one element z_k is euqal to 1 and the others are all 0.

$$z = (0, 0, .., 1, 0, ..0) \tag{4}$$

So there are K possible states for z.And we let

$$p(z_k = 1) = \pi_k \tag{5}$$

That is to say,

$$p(z) = \prod_{k=1}^{K}\pi_k{}^{z_k} \tag{6}$$

Then, we define the conditional distribution of x given a particular z:

$$p(x|z_k = 1) = \mathcal{N}(x|\mu_k,\Sigma_k) \tag{7}$$

which can also be written as:

$$p(x|z) = \prod_{k=1}^{K}\mathcal{N}(x|\mu_k,\Sigma_k)^{z_k} \tag{8}$$

Now we can easily compute the marginal distribution of x

$$\begin{aligned} p(x) = \Sigma_z p(x|z)p(z) &= \Sigma_z \prod_{k=1}^{K}\mathcal{N}(x|\mu_k,\Sigma_k)^{z_k}\prod_{k=1}^{K}\pi_k{}^{z_k} \\ &= \Sigma_{k=1}^{K}\pi_k\mathcal{N}(x|\mu_k,\Sigma_k) \end{aligned}$$

Now, instead of working with $p(x)$ we can work with $p(x,z) = p(x|z)p(z)$, which will lead to significant simplification when we are introducing the EM algorithm.

Another quantity $p(z|x)$ will also be very important. We use $\gamma(z_k)$ to denote $p(z_k = 1|x)$, and we can use Bayes' theorem to deride its value.

$$\begin{aligned} \gamma(z_k) = p(z_k = 1|x) &= \frac{p(x|z_k = 1)p(z_k = 1)}{p(x)} \\ &= \frac{\pi_k\mathcal{N}(x|\mu_k,\Sigma_k)}{\Sigma_{j=1}^{K}\pi_j\mathcal{N}(x|\mu_j,\Sigma_j)} \end{aligned} \tag{9}$$

We usually say $\gamma(z_{nk})$ is the responsibility of component k for x_n.

Maximum likelihood Suppose we have a data set of observations $\{x_1, ..., x_N\}$. And we wish to model this data set using Guassian Mixture model. We could represent this data set as an N*D matrix **X**, where N is the number of data vectors and D is the dimension of the vector. Then the log likelihood function is given by

$$\ln p(X|\pi,\mu,\Sigma) = \Sigma_{n=1}^{N}\ln\{\Sigma_{k=1}^{K}\pi_k\mathcal{N}(x_n|\mu_k,\Sigma_k)\} \tag{10}$$

Parameter μ

The expectation-maximization algorithm is an elegant and

powerful method for finding maximum likelihood solutions for models with latent variables.
First, we set the derivatives of $lnp(X|\pi,\mu,\Sigma)$ in equation 10 with repect to μ_k to zero.

$$\begin{aligned} 0 &= -\Sigma_{n=1}^{N} \frac{\pi_k \mathcal{N}(x_n|\mu_k,\Sigma_k)}{\Sigma_{j=1}^{K}\pi_j \mathcal{N}(x_n|\mu_j,\Sigma_j)} \Sigma_k (x_n - \mu_k) \\ &= -\Sigma_{n=1}^{N} \gamma(z_{nk}) \Sigma_k (x_n - \mu_k) \end{aligned} \tag{11}$$

If we assume Σ_k to be nonsingular, we obtain

$$\mu_k = \frac{\Sigma_{n=1}^{N}\gamma(z_{nk})x_n}{\Sigma_{n=1}^{N}\gamma(z_{nk})} \tag{12}$$

Parameter Σ_k
We set $N_k = \Sigma_{n=1}^{N}\gamma(z_{nk})$, as the effective number of points assigned to cluster k.
If we set the derivative of $lnp(X|\pi,\mu,\Sigma)$ with respect to Σ_k to zero,we get

$$\Sigma_k = \frac{1}{N_k}\gamma_k(z_{nk})(x_n - \mu_k)(x_n - \mu_k)^T \tag{13}$$

Parameter π_k
Finally, when we maximize the log likelihood with respect to π, we need to take the constraint $\Sigma_{k=1}^{K}\pi_k = 1$ into consideration. This is done by using the Lagrange multiplier.

We want to maximize $f(x,y)$ subject $g(x,y) = c$.
Let $\Lambda(x,y,\lambda) = f(x,y) + \lambda(g(x,y) - c)$.Then if $(x_0,y_0)$is a maximum of the original f, there exists (x_0,y_0,λ_0) is a stationary point for the Λ function.

The contour lines of f and g touch when the tangent vectors of the contour lines are parallel. Since the gradient of a function is perpendicular to the contour lines, this is the same as saying that the gradients of f and g are parallel.

So $\nabla_{x,y} f = -\lambda \nabla_{x,y} g$.
Combining with the constraint, we get $\nabla_{x,y,\lambda}\Lambda = 0$

Now, we apply the Lagrange Multiplier to maximize with respect to π. We will be maximizing

$$\log p(x|\pi,\mu,\Sigma) + \lambda(\Sigma_{k=1}^{K}\pi_k - 1) \tag{14}$$

By maximizing it we will get

$$\pi_k = \frac{N_k}{N} \tag{15}$$

We have to note that the solutions 12, 13, 15 are not closed. Because the responsibilities $\gamma(z_{nk})$ depend on the parameters. However, they suggest a iterative scheme for finding a solution to the maximum likelihood problem.

E-M algorithm for Gaussian mixtures

Initialize Initialize the parameters μ_k, Σ_k, and π_k

E-step Evaluate the responsibilities using the current parameter values.

$$\gamma(z_{nk}) = \frac{\mathcal{N}(x_n|\mu_k,\Sigma_k)}{\Sigma_{k=1}^{K}\pi_k \mathcal{N}(x_n|\mu_k,\Sigma_k)} \tag{16}$$

$$N_k = \sum_n \gamma_{n,k} \tag{17}$$

M-step Re-estimate the parameters using the current responsibilities.

$$\begin{aligned} \mu_k^{new} &= \frac{1}{N_k}\Sigma_{n=1}^{N}\gamma(z_{nk})x_n \\ \Sigma_k^{new} &= \frac{1}{N_k}\gamma_k(z_{nk})(x_n - \mu_k)(x_n - \mu_k)^T \\ \pi_k^{new} &= \frac{N_k}{N} \end{aligned} \tag{18}$$

Likelihood Recalculate the log likelihood function to see if it converges, if not, go to step 1 again.

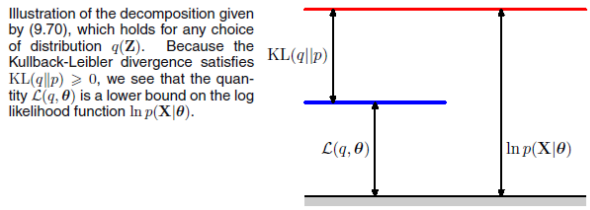

Fig. 7. **Intuition behind EM algorithm.** One can decompose the log-likelihood $p(X|\Theta)$ into two components a Lower-bound of a functional q and a Kullback-Leibler divergence between q and p (i.e. a measure of how much q is equal to p).
$p(X|\Theta) = L(q,\Theta) + KL(q||P)$.

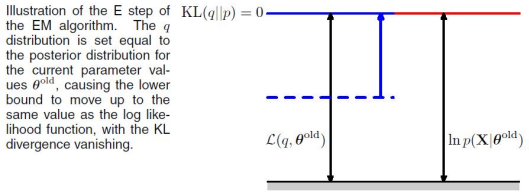

Fig. 8. From [14]

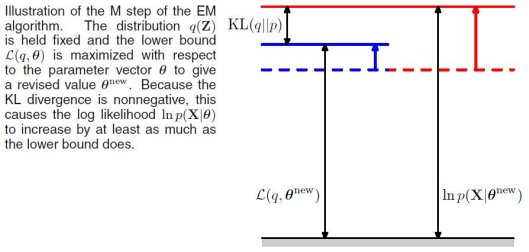

Fig. 9. From [14]