From Least Squares Regression to High-dimensional Motion Primitives

Part II

Sylvain Calinon
Senior Researcher
Idiap Research Institute, Martigny, Switzerland
Lecturer
EPFL, Lausanne, Switzerland
External Collaborator
IIT, Genoa, Italy
Superposition of movement primitives vs Fusion of movement primitives
• Combination as fusion problem

• Application I: Ridge regression

• Application II: Model predictive control

• Application III: Task-parameterized models

• Examples of robot applications
Combination of primitives as a fusion problem

Superposition

\[ \hat{x} = \sum_{k=1}^{K} w_k \phi_k \]

Fusion

\[ \hat{x} = \left( \sum_{k=1}^{K} W_k \right)^{-1} \sum_{k=1}^{K} W_k \phi_k \]

\[ = \arg \min_x \sum_{k=1}^{K} \left\lVert \phi_k - x \right\rVert^2_{W_k} \]

\[ = \arg \min_x \sum_{k=1}^{K} (\phi_k - x)^\top W_k (\phi_k - x) \]

Choosing scalar weights or full weight matrices is not a detail...
Motivating example:
A probabilistic view on segment crossing!

\[ \mu_i \] center of the Gaussian
\[ \Sigma_i \] covariance matrix
\[ W_i \] precision matrix
\[ (W_i = \Sigma_i^{-1}) \]

\[ \hat{x} = \arg\min_x \left\| \mu_1 - x \right\|_{W_1}^2 + \left\| \mu_2 - x \right\|_{W_2}^2 \]
\[ = (W_1 + W_2)^{-1} (W_1 \mu_1 + W_2 \mu_2) \]
\[ = (\Sigma_1^{-1} + \Sigma_2^{-1})^{-1} (\Sigma_1^{-1} \mu_1 + \Sigma_2^{-1} \mu_2) \]

\[ \mathcal{N}(\mu_1, \Sigma_1) \]
\[ \mathcal{N}(\mu_2, \Sigma_2) \]
\[ \mathcal{N}(\mu_1, \Sigma_1) \rightarrow \text{Product of Gaussians} \]

\( \hat{x} = \frac{1}{2} \mu_1 + \frac{1}{2} \mu_2 \)

(superposition)

(fusion)
Motivating example: Fusion of IK and joint angle controllers

Goal 1: Tracking this point

Goal 2: Keeping first joint upright

Superposition

Fusion
Combination of primitives as a fusion problem

\[ N(\mu, \Sigma) \propto N(\mu^{(1)}, \Sigma^{(1)}) \cdot N(\mu^{(2)}, \Sigma^{(2)}) \]

- Null space projection (hierarchy constraints)
- Scalar superposition

The full weight matrices approach covers both scalar weights (with isotropic diagonal matrix) and null space projection operations!
Bayesian linear regression
and its connection to ridge regression
Ridge regression

Input data: \( \mathbf{X} \in \mathbb{R}^{N \times D_T} \)
Output data: \( \mathbf{Y} \in \mathbb{R}^{N \times D_O} \)
Goal: estimating \( \mathbf{\beta} \in \mathbb{R}^{D_T \times D_O} \) to have \( \mathbf{Y} = \mathbf{X} \mathbf{\beta} \)

Ridge regression: 
\[
\hat{\mathbf{\beta}} = \arg \min_{\mathbf{\beta}} \| \mathbf{Y} - \mathbf{X} \mathbf{\beta} \|^2 + \| \lambda \mathbf{\beta} \|^2 \\
= \arg \min_{\mathbf{\beta}} (\mathbf{Y} - \mathbf{X} \mathbf{\beta})^\top (\mathbf{Y} - \mathbf{X} \mathbf{\beta}) + \lambda^2 \mathbf{\beta}^\top \mathbf{\beta} \\
= (\mathbf{X}^\top \mathbf{X} + \lambda^2 \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{Y}
\]
Least squares with a regularization term corresponds to a maximum a posteriori (MAP) estimate with a Gaussian likelihood and a Gaussian prior.
Bayesian linear regression

Product of Gaussians:

\[ \hat{x} = \left( \Sigma_1^{-1} + \Sigma_2^{-1} \right)^{-1} \left( \Sigma_1^{-1} \mu_1 + \Sigma_2^{-1} \mu_2 \right) \]

Proposed Bayesian view of ridge regression:

\[ \mathcal{N} \left( \hat{\beta}, \hat{\Sigma}^\beta \right) \propto \mathcal{N} \left( X^\dagger Y, (X^\top X)^{-1} \right) \mathcal{N} \left( 0, \lambda^{-2} I \right) \]

Verification:

\[ \hat{\beta} = \left( X^\top X + \lambda^2 I \right)^{-1} X^\top X X^\dagger Y \]

\[ = \left( X^\top X + \lambda^2 I \right)^{-1} X^\top Y \]

\[ \text{ridge regression!} \]
Model predictive control (MPC) & Linear quadratic tracking (LQT)
**Linear quadratic tracking (LQT)**

\[
\min_u \sum_{t=1}^{T} \left( \frac{1}{Q_t} \left\| \mu_t - x_t \right\|^2 + \frac{1}{R_t} \left\| u_t \right\|^2 \right)
\]

s.t. \[ x_{t+1} = Ax_t + Bu_t \]  

Use low control commands!

Track path!

---

**Model predictive control (MPC):**

- \( x_t \) state variable (position+velocity)
- \( \mu_t \) desired state
- \( u_t \) control command (acceleration)
- \( Q_t \) precision matrix
- \( R_t \) control weight matrix

System dynamics

\[ Q_T = \Sigma_T^{-1} \]
How to solve this objective function?

\[
\min_u \sum_{t=1}^{T} \left( \frac{1}{2} \| \mu_t - x_t \|^2_{Q_t} + \frac{1}{2} \| u_t \|^2_{R_t} \right)
\]

s.t. \( x_{t+1} = Ax_t + Bu_t \) System dynamics

Pontryagin’s maximum principle → Riccati equation
(\textit{the Physicist perspective})

Dynamic programming
(\textit{the Computer Scientist perspective})

Linear algebra
(\textit{the Algebraist perspective})
Let's first re-organize the objective function...

\[
c = \sum_{t=1}^{T} \left( (\mu_t - x_t)^\top Q_t (\mu_t - x_t) + u_t^\top R_t \, u_t \right)
\]

\[
= (\mu - x)^\top Q (\mu - x) + u^\top R u
\]

\[
Q = \begin{bmatrix}
Q_1 & 0 & \cdots & 0 \\
0 & Q_2 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & Q_T
\end{bmatrix}
\]

\[
R = \begin{bmatrix}
R_1 & 0 & \cdots & 0 \\
0 & R_2 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & R_T
\end{bmatrix}
\]

\[
\mu = \begin{bmatrix}
\mu_1 \\
\mu_2 \\
\vdots \\
\mu_T
\end{bmatrix}
\quad x = \begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_T
\end{bmatrix}
\quad u = \begin{bmatrix}
u_1 \\
u_2 \\
\vdots \\
u_T
\end{bmatrix}
\]
Let’s then re-organize the constraint...

\[ x_{t+1} = A x_t + B u_t \]

\[ x_2 = A x_1 + B u_1 \]
\[ x_3 = A x_2 + B u_2 = A(A x_1 + B u_1) + B u_2 \]
\[ \vdots \]
\[ x_T = A^{T-1} x_1 + A^{T-2} B u_1 + A^{T-3} B u_2 + \cdots + B_{T-1} u_{T-1} \]

\[
\begin{bmatrix}
  x_1 \\
  x_2 \\
  x_3 \\
  \vdots \\
  x_T \\
\end{bmatrix} = 
\begin{bmatrix}
  I \\
  A \\
  A^2 \\
  \vdots \\
  A^{T-1} \\
\end{bmatrix}
\begin{bmatrix}
  x_1 \\
\end{bmatrix} + 
\begin{bmatrix}
  0 & 0 & \cdots & 0 & 0 \\
  B & 0 & \cdots & 0 & 0 \\
  AB & B & \cdots & 0 & 0 \\
  \vdots & \vdots & \ddots & \vdots & \vdots \\
  A^{T-2} B & A^{T-3} B & \cdots & B & 0 \\
\end{bmatrix}
\begin{bmatrix}
  u_1 \\
  u_2 \\
  \vdots \\
  \vdots \\
  u_T \\
\end{bmatrix}
\]

\[ x = S^x x_1 + S^u u \]
Linear quadratic tracking: Analytic solution

The constraint can then be put into the objective function:

\[ x = S^x x_1 + S^u u \]

\[ c = (\mu - x)^T Q (\mu - x) + u^T R u \]

\[ = (\mu - S^x x_1 - S^u u)^T Q (\mu - S^x x_1 - S^u u) + u^T R u \]

Solving for \( u \) results in the analytic solution:

\[ \hat{u} = (S^{ru^T} Q S^{ru} + R)^{-1} S^{ru^T} Q (\mu - S^x x_1) \]

\[ N \propto N \mathbb{N} \]
MPC/LQT as a product of Gaussians

\[ \min_u \sum_{t=1}^{T} \left\| \mu_t - x_t \right\|^2_{Q_t} + \left\| u_t \right\|^2_{R_t} \]

\[ \mathcal{N}(\hat{u}, \hat{\Sigma}_u) \propto \mathcal{N}(\mu_u, Q_u^{-1}) \mathcal{N}(0, R^{-1}) \]

\[ \mu_u \triangleq S_u^\dagger (\mu - S^x x_1) \]

\[ Q_u \triangleq S_u^\dagger Q S_u \]

\[ \xrightarrow{\text{Bayesian view on MPC with variables}} \text{spanning a given time horizon} \]
The probabilistic representation of MPC/LQT

\[
\hat{u} = (S_u^\top Q S_u + R)^{-1} S_u^\top Q (\mu - S_x x_1)
\]
\[
\hat{\Sigma}_u = (S_u^\top Q S_u + R)^{-1}
\]
\[
\hat{x} = S_x x_1 + S_u \hat{u}
\]
\[
\hat{\Sigma}_x = S_u (S_u^\top Q S_u + R)^{-1} S_u^\top
\]

The distribution in control space can be projected back to the state space.

Passing through 3 keypoints with varying precision.
Model Predictive Control (MPC) combined with probabilistic representation of movement primitives

\[ c = (\mu - x)^\top Q (\mu - x) + u^\top Ru \]

\[
\mu = \begin{bmatrix}
\mu_1 \\ \mu_2 \\ \vdots \\ \mu_T
\end{bmatrix}
\]

\[
Q = \begin{bmatrix}
Q_1 & 0 & \cdots & 0 \\
0 & Q_2 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & Q_T
\end{bmatrix}
\]
HSMM provides a model of the state duration instead of relying on self-transition probabilities as in standard HMM.
Learning minimal intervention controllers

- Analytical solution to generate movements by following minimal intervention control principle

Stepwise reference path given by:

\[ \hat{x}_t = \mu_{s_t}, \quad Q_t = \Sigma_{s_t}^{-1} \]

\[ S_t \quad \{11111111222222222222333333333333 \} \]

Transition and state duration (HSMM)

- Center of the Gaussian \( \mu_i \)
- Covariance matrix \( \Sigma_i \)

\[ x_0 \quad \mathcal{N}(\mu_1, \Sigma_1) \quad \mathcal{N}(\mu_2, \Sigma_2) \quad \mathcal{N}(\mu_3, \Sigma_3) \]
Learning minimal intervention controllers

\[ s = \{1, 1, 1, 2, \ldots, 4\} \]

\[
\begin{bmatrix}
\Sigma_1^{-1} & 0 & 0 & 0 & \cdots & 0 \\
0 & \Sigma_1^{-1} & 0 & 0 & \cdots & 0 \\
0 & 0 & \Sigma_1^{-1} & 0 & \cdots & 0 \\
0 & 0 & 0 & \Sigma_2^{-1} & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & 0 & \cdots & \Sigma_K^{-1}
\end{bmatrix}
\begin{bmatrix}
\mu_1 \\
\mu_1 \\
\mu_1 \\
\mu_2 \\
\vdots \\
\mu_K
\end{bmatrix}
\]

\[
\hat{u} = (S_u^\top QS_u^u + R)^{-1} S_u^\top Q (\mu - S^x x_1)
\]

\[
\begin{bmatrix}
\hat{u}_1 \\
\hat{u}_2 \\
\hat{u}_3 \\
\vdots \\
\hat{u}_{T-1}
\end{bmatrix}
\begin{bmatrix}
0 & 0 & \cdots & 0 \\
B & 0 & \cdots & 0 \\
AB & B & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
A^{T-2}B & A^{T-3}B & \cdots & B
\end{bmatrix}
\begin{bmatrix}
I \\
A \\
A^2 \\
\vdots \\
A^{T-1}
\end{bmatrix}
\]
Learning controllers instead of trajectories
Task-parameterized movement models
Movements most often relate to objects, tools or body landmarks
Conditioning-based approach

Regression with a context variable $c$:
- Learning of $\mathcal{P}(c, x)$
- Retrieval with $\mathcal{P}(x|c)$

$\rightarrow$ Generic approach, but limited generalization capability
MPC/LQT in multiple coordinate systems

\[
\min_u \sum_{t=1}^{T} \sum_{j=1}^{P} \left( \| \mu_{t}^{(j)} - x_t \|_2^2 Q_{t}^{(j)} + \| u_t \|_2^2 R_t \right)
\]

s.t. \[ x_{t+1} = A x_t + B u_t \]

Track path in coordinate system \( j \)

Use low control commands

System dynamics

Two candidate coordinate systems \( (P=2) \)

New position and orientation of coordinate systems 1 and 2

Set of demonstrations

Reproduction in new situation
MPC/LQT in multiple coordinate systems

\[
\min_u \sum_{t=1}^{T} \sum_{j=1}^{P} \left( \left\| \mu^{(j)}_t - x_t \right\|^2_{Q^{(j)}_t} + \left\| u_t \right\|^2_{R_t} \right)
\]

subject to \( x_{t+1} = Ax_t + Bu_t \)

Track path in coordinate system j

Use low control commands

System dynamics
MPC/LQT in multiple coordinate systems

\[
\min_u \sum_{t=1}^{T} \sum_{j=1}^{P} \left( \| \mu_t^{(j)} - x_t \|^2_{Q_t^{(j)}} + \| u_t \|^2_{R_t} \right)
\]

Track path in coordinate system j

Use low control commands

...
MPC/LQT in multiple coordinate systems

\[
\min_{u} \sum_{t=1}^{T} \sum_{j=1}^{P} \left( \| \mu_t^{(j)} - x_t \|_{Q_t^{(j)}}^2 + \| u_t \|_{R_t}^2 \right)
\]

In many robotics problems, the parameters describing the task or situation can be interpreted as coordinate systems.
MPC/LQT in multiple coordinate systems

\[
\min_u \sum_{t=1}^{T} \sum_{j=1}^{P} \| \mu_t^{(j)} - x_t \|^2_{Q_t^{(j)}} + \| u_t \|^2_{R_t}
\]

- Learning of a controller (instead of learning a trajectory) that adapts to new situations while regulating the gains according to the precision and coordination required by the task.
MPC/LQT in multiple coordinate systems

\[
\min_u \sum_{t=1}^{T} \sum_{j=1}^{P} \left\| \mu_t^{(j)} - \mathbf{x}_t \right\|^2_{Q_t^{(j)}} + \left\| u_t \right\|^2_{R_t}
\]

➤ Retrieval of control commands in the form of trajectory distributions, facilitating exploration and adaptation (in either control or state space)
**Exploitation in other probabilistic models**

- TP model with raw trajectory distribution
- TP model with MPC
- TP model with GMR
- TP model with Trajectory-HMM
- TP model with ProMP

Reproductions in same situations

Reproductions in new situations

http://www.idiap.ch/software/pbdlib/
Robot application examples
**Application: Editing movements with variations**

User interface to edit and generate natural and dynamic motions by considering variation and coordination.

Compliant controller to retrieve safe and human-like motions.

---

[Berio, Calinon and Leymarie, IROS’2016]  [Berio, Calinon and Leymarie, MOCO’2017]

Daniel Berio  Frederic Fol Leymarie

"Baxter"
Application: Shared control

DexROV will introduce new levels of safety, effectiveness, reduce operational costs for ROV operations.

http://dexrov.eu
Application: Shared control

Teleoperator side

Robot side

only Gaussian ID is transmitted


DexROV http://dexrov.eu
Adaptation to different object shapes

Coordinate system as task parameter

[Calinon, Alizadeh and Caldwell, IROS’2013]
Learning & generalizing tasks prioritization

\[ \hat{q} = \begin{bmatrix} J_1^\dagger & N_1 J_2^\dagger \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} \]

Candidate hierarchy \( A_1 \)

\[ \hat{q} = \begin{bmatrix} N_2 J_1^\dagger & J_2^\dagger \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} \]

Candidate hierarchy \( A_2 \)

[Silvério, Calinon, Rozo and Caldwell (2018), Arxiv 1707.06791] [Calinon, ISRR’15]
Learning & generalizing tasks prioritization

\[
\hat{q} = \begin{bmatrix} J_1^\dagger & \lambda_1 J_2^\dagger \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix}
\]

Candidate hierarchy $A_1$

\[
\hat{q} = \begin{bmatrix} \lambda_2 J_1^\dagger & J_2^\dagger \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix}
\]

Candidate hierarchy $A_2$

[Silvério, Calinon, Rozo and Caldwell (2018), Arxiv 1707.06791] [Calinon, ISRR’15]
Learning & generalizing tasks prioritization

\[ \hat{q} = \begin{bmatrix} J_1^\dagger & N_1 J_2^\dagger \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} \]

Candidate hierarchy \( A_1 \)

\[ \hat{q} = \begin{bmatrix} N_2 J_1^\dagger & J_2^\dagger \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} \]

Candidate hierarchy \( A_2 \)

[Silvério, Calinon, Rozo and Caldwell (2018), Arxiv 1707.06791] [Calinon, ISRR’15]
Summary

Combination as fusion problem

• Application I: Ridge regression
• Application II: Model predictive control
• Application III: Task-parameterized models

Further extensions and open issues

• Non-Gaussian distributions (e.g., L1-norm instead of L2-norm)
• Multimodal distributions (e.g., controllers with options)
• Approximation with linearization and quadratization


Source codes

http://www.idiap.ch/software/pbdlib/

Matlab / GNU Octave

C++

Python
**PbDlib**

PbDlib is a collection of source codes for robot programming by demonstration (learning from demonstration). It includes a varied set of functionalities at the crossroad of statistical learning, dynamical systems, optimal control and differential geometry. It is available in the following languages:

- Matlab / GNU Octave
- C++
- Python

PbDlib can be used in applications requiring task adaptation, human-robot skill transfer, safe controllers based on minimal intervention principle, as well as for probabilistic motion analysis and synthesis in multiple coordinate systems.
Source codes (Matlab/Octave, C++ and Python):
http://www.idiap.ch/software/pbdlib/

Contact:
sylvain.calinon@idiap.ch
http://calinon.ch

Photo: Basilio Noris