Simple Incremental GMM Modeling using Multidimensional Piecewise Linear Segmentation for Learning from Demonstration

Yasser Mohammad^{a,*}, Toyoaki Nishida^b

^aAssiut University, Assiut, Egypt

^bKyoto University, Kyoto, Japan

*Corresponding Author: yasserm@aun.edu.eg

Abstract

Learning from Demonstration is an important technology for the new wave of robots that are envisioned to work side-by-side with workers in factories as well as social robots. Most available techniques for learning from demonstration rely on the existence of a training set of demonstrations that is assumed to be pre-segmented and is usually processed in batch. Another problem with most available methods is the need to set the model complexity used to model the motion. In this paper, we propose a solution to both problems based on incremental piecewise linear segmentation of the motion using an extension of the SWAB algorithm. Evaluation experiments show that the proposed method is able to generate motion models with adequate complexity without the need for model comparison methods and assuming incremental streaming of demonstrations rather than the availability of the complete training set in advance.

The proposed method is applicable not only to robot learning from demonstration but to other industrial applications in which an accurate model of time variant data needs to be built from streaming input.

Keywords: imitation, learning from demonstration, GMM/GMR, SWAB.

1. Introduction

Learning from demonstration (LfD)⁽¹⁾ is one of the key technologies for robots working with humans in a shared workspace. LfD provides an easy to use modality for programming robots to achieve new tasks by naïve users who do not have special training in robot programming. This is of special importance for future industrial robots

that are expected to interact freely with human workers rather than being caged away doing precisely programmed repetitive tasks. This ability to learn from demonstrations would allow such robots to be easily adapted to different roles in the industrial workspace which is an important enabling technology in intelligent factories envisioned in Industry 4.0⁽²⁾ and similar initiatives.

LfD has a long history in robotics starting in the nineties with systems that could learn the function to be optimized in optimal control from demonstrations⁽³⁾.

Currently there are four main approaches to LfD: The first approach grew directly from optimal control formulation and is based on inverted reinforcement learning (IRL) in which the robot learns the reward function from the demonstrations and can then use RL for behaving⁽⁴⁾.

The second approach is exemplified by the Dynamic Motor Primitives (DMP) system⁽⁵⁾ which models the trajectory in one dimension as a dynamical system that is guaranteed to converge to the goal state. This dynamical system is then modulated by a set of kernel functions that allows it to behave similar to the demonstrated trajectory while still guaranteeing the convergence using a second dynamical system for modeling the current stage of the demonstration⁽⁶⁾. DMPs allow the robot to change the starting and goal positions when behaving as well as its rate of motion. It is not trivial though to combine multiple DMPs to form more complicated motion⁽⁷⁾. One important design decision for DMP systems is the number of kernels used and their location.

The third approach uses symbolization to represent the motion as a string and an algorithm for generating motion from this string with smoothing achieved using splines⁽⁹⁾. The demonstrations are first encoded using piecewise aggregate approximation then symbolic aggregate

approximation (SAX)⁽⁸⁾ is used for generating a symbolic representation. This approach was shown to provide high robustness to confusing demonstrations but suffer from inability to encode internal constraints of the task between different dimensions of the demonstration⁽⁹⁾.

The final LfD approach - which is the focus of this paper - is based on statistical modeling of the demonstrated trajectories. A successful example of this approach is using a Mixture of Gaussians for modeling the motion and Gaussian Mixture Regression for motion generation (GMM/GMR)⁽¹⁰⁾. This approach allows for easy modeling of the internal correlations between different dimensions of the demonstrated behavior (in contrast to DMPs for example) and it is trivial to combine multiple learned motions to generate more complex behaviors. It is not trivial though to modify the goal or rate of motion.

An important design decision in GMM/GMR is the number of Gaussians used for the GMM step. This is a model selection problem that is inherently hard to solve. Few approaches have been proposed to select this number.

The first approach is to try several numbers of Gaussians and use Bayesian Information Criteria (BIC) or a similar measure for model selection to balance prediction accuracy and model complexity⁽¹¹⁾. The problem of this approach is that it requires multiple learning runs and there is no guarantee that the *best* number of Gaussians lies within the range of numbers tried except if too many trials are performed.

Another approach that was proposed is to use a Dirichlet Process to model the motion with an effectively infinite number of Gaussians⁽¹²⁾. This approach requires the complete data set to be available at the time of learning and process trajectories in batch.

This paper proposes a simple approach that processes the trajectories incrementally (i.e. one point from each trajectory at a time) and selects an acceptable number of Gaussians based on piecewise linear modeling of the input trajectories.

The rest of the paper is organized as follows: Section 2 gives details of the proposed approach and section 3 provides comparative evaluation experiments of the proposed approach and discusses the implications of the results of these experiments. The paper is then concluded.

2. Proposed Approach

This section provides the rationale behind the proposed system and gives detailed description of each of

its steps.

2.1 System Rationale

Fig. 1 shows five example 2D trajectories and the corresponding GMM learned using the EM algorithm with 12 Gaussians. It is easy to see that the Gaussians correspond roughly to lines in the original dataset.

Fig. 1. Five 2D trajectories of letter “B” and the corresponding GMM learned using the EM algorithm.

This is not an accident. Equiprobable points of a 2D Gaussian is always an ellipse in which the transverse diameter is a line that rotates with changes in the covariance matrix and the conjugate diameter is another line that is affected by the determinant of this matrix. The location of the ellipse depends on the mean vector. From this, it is clear that a single Gaussian cannot faithfully represent more than a single line in the original distribution.

The main idea presented in this paper is to utilize this fact to base the number of Gaussians used on the number of straight lines in the input trajectory. This allows us to specify an acceptable model complexity without the need to use more complex model selection approaches. Moreover, using this technique, we can add Gaussians to the mixture whenever a new line is found in the trajectory.

In case of DMP, the rationale for using the number of lines in the trajectory to select the number of kernels used for nonlinear modifications of the original dynamical system is less straightforward and will not be discussed further in this paper.

The aforementioned discussion focused on 2D Gaussians but the idea presented can be extended to arbitrary dimensionality by noticing that a Multidimensional Gaussian can always be projected into a 2D Gaussian between any two of its dimensions.

In the case of motion trajectory, we can always use time as one of these two dimensions which means that we can use piecewise linear fitting at every other dimension (with time as the dependent variable) exactly as discussed for the 2D case.

2.2 Multidimensional Piecewise Linear Segmentation

There are three main approaches to piecewise linear segmentation of time-series: Top-Down, Bottom-Up, and sliding window approaches. In the first case, we start with a single line and then divide it optimally until the final segmentation is within some predefined error threshold. In Bottom-Up algorithms we start with a line for each two consecutive points then combine lines according to some policy until the final segmentation is just above the predefined threshold. In the final approach we start from the first point and continue to add points to the line until this segment is above a predefined threshold then a new line is introduced.

The main advantage of the Bottom-Up approach is that it usually generates superior results compared with the Top-Down approach and the main advantage of the sliding window approach is that it is incremental with the drawback of higher global error compared with other methods.

The SWAB algorithm was proposed by Keogh et al. to combine the advantages of sliding window and Bottom-Up approaches⁽¹³⁾.

The algorithm alternates between the sliding window and bottom-up algorithms to incrementally create a piecewise linear approximation of the input timeseries.

The main limitation of the SWAB algorithm for our purposes is that it is a single dimension algorithm.

The first contribution of this paper is the extension of SWAB to multiple dimensions. Notice that for our application, as discussed in section 2, we treat one of the dimensions differently (time) since it is the dependent variable for all other dimensions. This means that rather than worrying about correlations between different dimensions and fitting a hyperplane to each segment, we can just use multiple line fits of every dimension and then combine the resulting residues in deciding the best possible fit. To achieve this goal, a residue collapse function is introduced as a weighted average of individual residues.

$$r = \sum_{i=1}^N w_i * r_i \quad (1)$$

The weights in this equation can be selected by the user to encode domain knowledge but in our experiments it was automatically calculated as the coefficient of variation (CV) associated with this dimension (i.e. the ratio between the standard deviation and the mean).

$$w_i = \sigma_i / \mu_i \quad (2)$$

2.3 Dealing with Multiple Demonstration

The multidimensional extension of SWAB explained in section 2.2 allows us to model the incoming stream representing a demonstration incrementally but it does not deal with the case when multiple demonstrations are available.

There are several ways to deal with multiple incoming streams and in this paper we compare three of them.

The first and simplest approach is to treat all streams as a single demonstration with higher dimensionality and apply multidimensional SWAB to them together. This approach is called *Aggregate-MSWAB* (*aMSWAB*) hereafter.

A second simple method is to find the mean at every time step for all demonstrations and then apply multidimensional SWAB to the mean. This approach is called *Mean-MSWAB* (*mMSWAB*) hereafter.

The final approach is to apply multidimensional SWAB to every demonstration alone and then use Dynamic Programming to align the cut-points between consecutive lines in the piecewise linear models generated. This approach has the virtue of being applicable to the case when multiple demonstration are streamed incrementally one point at a time and also to the case when complete demonstrations are streamed one demonstration at a time. This approach is called *Individual-MSWAB* (*iMSWAB*) hereafter.

2.4 Incremental GMM Modeling

Give the piecewise linear model generated using the streamed demonstration (as described in section 2.2) or compiled from multiple demonstrations (as described in section 2.3), we can then generate the Gaussian Mixture Model (GMM) of the demonstration without the need for expensive application of Bayesian Information Criteria and similar approaches.

In this paper, two approaches are being compared for GMM modeling. The first approach (MSWAB+KMeans) uses the number of lines returned from the multidimensional SWAB application as the number of Gaussians to be used (see section 2.1 for the rationale behind this decision). K-means is then applied using this number of Gaussians followed by Expectation Maximization (EM) to generate the mixture model of the form:

$$p(y; \pi_k, \mu_k, \Sigma_k) = \sum_{k=1}^K \pi_k \frac{1}{(2\pi)^{D/2} |\Sigma_k|^{1/2}} e^{-\frac{1}{2}(y-\mu)^T \Sigma_k^{-1} (y-\mu)} \quad (3)$$

where $y = [t, x]^T$ and x is the input demonstration.

The second method is called Individual Line Fitting (MSWAB+ILF). In this case, we use the cut points of each line returned by multidimensional SWAB as the boundaries for a Gaussian under the assumption that a single Gaussian encodes approximately a single line in the piecewise linear fit. The original data is then used to fit the Gaussian to the points belonging to this line. The prior is set to an equal values for all of these Gaussians (i.e. $\pi_k = 1/K$) then Expectation Maximization is used to improve the likelihood of the model. If the data is streaming then the EM algorithm needs to be applied only when a new line and a new Gaussian is generated.

2.5 Behavior Generation using GMR

Behavior generation is achieved by Gaussian Mixture Regression (GMR). Given the GMM model returned from incremental Gaussian Modeling (see section 2.4), it is possible to generate the behavior corresponding to the demonstrations by simply conditioning on the time dimension of the learned Gaussians⁽¹⁰⁾.

This is achieved as follows. Firstly, the mean and covariance matrices are segmented into time and other dimensions:

$$\mu_k = \begin{bmatrix} \mu_k^t \\ \mu_k^x \end{bmatrix}, \Sigma_k = \begin{bmatrix} \Sigma_k^t & \Sigma_k^{tx} \\ \Sigma_k^{tx^T} & \Sigma_k^x \end{bmatrix} \quad (4)$$

An output trajectory for each component is then calculated as:

$$\bar{x}_k(t) = \mu_k^x + \Sigma_k^x \Sigma_k^{t-1} (t - \mu_k^t) \quad (5)$$

A weight for each of these trajectories is calculated using equation (6).

$$w_k(t) \propto \frac{\pi_k}{\sqrt{2\pi \Sigma_k^t}} e^{-\frac{(t-\mu_k^t)^2}{2\Sigma_k^t}} \quad (6)$$

where $\sum_{k=1}^K w_k = 1$.

The final trajectory output (x^o) is then calculated as a weighted summation of the component trajectories.

$$x^o(t) = \sum_{k=1}^K w_k(t) \bar{x}_k(t) \quad (7)$$

The main advantage of this approach is that it does not only generate a model behavior representing the demonstrations but it also generated a variability score at each point that can inform the controller about the allowed deviation from this model behavior at every time-step. For example, if the demonstrations represent pouring actions, the beginning and ending of the motion is much more important to be accurate than the exact path followed in the middle. This will appear as higher variability in the demonstrations in the middle of the motion and will be translated to higher variability score output from the GMR algorithm.

3. Evaluation and Discussion

The proposed approach provides two different methods for GMM fitting (i.e. KMEANS and ILF) and three different methods for combining multiple demonstrations (i.e. IDF, MEAN and AGGREGATE). This gives six variations of the proposed MSWAB based system for learning from demonstrations (LfD).

The first evaluation experiment compared each of these variations with two other approaches for selecting the number of Gaussians used for modeling. Namely, BIC and SAX-GMM⁽⁹⁾. The Bayesian Information Criteria (BIC) was used Calinon et al.⁽¹¹⁾ to find an appropriate number of Gaussians for LfD applications, while SAX-GMM was recently proposed by Mohammad and Nishida for robust LfD⁽⁹⁾. Both approaches require the full dataset of demonstrations be available while the proposed approach processes each point of the demonstration incrementally.

Five subjects were recruited. Each subject drew two to five demonstrations of a single stroke character (e.g. 'C', 'B', 'α', etc) or figure (e.g. square, circle, etc). One of the datasets was corrupted and removed from the experiment. The final nine drawings (each having 2-5 demonstrations) were used as input to each of the six variations of the proposed MSWAB based algorithm as well as BIC-GMM and SAX-GMM. Fig. 2 shows one example from this

experiment. Five demonstrations of the letter 'a' were used to train each of the algorithms. The figure shows the five demonstrations, the single standard deviation equiprobable ellipse of learned Gaussians and the final model generated from the GMR algorithm. The expectation of the variability returned by GMR is not shown to reduce clutter. The figure shows that all algorithms more or less succeeded in modeling the character. SAX-GMM had the smallest number of Gaussians (5) which led to poor representation at the lowest right and top left corners of the character. BIC-GMM had the largest number of Gaussians (20) which caused some wiggle in the same location. The six variations of the proposed algorithm used between 14 and 16 Gaussians, provided a better balance and did not suffer from neither of the aforementioned problems with other approaches.

This is not a one-time phenomenon. Fig. 3 shows a box-plot of the number of Gaussians learned using each of the algorithms. As the figure shows, the average number of Gaussians was minimum for SAX-GMM (5.5) which is biased toward simpler models and maximal for BIC (19.5) and aMSWAB (26.5) which is biased toward complex models. iMSWAB and mMSWAB provided a middle ground between these two extremes with 16 and 15 Gaussians on average respectively. The large number of Gaussians induced by aMSWAB is due to the constraint of keeping the RMS error in each line within the accepted accuracy for ALL demonstrations at once. This constraint led to over segmentation of the trajectory into more lines than is the case for iMSWAB and mMSWAB.

Fig. 4 shows a box plot of the execution time of the eight algorithms compared in the first experiment. As the Figure shows, the fastest algorithm was SAX-GMM needing only 1.35seconds for each demonstration. This is expected as a single GMM fitting is executed and the algorithm uses piecewise constant rather than piecewise linear modeling of the data. The slowest algorithm is BIC-GMM which needed 14.36 seconds to process each demonstration on average. The six variations of the proposed algorithm had similar execution times ranging from 7 to 11.6seconds per demonstration. mSWAB variants had the least variance ($\sigma=7.8s$) so they are the most appropriate for real-time or time-sensitive applications. All of the proposed variants were faster than BIC-GMM even though they executed data incrementally.

Fig. 2. Five demonstrations of the character 'a' with the corresponding learned GMM and final model generated by GMR for the 8 algorithms compared.

Fig. 3. Number of Gaussians learned for each of the evaluated algorithms.

Fig. 4. Average Execution Time of Different Algorithms in the first experiment (2D drawing data).

Summarizing the results of this experiment, the proposed approach could provide a balance between execution time and the complexity of learned model. It provided a middle ground between the fast low complexity biased SAX-GMM algorithm and the flow high complexity biased GMM-BIC method. It has the extra advantage of processing the input demonstrations incrementally.

Fig. 5. Execution Time of Different Algorithms in the second experiment (CMU MoCap data).

The second experiment involved applying the same algorithms to realistic Motion Capture data representing human actions. We used the CMU MoCap dataset⁽¹⁵⁾. The dataset contains Motion Capture data and videos of people executing several sport and real world motions. We selected the time-series from the basketball category. A video annotation tool (ANVIL⁽¹⁶⁾) was used to mark the beginning and ending of recurring patterns of motion in the dataset. These time-series contained 21 motion patterns (e.g. shooting, dripping, etc). The data for each motion contained 62 dimensions (instead of 2 in the previous experiment). We applied the same algorithms to this dataset. Each pattern of motion was pre-processed to remove dimensions that did not change much during the demonstration (e.g. dimensions with low variance) before being passed to the eight evaluated algorithms. This was achieved by removing all dimensions with a variance less than 62×10^{-16} .

Fig. 5 shows the execution time of different algorithms in this experiment. Again, SAX-GMM was the fastest algorithm with an average execution time of 1.2 seconds. The proposed mMSWAB was the second fastest with an

average execution time of 7.5 seconds for the mMSWAB-KMeans variant and 7.2 seconds for the mMSWAB-ILF variant. The slowest two algorithms were BIC (29.75s) and iMSWAB variants (35.5s and 35.2s). iMSWB was much slower in this case compared with the first experiment because the increase in the input dimensionality led to slower line fitting that was exaggerated by the number of demonstrations. mMSWAB and aMSWAB did not suffer from this problem as the line fitting is always done to a single time-series.

To calculate the accuracy of various algorithm in modeling the motion, we used leave-one-out cross validation by training the algorithms on all demonstrations but one then comparing the final model with this left out demonstration. This process was repeated for all demonstrations leaving a different one out each time calculating the error each time and then averaging it. This average error was normalized by dividing it by the error in the BIC case.

Fig. 6. Leave-one-out Normalized Error for different algorithms in the second experiment (CMU MoCap data).

Fig. 6 shows the results of this analysis. iMSWAB-KMeans was the most accurate of all the algorithm (despite the fact that it is an incremental algorithm) with a normalized error of 0.9 while SAX-GMM was the worst with a normalized error of 3.

The second experiment shows that the proposed approach provides a comparable accuracy to slower approaches like BIC for model selection while having the advantage of being incremental.

One limitation of the proposed method is that it

requires pre-segmentation of motion trajectories to examples of the demonstration in advance. Combining this approach with action segmentation within the fluid imitation framework⁽¹⁴⁾ will be one of the future directions of research.

Another limitation of the proposed approach is the implicit assumption that the dependent variable is time. This means that it cannot be used to model the motion as an autonomous dynamical system (i.e. $\dot{x} = f(x)$ for state variable x). Extending the system to this type of modeling will require replacing line fitting with hyperplane fitting in the MSWAB step.

The proposed approach processes data in the chronological order, which is an advantage for streaming sources. Nevertheless, in most cases in robotics LfD, demonstrations are provided one at a time and where demonstrations from different skills are interleaved. Learning under these circumstances is more challenging but necessary for life-long acquisition of behaviors in robots. The idea of piecewise linear segmentation combined with GMM modeling can be extended to such cases and will be the focus of our future research.

4. Conclusion

This paper presented a novel approach for incremental GMM modeling of motion trajectories for industrial robots that share their workspace with naïve human operators/partners. The proposed method is based on an extension of the SWAB algorithm to multiple dimensions then using the learned lines as starting points for the GMM or HMM modeling. The proposed approach is not optimal in the sense that it is not guaranteed to learn the best mixture (in minimum least square error sense) of the data of a given number of Gaussians but it has the advantages of needing no specification of the number of Gaussians in advance. Moreover, the proposed approach can be used to learn the Gaussians incrementally by processing points of the trajectories one by one rather than by processing trajectories serially.

Evaluation of the proposed method showed that it can achieve comparable performance to batch approaches even without the need to specify the number of Gaussians in advance.

Acknowledgment

This research is partially supported by the Center of Innovation Program from Japan Science and Technology Agency, JST and AFOSR/AOARD Grant No. FA2386-14-1-0005.

References

- (1) Argall, Brenna D., Sonia Chernova, Manuela Veloso, and Brett Browning. "A survey of robot learning from demonstration". *Robotics and autonomous systems*, Vol. 57, No. 5, pp. 469-483, 2009
- (2) Lee, Jay, Hung-An Kao, and Shanhu Yang. "Service innovation and smart analytics for industry 4.0 and big data environment". *Procedia CIRP* 16, 3-8, 2014
- (3) C. G. Atkeson and S. Schaal. "Learning tasks from a single demonstration". In *IEEE IROS*, Vol. 2, pp. 1706-1712, 1997
- (4) G. Konidaris, S. Kuindersma, R. Grupen, and A. S. Barreto. "Constructing skill trees for reinforcement learning agents from demonstration trajectories. In *Advances in neural information processing systems*", pp. 1162-1170, 2010
- (5) A. J. Ijspeert, J. Nakanishi, and S. Schaal. "Learning attractor landscapes for learning motor primitives". In *Advances in neural information processing systems*, pp. 1523-1530, 2002
- (6) S. Khansari-Zadeh and A. Billard. "Learning stable nonlinear dynamical systems with Gaussian Mixture Models". *IEEE Transactions on Robotics*, Vol. 27, No. 5, pp. 943-957, 2011
- (7) Manschitz, Simon, Jens Kober, Michael Gienger, and Jan Peters. "Learning to sequence movement primitives from demonstrations". In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4414-4421, 2014
- (8) J. Lin, E. Keogh, S. Lonardi, and B. Chiu. "A symbolic representation of time series, with implications for streaming algorithms". In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pp. 2-11, 2003
- (9) Yasser Mohammad and Toyooki Nishida, "Robust Learning from Demonstration using Multidimensional SAX", 2014 14th International Conference on Control, Automation and Systems (ICCAS), pp. 64-71, 2014
- (10) Sylvain Calinon, A. Pistillo, and D. G. Caldwell. "Encoding the time and space constraints of a task in

- explicit-duration Hidden Markov Model”. In IEEE IROS, pp. 3413–3418, 2011
- (11) Sylvain Calinon, Florent Guenter, and Aude Billard. “On learning the statistical representation of a task and generalizing it to various contexts” In IEEE International Conference on Robotics and Automation, (ICRA), pp. 2978-2983, 2006
- (12) Chatzis, Sotirios P., Dimitrios Korkinof, and Yiannis Demiris. “A nonparametric Bayesian approach toward robot learning by demonstration” *Robotics and Autonomous Systems*, Vol. 60, No. 6, 789-802, 2012
- (13) Keogh, E., Chu, S., Hart, D. & Pazzani, M. “An Online Algorithm for Segmenting Time Series”. In *Proceedings of IEEE International Conference on Data Mining*, pp. 289-296, 2001
- (14) Mohammad, Yasser, and Toyooki Nishida. “Fluid imitation”. *International Journal of Social Robotics*, Vol. 4, No. 4, 369-382, 2012
- (15) CMU motion capture dataset. <http://mocap.cs.cmu.edu>
- (16) M. Kipp. Anvil - a generic annotation tool for multimodal dialogue. In *Proceedings of the 7th European Conference on Speech Communication and Technology (Eurospeech)*, pp. 1367–1370, 2001