Learning to Serve: An Experimental Study for a New Learning From Demonstrations Framework

Okan Koç and Jan Peters

***Abstract*—Learning from demonstrations is an easy and intuitive way to show examples of successful behavior to a robot. However, the fact that humans optimize or take advantage of their body and not of the robot, usually called the *embodiment problem* in robotics, often prevents industrial robots from executing the task in a straightforward way. The shown movements often do not or cannot utilize the degrees of freedom of the robot efficiently, and moreover can suffer from excessive execution errors. In this letter, we explore a variety of solutions that address these shortcomings. In particular, we learn sparse movement primitive parameters from several demonstrations of a successful table tennis serve. The number of parameters learned using our procedure is independent of the degrees of freedom of the robot. Moreover, they can be ranked according to their importance in the regression task. Learning few parameters, which are ranked, is a desirable feature to combat the curse of dimensionality in reinforcement learning. Real robot experiments on the Barrett WAM for a table tennis serve using the learned movement primitives show that the representation can capture successfully the style of the movement with few parameters.**

***Index Terms*—Learning from Demonstration, Learning and Adaptive Systems, Optimization, Learning a Sparse Representation.**

I. Introduction

HUMANS are good at using their bodies to great effect, taking advantage of their muscular structure and soft but flexible actuation. Much of dexterous manipulation, or dynamic movement generation reflects this awareness of the human body. When teaching the robots to achieve similar tasks autonomously, however, we inevitably impose and transfer our biases to the robot. This problem of *embodiment* can cripple the execution, possibly also preventing the robots from taking advantage of their kinematics structure and actuation mechanisms.

In dynamic games like table tennis, we can easily observe humans taking utmost advantage of their bodies and pushing it to its maximum, i.e., optimizing their output bearing in mind their kinematic and dynamic limits. Table tennis serves, for instance, incorporate flicks (very fast accelerations of the wrist) that are designed to give an unsuspected spin and motion profile to the ball. Teaching such movements to the robots in a learning from demonstrations framework using kinesthetic teach-in, where the robot joint movements are recorded, suffers in particular from two drawbacks. Firstly, during the shown movement, as discussed above, the human is unable to move the shoulder joints of the robot adequately, which could potentially be used by the robot to great effect. Secondly, the fast movements of the wrists may not be tracked accurately by the robot, which is the case for the cable-driven seven degree of freedom (DoF) Barrett WAM arm, see Figure 1.

In this letter, we explore different learning from demonstrations (LfD) approaches to compensate for the execution and transfer deficiencies resulting from the demonstrated serves. The demonstrations are acquired and the movement primitives are trained in the joint-space of the robot, using kinesthetic teach-in, where the movements of the robot are recorded using the joint-level sensors. The initial policy or the movement template, extracted as a set of movement primitives, can be thought of as a good initialization for a reinforcement learning (RL) agent. By capturing the essence of the shown demonstrations in as few parameters as possible, we simplify and increase the effectiveness of the skill transfer to the robot.

Sparsity is achieved in our framework in joint-space[1] by using a new iterative optimization approach, where a multi-task Elastic Net regression is alternated with a nonlinear optimization. The Elastic Net projects the solutions to a sparse set of features, and during the nonlinear optimization these features (the basis functions) are adapted to the data in a secondary optimization. Moreover these features are shared across multiple demonstrations, increasing the effectiveness of the feature learning strategy.

The fewer number of learned parameters using our iterative optimization procedure, compared to more traditional approaches, is independent of the robot DoF. This is a desirable property for Reinforcement Learning to adapt the learned parameters online. Moreover, by using the Elastic Net path, we can rank the parameters in terms of importance, or effectiveness in explaining the demonstration data. We perform preliminary

Manuscript received September 10, 2018; accepted January 13, 2019. Date of publication January 30, 2019; date of current version February 21, 2019. This letter was recommended for publication by Associate Editor M. Howard and Editor D. Lee upon evaluation of the reviewers' comments. *(Corresponding author: Okan Koç.)*

O. Koç is with the Max Planck Institute for Intelligent Systems, Tübingen 72076, Germany (e-mail: okan.koc@tuebingen.mpg.de).

J. Peters is with the Max Planck Institute for Intelligent Systems, Tübingen 72076, Germany, and also with the FG Intelligente Autonome Systeme Hochschulstr, Technische Universitaet Darmstadt, Darmstadt 64289, Germany (e-mail: peters@ias.tu-darmstadt.de).

This letter has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/LRA.2019.2896466

[1]Discarding the joint-level information and using only the Cartesian coordinates of the resulting movements, in a similar attempt to reduce the dimensionality of the robot learning problem, necessitates the use of inverse kinematics, running into feasibility and additional execution problems that might be artificially introduced.

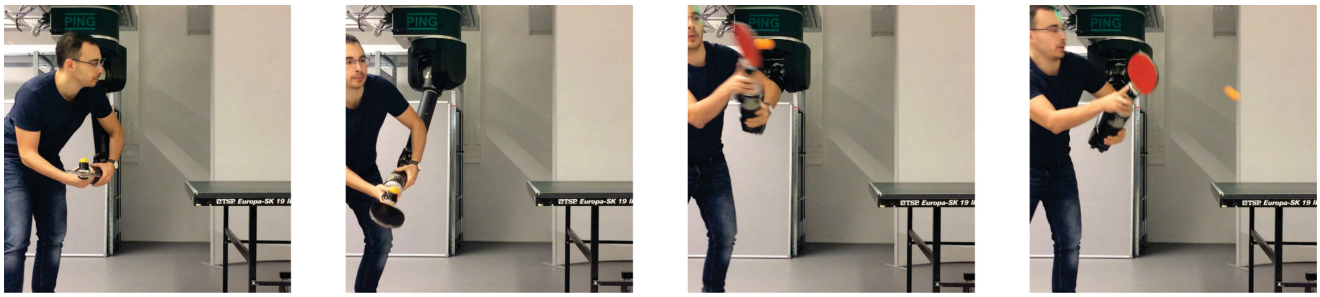

Fig. 1. Our robot table tennis setup with a seven DoF Barrett WAM, where we demonstrate, using kinesthetic teach-in, multiple good table tennis serve movements while recording the resulting joint-space robot trajectories. A metal piece is attached to the end effector of the Barrett WAM, which connects to a standard sized table tennis racket. An egg-holder on the metal piece holds the ball initially before the serve. The demonstrator, after finding a good starting posture, starts by swinging the arm, giving the ball enough acceleration to propel it away from the robot. The ball is then hit in midair by a careful adjustment of the robot wrist. The initial posture, the swinging movement of the robot shoulder joints and the elbow, and finally the turning of the wrist all contribute to the style of the shown movement. Multiple demonstrations starting from different initial postures are recorded in one session. We compare and evaluate throughout the letter different learning from demonstrations approaches using these demonstrations. We propose a new iterative optimization approach that can learn sparse parameters while adapting the features of the movement primitives to the demonstration data.

experiments on the Barrett WAM on a table tennis serve to validate the effectiveness of our new movement primitives.

Robot table tennis has, since the nineties, captivated the attention of the robot control and learning communities as a challenging and dynamic task, and research in it has been ongoing ever since. After the pioneering work of Anderson's analytical player [1], there have been various approaches focusing on certain parts of the game, such as simplifications in trajectory generation using a virtual hitting plane [2], [3] or learning striking trajectories from demonstrations [4]. Learning approaches to generate better strikes with Reinforcement Learning (RL) include [5], [6]. Recently, Koc et al. [7] have introduced a new trajectory generation framework in table tennis, where they solve a free final-time optimal control problem, generating minimum acceleration striking trajectories. This kinematic optimization approach was extended and evaluated in the real robot table tennis setup in [8].

The success of this and other similar model-based optimization approaches in dynamic tasks like table tennis heavily depends on the accuracy of the models. In the case of table tennis, an accurate *ball model* [9], [8] is especially difficult to acquire. The high spin rates make the ball flight difficult to model from first (physical) principles, while the various types of impacts make it also difficult to train machine learning approaches from raw ball position data. For the serve, an additional complication results from the ball take-off phenomena, which is similarly difficult to model or to learn.

Learning from demonstrations (LfD) is a promising framework for learning various robotic tasks efficiently without using hard-coded approaches or physical insights to model the specific aspects of each task. It has been used in many different robot scenarios to great effect, including robot manipulation and human-robot collaboration [10]. It was also useful in initializing the parameters of policy-search RL approaches for robot learning [11]. There are, by now, many different frameworks for LfD, including dynamical system representations such as the Dynamical Movement Primitives (DMP) [12], learning control Lyapunov-functions [13], and various other probabilistic approaches, such as the probabilistic movement primitives [14] or Gaussian mixture models [15]. These last two methods can, unlike DMPs, capture multiple demonstrations in a parametric form, and can moreover be used to condition on way-points or different targets in joint or in task space. One particular disadvantage of all of the LfD approaches introduced above is that the features chosen to regress on the demonstration data are often manually tuned and the number of parameters to learn are explicitly specified. We think that fixing the features and tuning their hyperparameters for particular tasks harm the generalization and applicability of the movement primitives to novel scenarios.

The l_1-regularized l_2-norm regression (from hereon referred to as *Lasso*) is often used in the statistics and machine learning communities as a regression method that can simultaneously also perform automatic feature selection. A detailed introduction and analysis of Lasso can be found in [16]. Lasso was extended to the *multi-task* case (i.e., multi-output regression with shared features) in [17]. Our interest in Lasso lies in the fact that (multi-task) Lasso can perform systematic feature selection while training (multiple) movement primitives, augmenting the applicability of LfD to novel tasks. Moreover, selection and early pruning of features can be used to great effect in RL, possibly reducing the amount of interaction time with the real robot.

A new incremental procedure to solve ordinary least squares regression as well as Lasso problems was proposed in [18]. This algorithm, called *Least Angle Regression* or *LARS* for short, yields piecewise linear homotopy paths of the regression problem as a function of the l_1-regularization term. These paths can be used to rank the features in terms of importance, as will be detailed later. Ranking the features of the trained movement primitives can reduce the curse of dimensionality in RL, decreasing as before the robot interaction time and possibly making the adapted movements also more interpretable to the humans.

The *Elastic Net* imposing additional l_2-regularization to Lasso was introduced in [19], where it was noted that a basic transformation converts the problem to a standard Lasso regression, and this is also valid in the multi-task setting. For the training of movement primitives, especially for dynamic

trajectories like the table tennis serves, the *Elastic Net* with its l_2-regularization can help to reduce the excessive accelerations throughout the learned movements, making them safer to implement on the robot.

In the next sections, we will detail how the sparse representation-learning of movement primitives can be formulated using the multi-task Elastic Net, coupled with nonlinear optimization on the feature parameters. To the best of our knowledge, the multi-task Elastic Net was not combined before with Radial Basis Functions in a (iterative) nonlinear feature selection and optimization framework. We also think that ranking the learned parameters in terms of importance is a new idea that can benefit the RL community.

II. Notation

The notation that we use throughout the letter is standard: for a robot arm with n degrees of freedom (DoF), the joint configurations are $\mathbf{q} \in \mathbb{Q} = \{\mathbf{q} \in \mathbb{R}^n \,|\mathbf{q}_{\min} \leq \mathbf{q} \leq \mathbf{q}_{\max}\}$. The recorded joint positions over a movement are represented as a matrix $\mathbf{q}(t) \in \mathbb{R}^{N\times n}$ of N rows, with column $i = 1, \ldots, n$ storing the positions throughout the movement corresponding to joint i.

Whenever multiple demonstrations are used for learning, i.e., $\mathbf{q}_{ij}(t)$ is recorded for $i = 1, \ldots, n$ DoF and $j = 1, \ldots, d$ demonstrations, these recordings are stacked to form the $\mathbf{Q}$ matrix. The degrees of freedom are concatenated vertically in this case for a single demonstration, while the columns store the different demonstration data, i.e., $\mathbf{q}_{ij}(t) \rightarrow \mathbf{Q}_{N(i-1)+t/dt,j}$ for a recording of N time points with dt time intervals.

The Frobenius norm of a matrix is the square-root of the sum of its squared elements, $\|\mathbf{M}\|_F^2 = \sum_i \sum_j m_{ij}^2$, whereas the $\|\cdot\|_{21}$ norm used in the multi-task Elastic Net is defined instead as $\|\mathbf{M}\|_{21} = \sum_i \sqrt{\sum_j m_{ij}^2}$, i.e., l_2-norm along the columns (degrees of freedom in our setting) and l_1-norm along the rows (time steps). This norm is used to induce sparsity on the features, whose centers are initially located uniformly along the time axis.

III. Method

In this section, we discuss how one can acquire a sparse movement pattern from human demonstrations. We present first an algorithm that requires only a single human demonstration, and then present a suitable variant that can be employed for multiple demonstrations. This variant of the algorithm decouples the number of learned parameters from the degrees of freedom of the robot.

A. Learning a Sparse Representation From a Single Demonstration

Given a single demonstration $\mathbf{q}(\mathbf{t})$ at the (observed) time points $\mathbf{t}$, we'd like to extract a movement primitive that is sparse. That is, throughout the parametric optimization, we'd like to impose a good fit with as few basis functions as possible, while keeping the accelerations low during the trained movement pattern. Having low accelerations is beneficial both for robot safety as well as improving the tracking (execution) accuracy of the trajectories [8]. Mathematically, the criterion that we optimize can be written as

$$\min_{\boldsymbol{\beta},\boldsymbol{\theta}} \|\mathbf{q}(\mathbf{t}) - \boldsymbol{\Psi}(\mathbf{t},\boldsymbol{\beta})\boldsymbol{\theta}\|_F^2 + \lambda_1\|\boldsymbol{\theta}\|_{21} + \lambda_2\|\ddot{\boldsymbol{\Psi}}(\mathbf{t},\boldsymbol{\beta})\boldsymbol{\theta}\|_F^2, \tag{1}$$

where $\boldsymbol{\Psi}(\mathbf{t},\boldsymbol{\beta}) \in \mathbb{R}^{N\times p}$ are the evaluations of the basis functions at $\mathbf{t}$, $\boldsymbol{\theta} \in \mathbb{R}^{p\times n}$ are the (sparse) regression parameters, and $\mathbf{q}(\mathbf{t})$ are the joint observations during the shown movement. The nonlinear radial basis functions (RBF) are parameterized by $\boldsymbol{\beta} \in \mathbb{R}^p$. An l_2-penalty is put on the accelerations $\ddot{\boldsymbol{\Psi}}(\mathbf{t},\boldsymbol{\beta})\boldsymbol{\theta}$ of the extracted movement pattern $\boldsymbol{\Psi}(\mathbf{t},\boldsymbol{\beta})\boldsymbol{\theta}$, while a penalty with the l_1-norm on the (rows of the) regression parameters $\boldsymbol{\theta}$ encourages sparsity of the found solutions.

This regression problem, for fixed $\boldsymbol{\beta}$, is known as the multi-task *Elastic Net* in the literature, where the features are shared among the sparse parameters along each degree of freedom. As opposed to the standard (multi-task) Lasso, the l_2-norm penalty in the optimization (1) penalizing the accelerations throughout the motion, also adds stability to the Lasso solutions [19].

The solution to the weighted Elastic Net problem (1) for fixed $\boldsymbol{\beta}$ can be obtained by transforming the problem to an equivalent (unweighted) Lasso problem, solving it via a convex optimizer (e.g., *coordinate descent* is very effective for Lasso problems), and then transforming the solutions back to the Elastic Net parameters.

We can solve the original problem (1) iteratively (as in Expectation-Maximization type of algorithms) by first starting the iteration with a Lasso solution of an overly-parameterized radial basis function regression. At each iteration, the RBF parameters $\boldsymbol{\beta}_i$ corresponding to the basis functions with nonzero Lasso regression parameters $\boldsymbol{\theta}_{ij} > 0$, $j = 1, \ldots, n$ are updated for each $i = 1, \ldots, p$ via nonlinear optimization. The Elastic Net regression is then performed, and the features corresponding to parameters with zero coefficients are removed. These two alternating steps can be continued till convergence, or rather terminated in a fixed number of steps. The iterations converge when the change in function value of the total cost in (1) is below a certain tolerance ϵ. Depending on the initial solution parameters $\boldsymbol{\beta}_0$ and $\boldsymbol{\theta}_0$, the iteration converges to a local minimum.

The full procedure is shown in Algorithm 1 in detail. We call the resulting algorithm *Learning Sparse Demonstration Parameters* or *LSDP* for short. The algorithm alternates between the multi-task Elastic Net (lines 4 and 10) and the nonlinear optimizer (BFGS, in line 8). In between, the zero entries of the regression parameters $\boldsymbol{\theta}$ and the corresponding columns of $\boldsymbol{\Psi}$, $\ddot{\boldsymbol{\Psi}}$ are removed in the Prune step (lines 5 and 13). The pruning operation simplifies the optimization in the upcoming iterations, as the removed RBF parameters cannot then be re-elected later. We use the squared exponential kernel to construct our basis functions, i.e., for every i, j we use

$$\boldsymbol{\Psi}_{ij}(t_i) = \exp(-(t_i - \boldsymbol{\mu}_j)^2/(2\boldsymbol{\sigma}_j^2)),$$

to form the (i, j)'th element of the matrix $\boldsymbol{\Psi}$. The data is initially centered (line 2), i.e., the mean of each joint recording is subtracted from the signal, and the means $\mathbf{q}_0$ are stored as the intercepts for the particular demonstration.

Algorithm 1: Learning Sparse Parameters with Regression ($LSDP$) for a Single Demonstration.

Require $\mathbf{q}$, $\mathbf{t}$, $\boldsymbol{\mu}$, $\boldsymbol{\sigma}^2$, λ_1, λ_2, $\epsilon > 0$
1: Initialize $\boldsymbol{\beta}_0 = [\boldsymbol{\mu}, \boldsymbol{\sigma}^2]$
2: Center the data, $\mathbf{q}_0, \mathbf{q} \leftarrow$ `Center`$(\mathbf{q})$
3: Form $\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}$ using $\boldsymbol{\beta}_0$ and $\mathbf{t}$
4: $\boldsymbol{\theta}_0 \leftarrow$ `MultiTaskElasticNet`$(\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}, \mathbf{q}, \lambda_1, \lambda_2)$
5: $\boldsymbol{\theta}_0, \boldsymbol{\beta}_0 \leftarrow$ `Prune`$(\boldsymbol{\theta}_0, \boldsymbol{\beta}_0)$
6: Form $\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}$ using $\boldsymbol{\beta}$ and $\mathbf{t}$
7: **repeat** $k = 1, \ldots,$
8: $\quad\boldsymbol{\beta}_k \leftarrow$ `BFGS`$(\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}, \boldsymbol{\beta}_{k-1}, \boldsymbol{\theta}_{k-1}, \mathbf{q}, \lambda_1, \lambda_2)$
9: $\quad$Form $\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}$ using $\boldsymbol{\beta}_k$ and $\mathbf{t}$
10: $\quad\boldsymbol{\theta}_k \leftarrow$ `MultiTaskElasticNet`$(\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}, \mathbf{q}, \lambda_1, \lambda_2)$
11: $\quad$Calculate residual norm r_k, total cost f_k using (1)
12: $\quad$Scale penalties $\lambda_i \leftarrow \lambda_i r_k^2 / r_{k-1}^2$, $i = 1, 2$
13: $\quad\boldsymbol{\theta}_k, \boldsymbol{\beta}_k \leftarrow$ `Prune`$(\boldsymbol{\theta}_k, \boldsymbol{\beta}_k)$
14: $\quad$Form $\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}$ using $\boldsymbol{\beta}_k$ and $\mathbf{t}$
15: **until** $\|f_k - f_{k-1}\| < \epsilon$

For a good performance of the algorithm, i.e., obtaining low residuals with a sparse set and low accelerations, choosing the regularizer weights λ_1 and λ_2 suitably is crucial. These parameters can be set using cross-validation either before Algorithm 1 or together with the initial regression (line 4). The regularizers should be scaled down accordingly with the decreasing residual norms (see line 12), otherwise the algorithm can converge to the empty set for the parameters $\boldsymbol{\theta}$.

The optimization problem, depending on the parameterization and the features used, can be highly nonconvex, possibly with many local minima. The number of local minima, fortunately, does not seem to pose a problem in terms of residual norm. As long as the initial representation is sufficiently (over) parameterized, most solutions fit well to the demonstration data. For more sparse representations, however, one may choose to restart the training procedure a few times from perturbed initial conditions, especially for the RBF parameters $\boldsymbol{\beta}$. See the Experiments section for more discussion on the implementation details.

The computational complexity of the algorithm overall is dominated by the complexity of the multi-task Elastic Net step (line 10), where the coordinate descent algorithm is used to solve a Lasso problem (after a transformation in constant time $\mathcal{O}(1)$). The time-complexity of the LARS algorithm to solve Lasso problems is known to be $\mathcal{O}(Np^2)$ [18], but coordinate-descent converges often faster, in our experience. One step of Quasi-Newton methods has time-complexity $\mathcal{O}(p^2)$ (plus the cost for function and gradient evaluations [20]), coming from the matrix multiplication operations. Quasi-Newton optimization may, depending on the initialization, require many of these steps, in our case we limit it to 1000 steps for each iteration of $LSDP$.

B. Coupling the Parameters Across Dimensions

The algorithm $LSDP$ discussed in the previous subsection uses the multi-task Elastic Net to enforce the same basis functions for each degree of freedom (along the columns of $\mathbf{q}(t)$ and the parameter matrix $\boldsymbol{\theta}$), while the parameter vectors corresponding to each joint movement are different and optimized independently: the regression parameters are decoupled across the degrees of freedom (DoF) of the robot. In particular, the number of regression parameters grow linearly with the robot DoF, which may be undesirable for applying policy search RL approaches to high dimensional robotic systems especially.

Furthermore, the algorithm has to be applied for each demonstration separately, i.e., there is no *coupling* or information shared between the demonstrations. In order to enforce rather the features to be shared *across demonstrations* rather than the robot DoFs, we discuss here a variant of the algorithm $LSDP$, which we call coupled $LSDP$, or $cLSDP$ for short.

The algorithm $cLSDP$, shown in Algorithm 2, requires only a few changes compared to Algorithm 1. The data is centered for each demonstration to obtain the intercepts $\mathbf{Q}_0$. The algorithm then stacks (lines $1-3$) the dependent regression variables $\mathbf{q}_i$ and the RBF parameters $\boldsymbol{\beta}_i$ vertically for each degree of freedom $i = 1, \ldots, n$ to form the matrices $\mathbf{Q} \in \mathbb{R}^{Nn \times d}$ and $\boldsymbol{\Psi} \in \mathbb{R}^{Nn \times p}$. The second time derivative of the data matrix, $\ddot{\boldsymbol{\Psi}}$, is stacked as well to form the regression model as in (1).

As opposed to $LSDP$, in this procedure there are n times the number of RBF parameters $\boldsymbol{\beta}$ to be optimized (line 8), as the features are adapted independently for each DoF. The regression parameters $\boldsymbol{\theta}$, on the other hand, are coupled across the DoFs, and their cardinality is reduced by n times. The nonlinear optimization computational complexity in this case dominates that of the multi-task Elastic Net and the net result is roughly a n times increase in the computation time between each iteration of $cLSDP$.

Note that the parameters for each demonstration are estimated together, i.e., the columns of the $\boldsymbol{\theta}$ matrix correspond to the regression parameters for different demonstrations. One way to generalize the learned movement primitives to different task conditions (such as varying initial joint states) would be to interpolate between these regression parameters. A policy could then be effectively created, whose generalization would be limited by the number and the quality (e.g. variety, success rate) of the demonstrations.

C. Ranking the Demonstration Parameters

The regression parameters estimated with $cLSDP$ can also be ranked in terms of statistical significance, i.e., correlation. The Elastic Net *regularization path* of the *LARS* algorithm [18] traces the evolution of the parameters as the l_1-penalty weight λ_1 of equation (1) increases. An example regularization path for twenty selected regression parameters $\boldsymbol{\theta} \in \mathbb{R}^{20}$ are plotted in Figure 2. Initially when the regularization is low ($\lambda_1 \approx 0$) on the right side of the Figure, the coefficients are close to their (nonzero) values in ordinary Least Squares. As the regularization term increases, some of these terms drop out, i.e., the coefficients become zero as the path is traced towards the left-hand side of the Figure. The corresponding features can then be eliminated from the regression model, leading not only to a sparse, but also a ranked set of features.

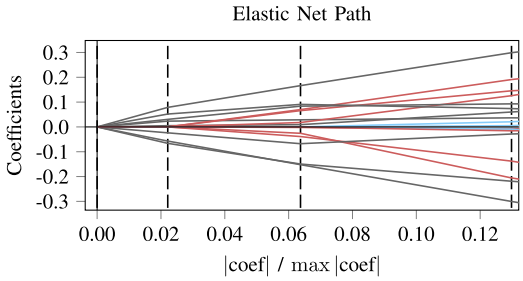

Fig. 2. An example Elastic Net path with twenty selected parameters is shown after training Algorithm 2, $cLSDP$, with five demonstrations. This *regularization path* can be generated in the final step of the algorithm. As the l_1-penalty term λ_1 of the regression problem (1) is reduced, the coefficients converge to their (maximal) ordinary least squares values at the right hand side of the plot (not shown). Each dashed line signals an entry of a parameter, and the slope of the coefficients are updated accordingly. The algorithm *LARS* [18] can be used to generate these piece-wise linear regularization paths. One possible way to use this path is to rank the sparse parameters of the learned movement primitives in terms of statistical importance. For example, in the shown plot, the parameters corresponding to the red lines would be ranked after the other parameters appearing before (black lines). The parameter paths, whose coefficients become nonzero close to each other, are drawn with the same color.

In the proposed method $cLSDP$, the *LARS* algorithm instead of coordinate descent can be used in the final Elastic Net computation step (line 11 of Algorithm 2) to generate the full regularization path. The addition of the selected movement primitive parameters can then be traced. An example path for twenty parameters selected by the Algorithm is plotted in Figure 2 against their normalized l_1-norm. These parameters can be ranked according to their evolution, i.e., the coefficients that early on during the path become nonzero are likely to signal more causally effective components of the motion. For example, in the shown plot, the parameters corresponding to the red lines would be ranked after some of the parameters appearing before (black lines). More prominent components of the motion can be identified this way. These movement components could be adapted earlier with RL strategies, reducing the curse of dimensionality in high dimensional robot learning problems.

IV. Experiments

In this section, we conduct experiments to learn a sparse set of movement primitive parameters using the proposed approaches (see Algorithms 1 and 2). The two algorithms are also compared against two competing movement primitive learning methods (DMPs and l_2-regularized regression). Finally we present real robot experiments on our table tennis platform where we show that the learned sparse movements nevertheless look similar to the shown demonstrations in style. They can also be implemented safely on the robot.

A. Learning From Demonstrations

The algorithms $LSDP$ and its coupled variant $cLSDP$, discussed in Section III, are applied here on the demonstrated Barrett WAM serve movements, see Figure 1. From a continuous stream of joint values, recorded at 500 Hz during a kinesthetic teach-in session, a predetermined number of d movements are selected by detecting the maximum d velocities in joint space and windowing around these points for a fixed duration of one second. We implement the preprocessing as well as the Algorithms in Python, using the *scikit-learn* toolbox for the multi-task Elastic Net and the *scipy* toolbox for the nonlinear optimization (BFGS, see lines 8 and 9 in the Algorithms, respectively).

TABLE I
Comparison of Different Learning From Demonstrations Approaches, Averaged Over Five Different Serve Demonstrations

	No. par. ($\|\boldsymbol{\theta}\|_0$)	**Acc. norm**	**Res. norm**
LSDP	$(16.8 \pm 3.25) \times 7$	59.04 ± 7.0	0.59 ± 0.11
cLSDP	37	55.98 ± 11.78	0.73 ± 0.09
DMPs	11×7	621.73 ± 57.45	0.92 ± 0.06
l2-reg. regr.	11×7	215.45 ± 35.25	2.12 ± 0.47

Algorithm 2: Learning Coupled Sparse Parameters with Regression ($cLSDP$) across Multiple Demonstrations.

Require $\mathbf{q}_{ij}$, $\mathbf{t}$, $\boldsymbol{\mu}_i$, $\boldsymbol{\sigma}_i^2$, λ_1, λ_2, $\epsilon > 0$
1: Stack $\mathbf{q}_{ij}$ to form $\mathbf{Q}$, $i \in [1, n], j \in [1, d]$
2: Center the data, $\mathbf{Q}_0, \mathbf{Q} \leftarrow$ `CenterStacked`$(\mathbf{Q})$
3: Stack $\boldsymbol{\beta}_0 = [\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_n, \boldsymbol{\sigma}_1^2, \ldots, \boldsymbol{\sigma}_n^2]$
4: Stack $\boldsymbol{\Psi}$, $\ddot{\boldsymbol{\Psi}}$ using $\boldsymbol{\beta}_0$ and $\mathbf{t}$ across DoFs
5: $\boldsymbol{\theta}_0 \leftarrow$ `MultiTaskElasticNet`$(\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}, \mathbf{Q}, \lambda_1, \lambda_2)$
6: $\boldsymbol{\theta}_0, \boldsymbol{\beta}_0 \leftarrow$ `PruneStacked`$(\boldsymbol{\theta}_0, \boldsymbol{\beta}_0)$
7: Stack $\boldsymbol{\Psi}$, $\ddot{\boldsymbol{\Psi}}$ using $\boldsymbol{\beta}$ and $\mathbf{t}$ across DoFs
8: **repeat** $k = 1, \ldots,$
9: $\quad \boldsymbol{\beta}_k \leftarrow$ `BFGS`$(\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}, \boldsymbol{\beta}_{k-1}, \boldsymbol{\theta}_{k-1}, \mathbf{Q}, \lambda_1, \lambda_2)$
10: $\quad$ Stack $\boldsymbol{\Psi}$, $\ddot{\boldsymbol{\Psi}}$ using $\boldsymbol{\beta}_k$ and $\mathbf{t}$ across DoFs
11: $\quad \boldsymbol{\theta}_k \leftarrow$ `MultiTaskElasticNet`$(\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}, \mathbf{Q}, \lambda_1, \lambda_2)$
12: $\quad$ Calculate residual norm r_k, total cost f_k using (1)
13: $\quad$ Scale penalties $\lambda_i \leftarrow \lambda_i r_k^2 / r_{k-1}^2$, $i = 1, 2$
14: $\quad \boldsymbol{\theta}_k, \boldsymbol{\beta}_k \leftarrow$ `PruneStacked`$(\boldsymbol{\theta}_k, \boldsymbol{\beta}_k)$
15: $\quad$ Stack $\boldsymbol{\Psi}$, $\ddot{\boldsymbol{\Psi}}$ using $\boldsymbol{\beta}_k$ and $\mathbf{t}$ across DoFs
16: **until** $\|f_k - f_{k-1}\| < \epsilon$

The preprocessed examples using the above procedure result in the joint matrix $\mathbf{q}(t) \in \mathbb{R}^{500 \times 7}$ for each example demonstration. For the algorithm $LSDP$, the initial RBF centers $\boldsymbol{\mu}_0 \in \mathbb{R}^{500}$ are placed at every time point and the RBF widths $\boldsymbol{\sigma}_0^2$ are set uniformly to 0.1. The algorithm stretches, prunes and expands the basis functions throughout the optimization to produce a very sparse, nonuniform set of basis functions shared across the seven degrees of freedom (DoF). The columns of the regression parameter matrix $\boldsymbol{\theta}$, on the other hand, are separate for each DoF.

The Algorithm $cLSDP$, on the other hand, optimizes n times more RBF parameters, i.e., $\boldsymbol{\mu} \in \mathbb{R}^{3500}$ and $\boldsymbol{\sigma}^2 \in \mathbb{R}^{3500}$ for the Barrett WAM with $n = 7$. During the optimization, all of the recorded data from d demonstrations are used together, and the same set of basis function parameters $\boldsymbol{\beta} = [\boldsymbol{\mu}^{\mathrm{T}}, (\boldsymbol{\sigma}^2)^{\mathrm{T}}]^{\mathrm{T}}$ are learned across multiple demonstrations. The learned parameters $\boldsymbol{\mu}, \boldsymbol{\sigma}^2, \boldsymbol{\theta}$, along with the intercepts, are saved after the optimizations to a json file, to be loaded later by the real-time robot controller in $C{+}{+}$ during the online experiments.

Table I summarizes the results of learning movement primitives from five different demonstrations. The three columns used

to compare the different approaches show on average the number of features selected (equivalently, the number of regression parameters with nonzero coefficients), the norm of the second derivatives of the trained movement primitives and the norm of the residuals, respectively. The five demonstration parameters are estimated together in $cLSDP$, whereas $LSDP$ is run separately for each demonstration to obtain the mean and the standard deviations reported in the table. Note that the number of parameters in total used by $cLSDP$ (37) is much lower than the on-average 16.8 parameters used by $LSDP$ for each robot DoF. The residual is slightly higher, this is a result of the parameters being shared across the dimensions. In particular, we have observed that $cLSDP$ does not fit the last three joints, corresponding to the Barrett WAM wrist, as tightly as $LSDP$. This could be because the motion of the wrist is highly varying across the movements and the coupling of the features induced by the algorithm across demonstrations brings these movements closer.

The two proposed algorithms are compared against two baselines in Table I. The first baseline is the Dynamic Movement Primitives (DMPs) with a fixed number of basis functions. DMPs learn the parameters of an attractor dynamics, i.e., a set of differential equations that converge to a suitably chosen goal state [12]. A standard regression is performed on the estimated attractor dynamics accelerations. The second baseline is l_2-penalized standard regression, with the penalty on the accelerations. During the experiments we used a total of ten basis functions both for the DMPs and for the l_2-penalized regression. The basis functions are spread uniformly, as discussed before for the proposed algorithms, around the one second long (preprocessed) demonstrations.

DMPs, as a result of the dynamic constraint of reaching a desired goal position, can incur very high initial accelerations in joint space. Even if the hyperparameters are optimized accordingly to prevent such high accelerations, slight modifications of initial joint positions can again give rise to high accelerations. The suggestion proposed in [21] to modify the accelerations with the phase can reduce the initial accelerations, but then we have found that the convergence to the goal positions can suffer drastically. The fixed basis function regression does not have this problem, but as in DMPs, optimizes a fixed number of parameters. As shown in Table I, the number of parameters to fit the demonstrations well is, for both compared methods, on average double the number optimized by $cLSDP$.

See Figure 3 for two example regression results. The demonstrated movements are shown in blue and the regression results are shown in orange. The first three rows, q_1 through q_3, correspond to the shoulder movement in joint space. The fourth row q_4 shows the movement of the elbow. Finally, the last three rows (q_5 through q_7) show the wrist movements in joint space. Although the demonstrated movements are quite different, the training with the sparse set of features can still capture them well.

Three example demonstrations are plotted in task space in Figure 4 along with the recorded ball positions, detected and triangulated from two cameras opposite to the robot. The initial positions of the racket center and the ball in the egg-holder are marked as 0 in red and blue, respectively. The egg-holder is at a distance of roughly 14 cm to the racket center. During the movement the ball is hit by the human demonstrator moving the robot arm, and as the demonstrator slows down the motion to a halt, the ball is seen flying towards the table.

B. Robot Experiments

Finally, we conduct experiments in our real robot table tennis platform, see Figure 1. Our table tennis playing robot is a seven degree of freedom Barrett WAM arm that is capable of reaching high accelerations and velocities. However it is cable-driven and high accelerations can cause the cables to break easily. A standard size racket is attached to the end-effector via a metal bar. The racket has a radius of roughly $r_R = 7.6$ cm. The table and the table tennis balls are standard sized, balls have a radius of 2 cm, and the table geometry is roughly $276 \times 152 \times 76$ cm. Throughout the experiments, the Barrett WAM is placed at a distance of about one meter to the end of the table and its base is located 95 cm above the table. This makes it difficult (but not impossible) for the robot to hit the table. An egg-holder holds the table tennis ball initially, wrapped around the metal bar connecting the end-effector and the racket, see Figure 1.

A successful serve in our robot platform is shown in Figure 5. The ball is initially placed on top of the egg-holder (approximately 14 cm away from the racket center along the racket plane). The movements captured by the algorithm $cLSDP$ are then executed on the robot. During the movements, as a result of the robot's accelerating motion, the ball takes off from the robot arm. The ball is then hit by the robot towards the table. The arm finally decelerates towards a resting posture as the ball lands on the robot court, passes the net, and lands again on the opposite side. We notice that the initial accelerating motion and the final wrist movement are critical for a good serve. Without the initial accelerations, the ball has no chance to take-off, and without the final wrist movement (e.g., a quick rotation towards the ball) the ball is not hit well towards the table.

A video showing some demonstrated movements, as well as several actual rollouts on the Barrett WAM is available online: https://youtu.be/vj6jfX_MQmQ. Note that orchestrating the right movement during the demonstrations can be quite difficult, as moving the shoulder and the elbow joints can feel very awkward depending on the posture. When a good initial posture (both for the robot and the demonstrator) is found, the resulting demonstrations have a higher quality in general. These higher quality demonstrations also have a higher chance of being executed successfully.

Comparing our approach to the DMPs, we notice that the DMPs immediately start the movement with very large accelerations, these can be dangerous for the robot and the low-level Barrett WAM controllers do not support 80% of the movements. DMPs are good at capturing movements that converge to goal positions (with zero or low velocities), however they are less accurate in capturing the style (e.g., initial movement, final wrist turns) of dynamics movements such as table tennis serves, without manual tuning (the number of basis functions, locations

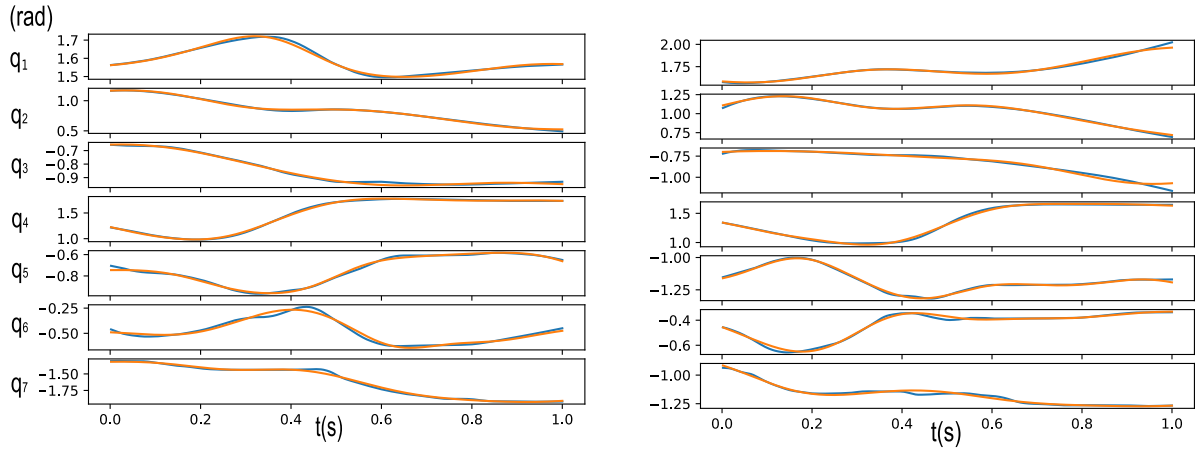

Fig. 3. Two movement primitives learned by the proposed algorithm $cLSDP$, are plotted in joint space against the recorded demonstrations. The table tennis serve movements, shown in blue, after preprocessing and segmenting the recorded time series are one second long each. The first three rows, q_1 through q_3, correspond to the shoulder movement in joint space. The fourth row q_4 shows the movement of the elbow. Finally, the last three rows (q_5 through q_7) show the wrist movements in joint space. The trained movement primitives, shown in orange, couple the sparse regression parameters across the degrees of freedom of the robot.

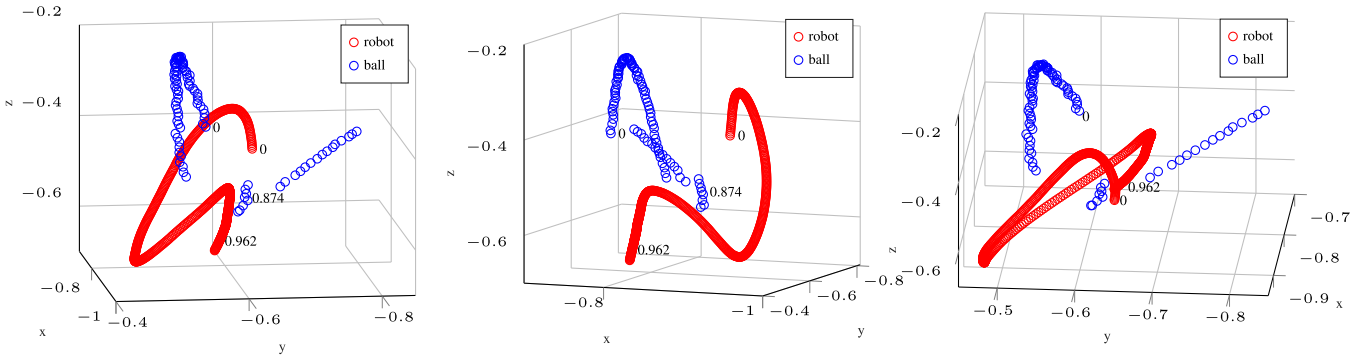

Fig. 4. Three example demonstrations in task space. The initial position of the racket center and the ball in the egg-holder are marked as 0 in red and blue circles, respectively. The egg-holder is located approximately 14 cm away from the racket centre. Before the racket stops moving, the ball is already hit, flying towards the table.

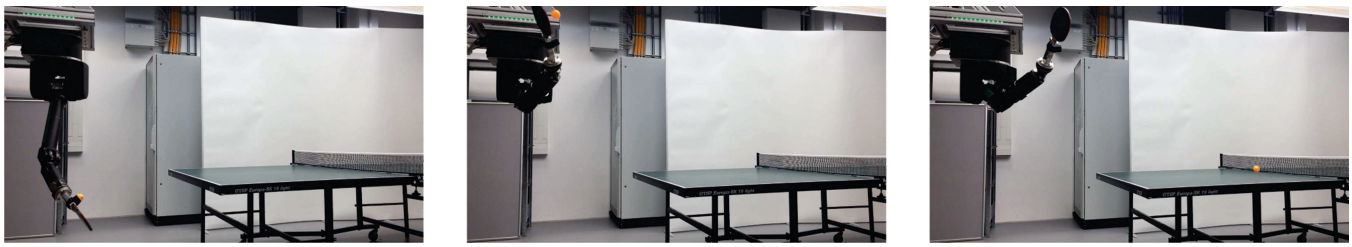

Fig. 5. A successful rollout during real robot experiments. The ball is initially on top of the egg-holder and during the movement, as a result of the acceleration of the arm, it takes-off from the robot, to be later hit by the racket towards the table. The arm then decelerates towards a safe resting posture.

and widths of the basis functions, etc.) for each task. We have seen that $cLSDP$[2] on the contrary, can capture the style of the movement, as shown in Figure 3 for some of the movements, with a sparse set of basis functions. The generalization capacity of these selected basis functions hinges on the quality and the number of the shown demonstrations.[3]

[2]Note that executing the Algorithm $LSDP$, trained on each demonstation independently, shows a very similar performance, to that of Algorithm $cLSDP$ at the moment. However we expect improvements on the learning performance, if Reinforcement Learning is applied on top of the more sparse set of $cLSDP$ parameters.

[3]If the number and the quality of the demonstrations is not enough, then the selected features and their ranking (using the regularization path) may be spurious, i.e., without any meaningful physical relevance. Increasing the number and the quality (e.g., increased variety of movements, higher success rates) of the movements could remedy such a limitation.

V. Conclusion

In this letter we presented a new learning from demonstrations (LfD) approach to represent and learn table tennis serve movements. The proposed algorithms $LSDP$ and $cLSDP$ learn sparse parameters of the radial basis functions (RBF) from single and multiple demonstrations, respectively. The algorithms employ iterative optimization, alternating between a weighted multi-task Elastic Net regression step that learns sparse parameters given the features and a nonlinear optimization step that adapts the features (more specifically, the widths and centers of the RBFs corresponding to the nonzero regression parameters). The algorithm $cLSDP$, unlike $LSDP$, learns (sparse) parameters that are independent of the robot DoF. This desirable property is achieved by having different basis functions that are adapted across each DoF separately. The multi-task Elastic Net, in this case, forces the joint-dependent features to be shared across multiple demonstrations.

The cost function chosen for the optimization includes the residual of the fit, as well as l_2-regularization terms on the accelerations and l_1-regularization on the regression coefficients. We compared the performance of the proposed algorithms with Dynamic Movement Primitives (DMPs) and the standard l_2-regularized regression, and we evaluated the performance of each on the different components of the chosen cost function (see Table I). Finally, we discussed the performance of the actual rollouts, using our framework, on the real robot table tennis setup. One can see in the video available online that the style of the movements are preserved while maintaining low accelerations throughout the motion, which is important for the safety of the robot.

The sparsity of the parameters, as well as their decoupling from the robot DoF, is a desirable property for policy-search RL approaches that could adapt the regression parameters online based on a suitable reward function. We have presented a way to rank these policy parameters, in the last subsection of Section III, based on how well the parameters explain the (multiple) demonstration recordings. We think that this is a promising research direction to combat the curse of dimensionality in high dimensional robot learning tasks, and we will focus on it more in future experiments.

References

[1] R. L. Anderson, *A Robot Ping-Pong Player: Experiment in Real-Time Intelligent Control*. Cambridge, MA, USA: MIT Press, 1988.

[2] M. Ramanantsoa and A. Durey, "Towards a stroke construction model," *Int. J. Table Tennis Sci.*, vol. 2, pp. 97–114, 1994.

[3] K. Muelling, J. Kober, and J. Peters, "A biomimetic approach to robot table tennis," *Adaptive Behav. J.*, vol. 19, no. 5, pp. 359–376, 2011. [Online]. Available: http://www.ias.informatik.tu-darmstadt.de/publications/Muelling_ABJ2011.pdf

[4] O. Koc, G. Maeda, G. Neumann, and J. Peters, "Optimizing robot striking movement primitives with iterative learning control," in *Proc. 15th IEEE-RAS Int. Conf. Humanoid Robots*, 2015, pp. 80–87.

[5] K. Muelling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *Int. J. Robot. Res.*, vol. 3, pp. 263–279, 2013. [Online]. Available: http://www.ias.informatik.tu-darmstadt.de/uploads/Publications/Muelling _IJRR_2013.pdf

[6] Y. Huang, D. Bchler, O. Ko, B. Schlkopf, and J. Peters, "Jointly learning trajectory generation and hitting point prediction in robot table tennis," in *Proc. IEEE-RAS 16th Int. Conf. Humanoid Robots*, Nov. 2016, pp. 650–655.

[7] O. Koc, G. Maeda, and J. Peters, "A new trajectory generation framework in robotic table tennis," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 3750–3756.

[8] O. Koc, G. Maeda, and J. Peters, "Online optimal trajectory generation for robot table tennis," *Robot. Auton. Syst.*, vol. 105, pp. 121–137, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889017306164

[9] J. Nonomura, A. Nakashima, and Y. Hayakawa, "Analysis of effects of rebounds and aerodynamics for trajectory of table tennis ball," in *Proc. SICE Annu. Conf.*, Aug. 2010, pp. 1567–1572.

[10] G. J. Maeda, G. Neumann, M. Ewerton, R. Lioutikov, O. Kroemer, and J. Peters, "Probabilistic movement primitives for coordination of multiple human–robot collaborative tasks," *Auton. Robots*, vol. 41, no. 3, pp. 593–612, 2017.

[11] J. Kober and J. Peters, "Policy search for motor primitives in robotics," in *Proc. Advances Neural Inf. Process. Syst.*, Cambridge, MA, USA, 2009, pp. 849–856. [Online]. Available: http://www-clmc.usc.edu/publications/K/kober_NIPS2008.pdf

[12] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Comput.*, vol. 25, no. 2, pp. 328–373, Feb. 2013. [Online]. Available: http://dx.doi.org/10.1162/NECO_a_00393

[13] S. M. Khansari-Zadeh and A. Billard, "Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions," *Robot. Auton. Syst.*, vol. 62, pp. 752–765, 2014.

[14] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Proc. Advances Neural Inf. Process. Syst.*, 2013, pp. 2616–2624. [Online]. Available: http://papers.nips.cc/paper/5177-probabilistic-movement-primitives.pdf

[15] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 943–957, Oct. 2011.

[16] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning* (Series Springer Series in Statistics). New York, NY, USA: Springer, 2001.

[17] G. Obozinski and B. Taskar, "Multi-task feature selection," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006.

[18] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Ann. Statist.*, vol. 32, pp. 407–499, 2004.

[19] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *J. Roy. Statistical Soc., Ser. B*, vol. 67, pp. 301–320, 2005.

[20] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York, NY, USA: Springer-Verlag, 1999.

[21] J. Kober, K. Muelling, O. Kroemer, C. Lampert, B. Schoelkopf, and J. Peters, "Movement templates for learning of hitting and batting," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2010, pp. 853–858. [Online]. Available: http://www.ias.informatik.tu-darmstadt.de/publications/ICRA2010-Kober_6 231[1].pdf