Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

Institute for AI and Autonomous Systems

Learning from Demonstration

An Overview With a Focus on Trajectory Learning

Dr. Alex Mitrevski
Master of Autonomous Systems

Structure

- What is learning from demonstration?
- Learning from one demonstration: Dynamic motion primitives
- Learning from multiple demonstrations: Gaussian mixture models

Annual Review of Control, Robotics, and Autonomous Systems

Recent Advances in Robot Learning from Demonstration

Harish Ravichandar,[1,*] Athanasios S. Polydoros,[2,*] Sonia Chernova,[1,†] and Aude Billard[2,†]

Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors

In Special Collection: CogNet

Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, Stefan Schaal

Neural Computation (2013) 25 (2): 328–373.

IEEE TRANSACTIONS ON ROBOTICS, VOL. 27, NO. 5, OCTOBER 2011 943

Learning Stable Nonlinear Dynamical Systems With Gaussian Mixture Models

S. Mohammad Khansari-Zadeh and Aude Billard

Learning from Demonstration Preliminaries

What is Learning from Demonstration?

F. J. Abu-Dakka et al. "Solving peg-in-hole tasks by human demonstration and exception strategies," *Industrial Robot: An International Journal*, vol. 41, no. 6, pp. 575–584, 2014.

T. Zhang et al., "Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2018, pp. 5628–5635

- Learning from demonstration is a **technique based on which a robot acquires data for learning by observing a human demonstrator**

What is Learning from Demonstration?

F. J. Abu-Dakka et al. "Solving peg-in-hole tasks by human demonstration and exception strategies," *Industrial Robot: An International Journal*, vol. 41, no. 6, pp. 575–584, 2014.

T. Zhang et al., "Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2018, pp. 5628–5635

- Learning from demonstration is a **technique based on which a robot acquires data for learning by observing a human demonstrator**
- This can essentially be seen as **a robot programming technique without writing explicit programs** — the knowledge about the robot's behaviour comes from the demonstrations rather than through manually written code

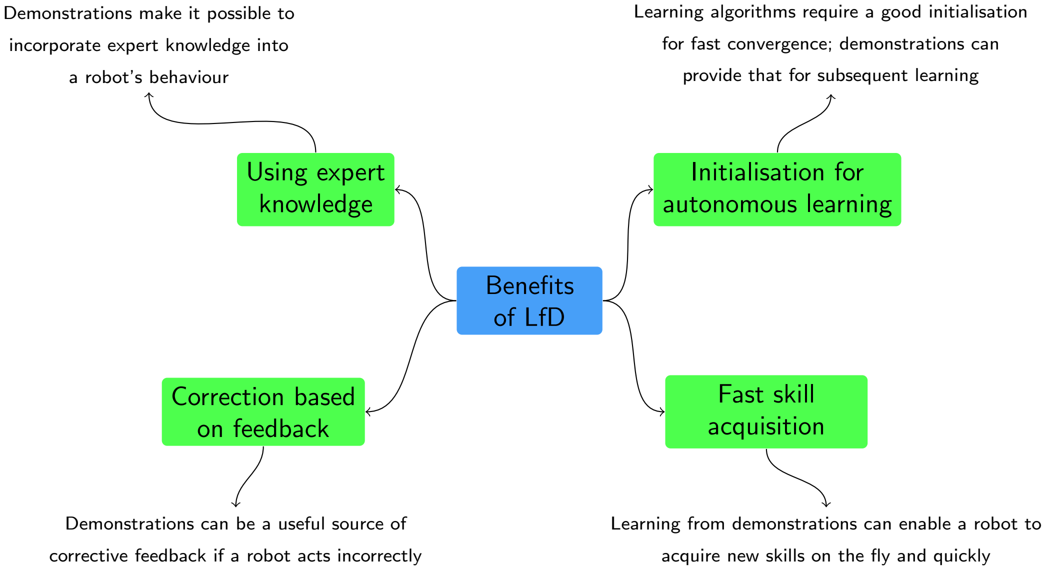
What is Learning from Demonstration?

F. J. Abu-Dakka et al. "Solving peg-in-hole tasks by human demonstration and exception strategies," *Industrial Robot: An International Journal*, vol. 41, no. 6, pp. 575–584, 2014.

T. Zhang et al., "Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2018, pp. 5628–5635

- Learning from demonstration is a **technique based on which a robot acquires data for learning by observing a human demonstrator**
- This can essentially be seen as **a robot programming technique without writing explicit programs** — the knowledge about the robot's behaviour comes from the demonstrations rather than through manually written code
- Demonstration-based learning can **enable end users to teach skills to a robot** without expert knowledge in robotics

Benefits of Learning from Demonstration



Demonstration Types

- When performing demonstration-based learning, an essential choice is the **modality used for performing and recording demonstrations**

Demonstration Types

- When performing demonstration-based learning, an essential choice is the **modality used for performing and recording demonstrations**
- There are three main types of demonstrations:
  - **Kinesthetic teaching**: A robot is physically moved by the demonstrator and the robot's internal sensors are used to record the demonstration

Demonstration Types

- When performing demonstration-based learning, an essential choice is the **modality used for performing and recording demonstrations**
- There are three main types of demonstrations:
  - **Kinesthetic teaching**: A robot is physically moved by the demonstrator and the robot's internal sensors are used to record the demonstration
  - **Teleoperation**: The demonstrator controls the robot remotely (e.g. with a joystick or through a virtual reality) and the demonstration is recorded using the robot's internal sensors

Demonstration Types

- When performing demonstration-based learning, an essential choice is the **modality used for performing and recording demonstrations**
- There are three main types of demonstrations:
  - **Kinesthetic teaching**: A robot is physically moved by the demonstrator and the robot's internal sensors are used to record the demonstration
  - **Teleoperation**: The demonstrator controls the robot remotely (e.g. with a joystick or through a virtual reality) and the demonstration is recorded using the robot's internal sensors
  - **Passive observation**: The demonstrator performs the task naturally and the robot observes the demonstration using its external sensors

Demonstration Types

- When performing demonstration-based learning, an essential choice is the **modality used for performing and recording demonstrations**
- There are three main types of demonstrations:
  - **Kinesthetic teaching**: A robot is physically moved by the demonstrator and the robot's internal sensors are used to record the demonstration
  - **Teleoperation**: The demonstrator controls the robot remotely (e.g. with a joystick or through a virtual reality) and the demonstration is recorded using the robot's internal sensors
  - **Passive observation**: The demonstrator performs the task naturally and the robot observes the demonstration using its external sensors

Correspondence Problem

- When a robot observes demonstrations from an external perspective, **there is a need to map the actions of the demonstrator to the robot's embodiment**

Correspondence Problem

- When a robot observes demonstrations from an external perspective, **there is a need to map the actions of the demonstrator to the robot's embodiment**
- **The mapping is mostly trivial if the embodiments are the same** (but there is an ambiguity regarding the mirroring of motions)

Correspondence Problem

- When a robot observes demonstrations from an external perspective, **there is a need to map the actions of the demonstrator to the robot's embodiment**
- **The mapping is mostly trivial if the embodiments are the same** (but there is an ambiguity regarding the mirroring of motions)
- If the embodiments differ, there is a **correspondence problem** — the robot needs to perform the actions of the demonstrator considering its own embodiment

Correspondence Problem

- When a robot observes demonstrations from an external perspective, **there is a need to map the actions of the demonstrator to the robot's embodiment**
- **The mapping is mostly trivial if the embodiments are the same** (but there is an ambiguity regarding the mirroring of motions)
- If the embodiments differ, there is a **correspondence problem** — the robot needs to perform the actions of the demonstrator considering its own embodiment
- Depending on the observed actions, **there may be many possible mappings to solve the correspondence problem, or no mapping at all**

Correspondence Problem

- When a robot observes demonstrations from an external perspective, **there is a need to map the actions of the demonstrator to the robot's embodiment**
- **The mapping is mostly trivial if the embodiments are the same** (but there is an ambiguity regarding the mirroring of motions)
- If the embodiments differ, there is a **correspondence problem** — the robot needs to perform the actions of the demonstrator considering its own embodiment
- Depending on the observed actions, **there may be many possible mappings to solve the correspondence problem, or no mapping at all**

The correspondence problem is concerned with how observations of one embodiment can be mapped to another — potentially different — embodiment

Comparison of the Demonstration Types

Kinesthetic teaching

Advantages

- Demonstrations are quite simple to perform
- Learning is simplified — the correspondence problem is eliminated

Disadvantages

- Not all robots have an interface that supports kinesthetic teaching
- Primarily useful for manipulator motions

Comparison of the Demonstration Types

Kinesthetic teaching

Advantages

- Demonstrations are quite simple to perform
- Learning is simplified — the correspondence problem is eliminated

Disadvantages

- Not all robots have an interface that supports kinesthetic teaching
- Primarily useful for manipulator motions

Teleoperation

Advantages

- More flexible demonstration interfaces
- Can be more easily extended to more robot embodiments

Disadvantages

- The demonstration is not necessarily intuitive
- Specialised hardware may be necessary for performing demonstrations

Comparison of the Demonstration Types

Kinesthetic teaching

Advantages

- Demonstrations are quite simple to perform
- Learning is simplified — the correspondence problem is eliminated

Disadvantages

- Not all robots have an interface that supports kinesthetic teaching
- Primarily useful for manipulator motions

Teleoperation

Advantages

- More flexible demonstration interfaces
- Can be more easily extended to more robot embodiments

Disadvantages

- The demonstration is not necessarily intuitive
- Specialised hardware may be necessary for performing demonstrations

Passive observation

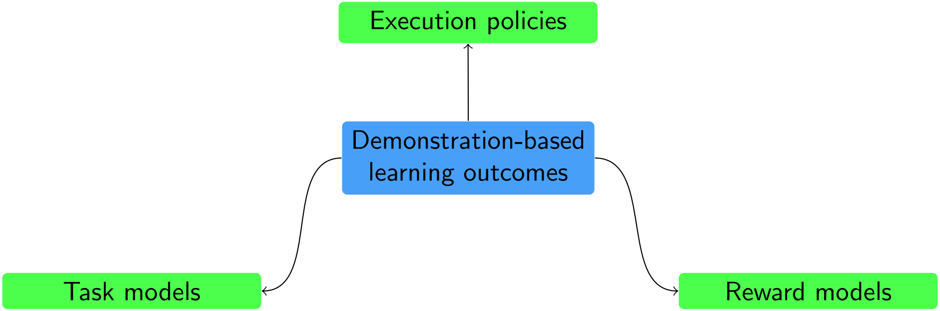
Advantages

- Simplest to perform for demonstrators
- Applicable to all types of robots

Disadvantages

- Challenging information extraction
- The learning is more complicated — the correspondence problem needs to be solved

What Can Be Learned from Demonstrations?



Single Demonstration vs. Multiple Demonstrations

- One dilemma when using learning from demonstration is **whether learning should be done from one demonstration or from a collection of demonstrations**

Single Demonstration vs. Multiple Demonstrations

- One dilemma when using learning from demonstration is **whether learning should be done from one demonstration or from a collection of demonstrations**
- **A single demonstration puts little burden on the demonstrator**

Single Demonstration vs. Multiple Demonstrations

- One dilemma when using learning from demonstration is **whether learning should be done from one demonstration or from a collection of demonstrations**
- **A single demonstration puts little burden on the demonstrator**, but:
  - **it may produce suboptimal behaviour** (demonstrations are rarely perfect)

Single Demonstration vs. Multiple Demonstrations

- One dilemma when using learning from demonstration is **whether learning should be done from one demonstration or from a collection of demonstrations**
- **A single demonstration puts little burden on the demonstrator**, but:
  - **it may produce suboptimal behaviour** (demonstrations are rarely perfect)
  - **a robot cannot generalise beyond the demonstration** (that is, beyond any generalisation capabilities provided by the underlying model representation)

Single Demonstration vs. Multiple Demonstrations

- One dilemma when using learning from demonstration is **whether learning should be done from one demonstration or from a collection of demonstrations**
- **A single demonstration puts little burden on the demonstrator**, but:
  - **it may produce suboptimal behaviour** (demonstrations are rarely perfect)
  - **a robot cannot generalise beyond the demonstration** (that is, beyond any generalisation capabilities provided by the underlying model representation)
- **Multiple demonstrations introduce more variety**

Single Demonstration vs. Multiple Demonstrations

- One dilemma when using learning from demonstration is **whether learning should be done from one demonstration or from a collection of demonstrations**
- **A single demonstration puts little burden on the demonstrator**, but:
  - **it may produce suboptimal behaviour** (demonstrations are rarely perfect)
  - **a robot cannot generalise beyond the demonstration** (that is, beyond any generalisation capabilities provided by the underlying model representation)
- **Multiple demonstrations introduce more variety**, but:
  - **multiple demonstrators are typically required for sufficient diversity** (one demonstrator is likely to perform the same task in a similar way)

Single Demonstration vs. Multiple Demonstrations

- One dilemma when using learning from demonstration is **whether learning should be done from one demonstration or from a collection of demonstrations**
- **A single demonstration puts little burden on the demonstrator**, but:
  - **it may produce suboptimal behaviour** (demonstrations are rarely perfect)
  - **a robot cannot generalise beyond the demonstration** (that is, beyond any generalisation capabilities provided by the underlying model representation)
- **Multiple demonstrations introduce more variety**, but:
  - **multiple demonstrators are typically required for sufficient diversity** (one demonstrator is likely to perform the same task in a similar way)
  - **it may not always be clear how to combine information from multiple demonstrations** (particularly if the demonstrations show seemingly contradictory behaviour)

Segmentation of Demonstrations

- In the case of learning based on passive observations, **demonstrations may contain sequences of actions of interest**
- In such cases, it is required to **perform demonstration segmentation** so that individual actions of interest can be extracted and then learned
- This is **particularly the case when learning full task models from demonstrations**
- **Changing contact interactions with objects** represent one common criterion that is used for segmenting observations
  - NB: These are equivalent to the mode transitions that we discussed in the lecture on learning for manipulation!

Learning Trajectories from One Demonstration: Dynamic Motion Primitives

(Reminder) Homogeneous Second-Order Differential Equations

- Consider the general form of a **homogeneous, second-order differential equation with constant coefficients**:

$$a\ddot{y} + b\dot{y} + cy = 0$$

(Reminder) Homogeneous Second-Order Differential Equations

- Consider the general form of a **homogeneous, second-order differential equation with constant coefficients**:

$$a\ddot{y} + b\dot{y} + cy = 0$$

- The general solution to this equation is of the form

$$y(t) = c_1 e^{k_1 t} + c_2 e^{k_2 t}$$

where k_1 **and** k_2 **are the zeros of the characteristic polynomial** $ak^2 + bk + c = 0$

(Reminder) Homogeneous Second-Order Differential Equations

- Consider the general form of a **homogeneous, second-order differential equation with constant coefficients**:

$$a\ddot{y} + b\dot{y} + cy = 0$$

- The general solution to this equation is of the form

$$y(t) = c_1 e^{k_1 t} + c_2 e^{k_2 t}$$

where k_1 **and** k_2 **are the zeros of the characteristic polynomial** $ak^2 + bk + c = 0$

- Depending on the nature of the zeros, we distinguish between three forms of the general solution:

Real and distinct roots k_1 and k_2 (overdamped system)

$$y(t) = c_1 e^{k_1 t} + c_2 e^{k_2 t}$$

(Reminder) Homogeneous Second-Order Differential Equations

- Consider the general form of a **homogeneous, second-order differential equation with constant coefficients**:

$$a\ddot{y} + b\dot{y} + cy = 0$$

- The general solution to this equation is of the form

$$y(t) = c_1 e^{k_1 t} + c_2 e^{k_2 t}$$

where k_1 **and** k_2 **are the zeros of the characteristic polynomial** $ak^2 + bk + c = 0$

- Depending on the nature of the zeros, we distinguish between three forms of the general solution:

Real and distinct roots k_1 and k_2 (overdamped system)

$$y(t) = c_1 e^{k_1 t} + c_2 e^{k_2 t}$$

Repeated roots $k_1 = k_2 = k$ (critically damped system)

$$y(t) = c_1 e^{kt} + c_2 t e^{kt}$$

(Reminder) Homogeneous Second-Order Differential Equations

- Consider the general form of a **homogeneous, second-order differential equation with constant coefficients**:

$$a\ddot{y} + b\dot{y} + cy = 0$$

- The general solution to this equation is of the form

$$y(t) = c_1 e^{k_1 t} + c_2 e^{k_2 t}$$

where k_1 **and** k_2 **are the zeros of the characteristic polynomial** $ak^2 + bk + c = 0$

- Depending on the nature of the zeros, we distinguish between three forms of the general solution:

Real and distinct roots k_1 and k_2 (overdamped system)

$$y(t) = c_1 e^{k_1 t} + c_2 e^{k_2 t}$$

Repeated roots $k_1 = k_2 = k$ (critically damped system)

$$y(t) = c_1 e^{kt} + c_2 t e^{kt}$$

Complex roots $k_{1/2} = \lambda \pm \mu i$ (underdamped system)

$$y(t) = c_1 e^{\lambda t} \cos(\mu t) + c_2 e^{\lambda t} \sin(\mu t)$$

(Reminder) Nonhomogenous Second-Order Differential Equations

- A nonhomogeneous second-order differential equation has the following general form:

$$\ddot{y} + p(t)\dot{y} + g(t)y + f(t) = 0$$

where $f(t)$ is an **external forcing term**

(Reminder) Nonhomogenous Second-Order Differential Equations

- A nonhomogeneous second-order differential equation has the following general form:

$$\ddot{y} + p(t)\dot{y} + g(t)y + f(t) = 0$$

  where $f(t)$ is an **external forcing term**

- The general solution of such an equation has the form

$$y(t) = y_c(t) + y_p(t)$$

  where
  - $y_c(t)$ **is a complementary solution** (the solution to the homogeneous equation) and
  - $y_p(t)$ **is a particular solution** (a specific solution to the nonhomogeneous equation)

(Reminder) Autonomous Differential Equation

- A dynamical system of equations that does not explicitly depend on the input variable is called an **autonomous system** and has the form

$$\frac{dy}{dt} = f(y(t))$$

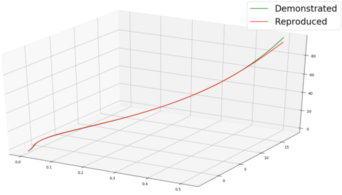
- A **non-autonomous system** is one where the dependence on the input variable is explicit:

$$\frac{dy}{dt} = f(y(t), t)$$

- If the independent variable of the system represents time, an autonomous system is also called **time-invariant**

Dynamic Motion Primitive (DMP) Idea

- In the DMP framework, **a trajectory is modelled as a second-order dynamical system with an external forcing term**
- The system is defined so that **a desired goal can be reached**
- The external forcing term **imposes a given shape on the overall trajectory**
- Learning a trajectory from demonstration is achieved by **parameterising the forcing term and learning its parameters based on a demonstration**



An example trajectory and its reproduction using with a learned DMP

System Formulation

- Formally, a DMP is defined as

$$\tau \ddot{y} = \alpha \left(\beta \left(g - y \right) - \dot{y} \right) + f$$

Here:

- y is the robot's state (e.g. position)
- g is a desired goal
- f is a forcing term
- τ is a time constant (controls the trajectory's speed)
- α and β are positive constants

System Formulation

- Formally, a DMP is defined as

$$\tau \ddot{y} = \alpha\left(\beta\left(g - y\right) - \dot{y}\right) + f$$

  Here:
  - y is the robot's state (e.g. position)
  - g is a desired goal
  - f is a forcing term
  - τ is a time constant (controls the trajectory's speed)
  - α and β are positive constants

- The system will converge to g if it is critically damped and $f \to 0$
  - Critical damping can be achieved by setting $\beta = \alpha/4$

System Formulation

Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors
In Special Collection: CogNet
Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, Stefan Schaal
Neural Computation (2013) 25 (2): 328–373.

- Formally, a DMP is defined as

$$\tau\ddot{y} = \alpha\left(\beta\left(g - y\right) - \dot{y}\right) + f$$

Here:

- y is the robot's state (e.g. position)
- g is a desired goal
- f is a forcing term
- τ is a time constant (controls the trajectory's speed)
- α and β are positive constants

- The system will converge to g if it is critically damped and $f \rightarrow 0$
  - Critical damping can be achieved by setting $\beta = \alpha/4$

- This is often rewritten in a first-order form with $z = \tau\dot{y}$ and

$$\tau\dot{z} = \alpha_z\left(\beta_z(g - y) - z\right) + f$$

Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors
In Special Collection: CogNet
Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, Stefan Schaal
Neural Computation (2013) 25 (2): 328–373.

Forcing Term

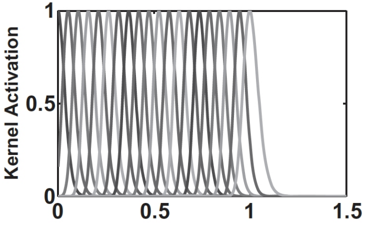
- The DMP forcing term is represented as **a linear combination of basis functions**:

$$f(t) = \frac{\sum_{i=1}^{n} \Psi_i(t) w_i}{\sum_{i=1}^{n} \Psi_i(t)}$$

- Each basis function is **a Gaussian kernel** of the form

$$\Psi_i(x) = \exp\left(-\frac{1}{2\sigma_i}(x - c_i)^2\right)$$

- Learning a DMP amounts to **learning the weights of the forcing term**



Discrete Canonical System

Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors
In Special Collection: CogNet
Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, Stefan Schaal
Neural Computation (2013) 25 (2): 328–373.

- In the previous formulation, **a DMP is a time-dependent system**; this can be problematic if the execution of multiple systems should be coupled

Discrete Canonical System

- In the previous formulation, **a DMP is a time-dependent system**; this can be problematic if the execution of multiple systems should be coupled
- The dependence on time is eliminated by introducing a first-order equation referred to as the **canonical system**:

$$\tau \dot{x} = -\alpha_x x$$

Discrete Canonical System

- In the previous formulation, **a DMP is a time-dependent system**; this can be problematic if the execution of multiple systems should be coupled
- The dependence on time is eliminated by introducing a first-order equation referred to as the **canonical system**:

$$\tau \dot{x} = -\alpha_x x$$

- The solution to this equation is a decaying exponential of the form $ce^{-\alpha_x t/\tau}$

Discrete Canonical System

Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors

In Special Collection: CogNet

Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, Stefan Schaal

Neural Computation (2013) 25 (2): 328–373.

- In the previous formulation, **a DMP is a time-dependent system**; this can be problematic if the execution of multiple systems should be coupled

- The dependence on time is eliminated by introducing a first-order equation referred to as the **canonical system**:

$$\tau \dot{x} = -\alpha_x x$$

- The solution to this equation is a decaying exponential of the form $ce^{-\alpha_x t/\tau}$
- The forcing term is then rewritten in terms of the canonical system as

$$f(x) = \frac{\sum_{i=1}^{n} \Psi_i(x) w_i}{\sum_{i=1}^{x} \Psi_i(x)} x \left(g - y_0\right)$$

Discrete Canonical System

Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors
In Special Collection: CogNet
Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, Stefan Schaal
Neural Computation (2013) 25 (2): 328–373.

- In the previous formulation, **a DMP is a time-dependent system**; this can be problematic if the execution of multiple systems should be coupled
- The dependence on time is eliminated by introducing a first-order equation referred to as the **canonical system**:

$$\tau \dot{x} = -\alpha_x x$$

- The solution to this equation is a decaying exponential of the form $ce^{-\alpha_x t/\tau}$
- The forcing term is then rewritten in terms of the canonical system as

$$f(x) = \frac{\sum_{i=1}^{n} \Psi_i(x) w_i}{\sum_{i=1}^{x} \Psi_i(x)} x \left(g - y_0\right)$$

- As x exponentially decays, the modulation by x ensures that $f \rightarrow 0$
- The term $g - y_0$ achieves spatial scaling depending on the initial distance to the goal

Using DMPs With Multiple Degrees of Freedom

Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors
In Special Collection: CogNet
Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, Stefan Schaal
Neural Computation (2013) 25 (2): 328–373.

- The DMP definition considers only the evolution of a single variable y; however, in practice, we want to control multiple variables (e.g. an end effector trajectory in 3D or the trajectory of multiple arm joints)

Using DMPs With Multiple Degrees of Freedom

- The DMP definition considers only the evolution of a single variable y; however, in practice, we want to control multiple variables (e.g. an end effector trajectory in 3D or the trajectory of multiple arm joints)
- The evolution of multiple degrees of freedom can be synchronised by **coupling the equations of the individual degrees of freedom**

Using DMPs With Multiple Degrees of Freedom

Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors
In Special Collection: CogNet
Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, Stefan Schaal
Neural Computation (2013) 25 (2): 328–373.

- The DMP definition considers only the evolution of a single variable y; however, in practice, we want to control multiple variables (e.g. an end effector trajectory in 3D or the trajectory of multiple arm joints)
- The evolution of multiple degrees of freedom can be synchronised by **coupling the equations of the individual degrees of freedom**
- One way to achieve this is to **use a single canonical system for controlling the evolution of multiple degrees of freedom**
  - Useful for controlling a single manipulator (this is the strategy of choice in our code base)

Using DMPs With Multiple Degrees of Freedom

Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors
In Special Collection: CogNet
Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, Stefan Schaal
Neural Computation (2013) 25 (2): 328–373.

- The DMP definition considers only the evolution of a single variable y; however, in practice, we want to control multiple variables (e.g. an end effector trajectory in 3D or the trajectory of multiple arm joints)
- The evolution of multiple degrees of freedom can be synchronised by **coupling the equations of the individual degrees of freedom**
- One way to achieve this is to **use a single canonical system for controlling the evolution of multiple degrees of freedom**
  - Useful for controlling a single manipulator (this is the strategy of choice in our code base)
- Another alternative is to **couple multiple canonical systems**
  - Can be used for synchronising the motion of multiple manipulators

Canonical System for Rhythmic Motions

Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors
In Special Collection: CogNet
Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, Stefan Schaal
Neural Computation (2013) 25 (2): 328–373.

- The DMP discussed until now has a decaying canonical system; this defines a point attractor and cannot be used to define rhythmic motions

Canonical System for Rhythmic Motions

Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors
In Special Collection: CogNet
Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, Stefan Schaal
Neural Computation (2013) 25 (2): 328–373.

- The DMP discussed until now has a decaying canonical system; this defines a point attractor and cannot be used to define rhythmic motions
- For a phase angle φ, the canonical system can be redefined as

$$\tau \dot{\varphi} = 1$$

Canonical System for Rhythmic Motions

Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors
In Special Collection: CogNet
Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, Stefan Schaal
Neural Computation (2013) 25 (2): 328–373.

- The DMP discussed until now has a decaying canonical system; this defines a point attractor and cannot be used to define rhythmic motions
- For a phase angle φ, the canonical system can be redefined as

$$\tau \dot{\varphi} = 1$$

- For a given amplitute r, f can then be defined as

$$f(\phi, r) = \frac{\sum_{i=1}^{n} \Psi_i(\phi) w_i}{\sum_{i=1}^{x} \Psi_i(\phi)} r$$

Canonical System for Rhythmic Motions

Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors
In Special Collection: CogNet
Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, Stefan Schaal
Neural Computation (2013) 25 (2): 328–373.

- The DMP discussed until now has a decaying canonical system; this defines a point attractor and cannot be used to define rhythmic motions
- For a phase angle φ, the canonical system can be redefined as

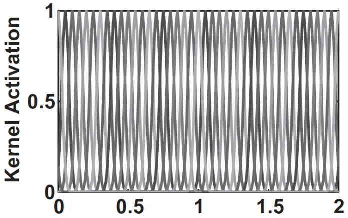
$$\tau\dot{\varphi} = 1$$

- For a given amplitute r, f can then be defined as

$$f(\phi, r) = \frac{\sum_{i=1}^{n} \Psi_i(\phi) w_i}{\sum_{i=1}^{x} \Psi_i(\phi)} r$$

- The basis functions in this case are sinusoidal:

$$\Psi_i(\phi) = \exp\left(h_i\left(\cos\left(\phi - c_i\right) - 1\right)\right)$$



Weight Learning Using Locally Weighted Regression (1/2)

Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors
In Special Collection: CogNet
Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, Stefan Schaal
Neural Computation (2013) 25 (2): 328–373.

- A demonstration is **a sequence of T measurements $D = (d_1, ..., d_T)$, where** $d_t = (t, y_{demo_t}, \dot{y}_{demo_t}, \ddot{y}_{demo_t}), 1 \leq t \leq T$
- Substituting the demonstration points into the equation of the system results in expression for the forcing term:

$$\tau^2 \ddot{y}_{demo} - \alpha_z \left(\beta_z \left(g - y_{demo} \right) - \tau \dot{y}_{demo} \right) = f^*$$

- The learning objective is that of **finding weights that bring the DMP forcing term f as close as possible to the desired forcing term f^***; this objective is expressed as

$$J_i = \sum_{i=1}^{T} \Psi_i(t) \left(f^*(t) - w_i \xi(t) \right)^2$$

- In this equation, $\xi(t) = x(g - y_0)$ for a discrete system and $\xi(t) = r$ for a rhythmic system

Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors
In Special Collection: CogNet
Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, Stefan Schaal
Neural Computation (2013) 25 (2): 328–373.

Weight Learning Using Locally Weighted Regression (2/2)

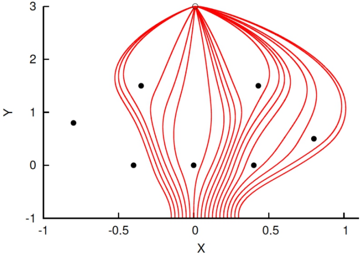
- A solution for the objective can be found using **locally weighted regression**
- Concretely, the weights $w_i, 1 \leq i \leq N$ are found as

$$w_i = \frac{\boldsymbol{\xi}^T \Gamma_i \boldsymbol{f}^*}{\boldsymbol{\xi}^T \Gamma_i \boldsymbol{\xi}}$$

where

$$\boldsymbol{\xi} = \begin{pmatrix} \xi(1) \\ \vdots \\ \xi(t) \\ \vdots \\ \xi(T) \end{pmatrix} \qquad \Gamma_i = \begin{pmatrix} \Psi_i(1) & & & 0 \\ & \ddots & & \\ & & \Psi(t) & \\ & & & \ddots \\ 0 & & & \Psi_i(T) \end{pmatrix} \qquad \boldsymbol{f}^* = \begin{pmatrix} f^*(1) \\ \vdots \\ f^*(t) \\ \vdots \\ f^*(T) \end{pmatrix}$$

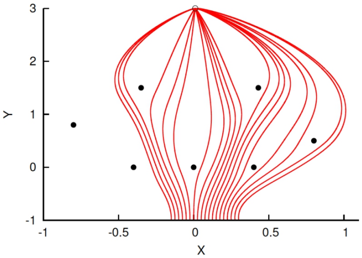
Obstacle Avoidance



H. Hoffmann et al., "Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2009, pp. 2587–2592.

- The dynamic system formulation makes it relatively simple to incorporate additional inputs to the system

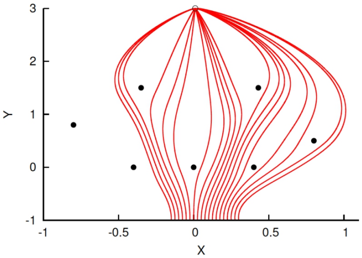
Obstacle Avoidance



H. Hoffmann et al., "Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2009, pp. 2587–2592.

- The dynamic system formulation makes it relatively simple to incorporate additional inputs to the system
- One way in which this can be achieved is by **changing the system equations online to include additional forcing terms** that can change the system's behaviour

Obstacle Avoidance

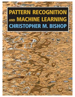


H. Hoffmann et al., "Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2009, pp. 2587–2592.

- The dynamic system formulation makes it relatively simple to incorporate additional inputs to the system
- One way in which this can be achieved is by **changing the system equations online to include additional forcing terms** that can change the system's behaviour
- For obstacle avoidance, **the external force should decay exponentially with the distance to the obstacle** so that the system can eventually return to its original goal
  - This idea is similar to how potential fields work

Learning Trajectories from Multiple Demonstrations: Gaussian Mixture Models

(Reminder) Gaussian Mixture Model (GMM)



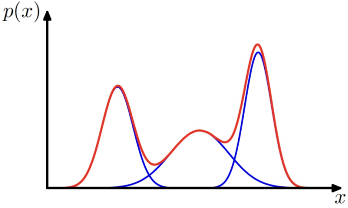
- A multimodal distribution can modelled by **a linear combination of K Gaussian distributions**

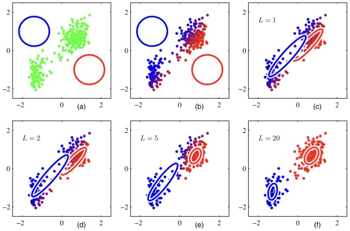
- A Gaussian mixture model (GMM) has the form

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \Sigma_k)$$

- The parameters π_k are the **mixing coefficients**, such that $0 \leq \pi_k \leq 1$ and $\sum_{k=1}^{K} \pi_k = 1$

- Given data to be modelled by a GMM, the **expectation-maximisation** algorithm is used to find the parameters of the mixture components





Learning from Multiple Demonstrations Objective

- When learning from multiple demonstrations, we are given a **collection of M demonstrations** $\mathrm{T} = \{D_1, ..., D_M\}$
  - Each $D_i, 1 \leq i \leq M$ can be defined as $D_i = (t, \boldsymbol{x}_t, \dot{\boldsymbol{x}}_t), 1 \leq t \leq T$

Learning from Multiple Demonstrations Objective


IEEE TRANSACTIONS ON ROBOTICS, VOL. 27, NO. 5, OCTOBER 2011 943

Learning Stable Nonlinear Dynamical Systems With Gaussian Mixture Models

S. Mohammad Khansari-Zadeh and Aude Billard

- When learning from multiple demonstrations, we are given a **collection of M demonstrations** $\mathrm{T} = \{D_1, ..., D_M\}$
  - Each $D_i, 1 \leq i \leq M$ can be defined as $D_i = (t, \boldsymbol{x}_t, \dot{\boldsymbol{x}}_t), 1 \leq t \leq T$
- The learning objective is to find a model that:

IEEE TRANSACTIONS ON ROBOTICS, VOL. 27, NO. 5, OCTOBER 2011 943

Learning Stable Nonlinear Dynamical Systems
With Gaussian Mixture Models
S. Mohammad Khansari-Zadeh and Aude Billard

Learning from Multiple Demonstrations Objective

- When learning from multiple demonstrations, we are given a **collection of M demonstrations** $\mathrm{T} = \{D_1, ..., D_M\}$
  - Each $D_i, 1 \leq i \leq M$ can be defined as $D_i = (t, \boldsymbol{x}_t, \dot{\boldsymbol{x}}_t), 1 \leq t \leq T$
- The learning objective is to find a model that:
  - **summarises the distribution of the M demonstrations**

Learning from Multiple Demonstrations Objective

IEEE TRANSACTIONS ON ROBOTICS, VOL. 27, NO. 5, OCTOBER 2011 943
Learning Stable Nonlinear Dynamical Systems With Gaussian Mixture Models
S. Mohammad Khansari-Zadeh and Aude Billard

- When learning from multiple demonstrations, we are given a **collection of M demonstrations** $\mathrm{T} = \{D_1, ..., D_M\}$
  - Each $D_i, 1 \leq i \leq M$ can be defined as $D_i = (t, \boldsymbol{x}_t, \dot{\boldsymbol{x}}_t), 1 \leq t \leq T$
- The learning objective is to find a model that:
  - **summarises the distribution of the M demonstrations**
  - **makes it possible to sample trajectories for execution**

Learning from Multiple Demonstrations Objective

- When learning from multiple demonstrations, we are given a **collection of M demonstrations** $\mathrm{T} = \{D_1, ..., D_M\}$
  - Each $D_i, 1 \leq i \leq M$ can be defined as $D_i = (t, \boldsymbol{x}_t, \dot{\boldsymbol{x}}_t), 1 \leq t \leq T$
- The learning objective is to find a model that:
  - **summarises the distribution of the M demonstrations**
  - **makes it possible to sample trajectories for execution**
- Such a model can be represented by a GMM, and the learning objective becomes that of learning the GMM parameters

IEEE TRANSACTIONS ON ROBOTICS, VOL. 27, NO. 5, OCTOBER 2011 943
Learning Stable Nonlinear Dynamical Systems With Gaussian Mixture Models
S. Mohammad Khansari-Zadeh and Aude Billard

Trajectory GMM

- The underlying system model is a first-order equation of the form

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + \boldsymbol{\epsilon}$$

Trajectory GMM

- The underlying system model is a first-order equation of the form

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + \boldsymbol{\epsilon}$$

- Given T, f is represented by a GMM as the joint distribution

$$P(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \sum_{k=1}^{K} \pi_k P(\boldsymbol{x}, \dot{\boldsymbol{x}}|k) = \sum_{k=1}^{K} \pi_k P\left(\boldsymbol{x}^{t,i}, \dot{\boldsymbol{x}}^{t,i}|k\right), \; 1 \leq i \leq M, \; 1 \leq t \leq T$$

Trajectory GMM

- The underlying system model is a first-order equation of the form

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + \boldsymbol{\epsilon}$$

- Given T, f is represented by a GMM as the joint distribution

$$P(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \sum_{k=1}^{K} \pi_k P(\boldsymbol{x}, \dot{\boldsymbol{x}}|k) = \sum_{k=1}^{K} \pi_k P\left(\boldsymbol{x}^{t,i}, \dot{\boldsymbol{x}}^{t,i}|k\right), \; 1 \leq i \leq M, \; 1 \leq t \leq T$$

- As the components of the mixture are Gaussian distributions, their parameters are the mean and covariance matrix, which are represented as

$$\boldsymbol{\mu}^k = \begin{pmatrix} \boldsymbol{\mu}^k_{\boldsymbol{x}} \\ \boldsymbol{\mu}^k_{\dot{\boldsymbol{x}}} \end{pmatrix} \qquad \Sigma^k = \begin{pmatrix} \Sigma^k_{\boldsymbol{x}} & \Sigma^k_{\boldsymbol{x}\dot{\boldsymbol{x}}} \\ \Sigma^k_{\dot{\boldsymbol{x}}\boldsymbol{x}} & \Sigma^k_{\dot{\boldsymbol{x}}} \end{pmatrix}$$

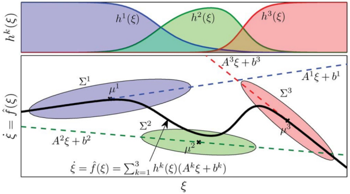
Trajectory Execution

- During execution, **appropriate velocities $\dot{x}$ need to be calculated** so that a robot can execute **a trajectory based on the model**
- The expectation of the posterior estimate $P(\dot{\boldsymbol{x}}|\boldsymbol{x})$ can be found as

$$\dot{\boldsymbol{x}} = \sum_{k=1}^{K} \frac{\pi_k P(\boldsymbol{x}|k)}{\sum_{i=1}^{K} \pi_i P(\boldsymbol{x}|k)} \left(\boldsymbol{\mu}_{\dot{\boldsymbol{x}}}^k + \Sigma_{\boldsymbol{x}\dot{\boldsymbol{x}}}^k \left(\Sigma_{\boldsymbol{x}}^k \right)^{-1} \left(\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{x}}^k \right) \right)$$
$$= \sum_{k=1}^{K} h^k(\boldsymbol{x}) \left(A^k \boldsymbol{x} + \boldsymbol{b}^k \right)$$

where

$$h^k(\boldsymbol{x}) = \frac{\pi_k P(\boldsymbol{x}|k)}{\sum_{i=1}^{K} \pi_i P(\boldsymbol{x}|k)} \quad A^k = \Sigma_{\boldsymbol{x}\dot{\boldsymbol{x}}}^k \left(\Sigma_{\boldsymbol{x}}^k \right)^{-1} \quad \boldsymbol{b}^k = \boldsymbol{\mu}_{\dot{\boldsymbol{x}}}^k - A^k \boldsymbol{\mu}_{\boldsymbol{x}}^k$$



Summary

- Learning from demonstration is a technique based on which a robot acquires learning data from expert demonstrations

Summary

- Learning from demonstration is a technique based on which a robot acquires learning data from expert demonstrations
- Demonstrations can be collected in different ways: using kinesthetic teaching, by teleoperation, or based on observations from an external perspective

Summary

- Learning from demonstration is a technique based on which a robot acquires learning data from expert demonstrations
- Demonstrations can be collected in different ways: using kinesthetic teaching, by teleoperation, or based on observations from an external perspective
- Learning from demonstration has various uses, namely it can be used to learn execution policies, complete task models, or reward models

Summary

- Learning from demonstration is a technique based on which a robot acquires learning data from expert demonstrations
- Demonstrations can be collected in different ways: using kinesthetic teaching, by teleoperation, or based on observations from an external perspective
- Learning from demonstration has various uses, namely it can be used to learn execution policies, complete task models, or reward models
- In the dynamic motion primitives (DMP) framework, motion is modelled by a second-order dynamical system; this system has a force term whose parameters are learned given a demonstration

Summary

- Learning from demonstration is a technique based on which a robot acquires learning data from expert demonstrations
- Demonstrations can be collected in different ways: using kinesthetic teaching, by teleoperation, or based on observations from an external perspective
- Learning from demonstration has various uses, namely it can be used to learn execution policies, complete task models, or reward models
- In the dynamic motion primitives (DMP) framework, motion is modelled by a second-order dynamical system; this system has a force term whose parameters are learned given a demonstration
- Multiple demonstrations can be used to learn a probabilistic model (a trajectory envelope), for instance in the form of a Gaussian mixture model