Learning a Repertoire of Actions with Deep Neural Networks

Mine Melodi Caliskan
CmpE58Y - Robot Learning

April 9, 2018

Outline

Introduction

During their development, children learn an increasingly complex set of actions while interacting with their environment.

- **Assimilation**: They build such a repertoire by progressively adapting known gestures to novel objects (Piaget [1])
- **Accommodation**: Differentiating novel actions for objects which cannot be assimilated smoothly by known actions

Introduction

Despite recent progress in control and perception, robots are still far from such capabilities.

- Several works aim to optimize a movement primitive depending on some parameters to generate movements which generalize to new objects. (e.g context parameters to DMPs)
- Some approaches provide a set of action for the robot which learns to associate these actions with different objects and contexts.
- Some works fuse both approaches by learning a repertoire of movement primitives, but are limited either by **the curse of dimensionality** or by **the amount of required prior knowledge.**

Motivation

Problem Statement: Endowing a robot with the capability to learn a repertoire of actions using as little prior knowledge as possible.

- This work presents a neural architecture which is able to learn a repertoire of actions with very few priors.
- The proposed architecture can generalize learned actions to new contexts through a parametrization of each action.
    - Gated connections to enable a distributed representation of actions shared by the whole repertoire.
    - Deep learning paradigm provides a natural framework to use the powerful dimensionality reduction properties.

Goal

Task: Illustrate this architecture on a digit writing task.

- First, train the network to generate trajectories for each of the ten digits.
- Second, illustrate the parametrization of actions by teaching the network to draw rotated digits.

RELATED WORK

"Learning a repertoire of movement primitives" using **Dynamical Movement Primitives (DMPs)**:

- Necessitates prior knowledge from an external designer:
  - Adequate parametrized primitives
  - Relevant context variables to circumvent the curse of dimensionality.
- Generally restricted to simple reaching or cyclic movements.

RELATED WORK

"Reproducing dynamical sequences" using neural networks:

- **Elman's network**: The hidden layer is copied at each time step into a context layer and fed back for the next time step.
    - Difficult to train due to the vanishing gradient problem.
- **Echo state networks and liquid state machines**: Recurrent hidden layers with randomly chosen weights which remain untrained.
    - Necessitate large hidden layers whose connectivity matrices are carefully tuned.

BACKGROUND: DEEP NETWORKS AND GATED CONNECTIONS

Though neural networks with a single hidden layer are universal approximators:

- The number of units required to approximate a given function grow exponentially with the desired accuracy.
- To decrease the number of required units, multiple levels of hidden layers are required.
  $\Rightarrow$ Vanishing gradient problem, and many poor local optima.
- To overcome : **Pre-train** each layer to learn a good representation of its input.
- **Autoencoders**: Aims at minimizing the reconstruction error.

Autoencoders

Given a visible input v and a hidden layer h, it learns the encoding:

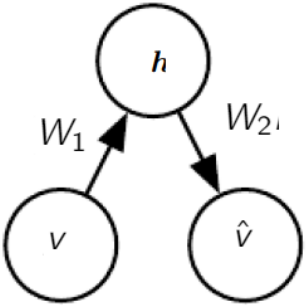
- $h = \sigma(W_1 v + b_h)$

This encoding is then decoded to reconstruct the input using:

- $\hat{v} = \sigma(W_2 h + b_v)$

The network is trained by backpropagating the reconstruction error:

- $||v - \hat{v}||_2^2$



BACKGROUND: Gated Connections

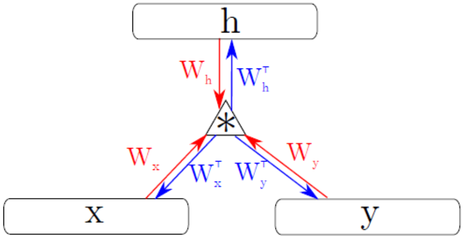
- After learning a relation between inputs, e.g the temporal evolution of a variable, it is useful to use **gated connections**.
- Such connections are useful to learn representations of multi-dimensional relations between different variables.
- Factorizing these connections reduces the number of weights and makes learning easier.

BACKGROUND: Gated Connections

Given two different input x and y, such connections introduce an intermediate factors layer, which performs an element-wise multiplication between projections f_x and f_y of both inputs. The resulting factors f_h are then projected to the hidden layer h.

- $f_x = W_x x$, $f_y = W_y y$
- $f_h = f_x * f_y$ (element-wise)
- $h = \sigma(W_h^T f_h)$

The role of x, y and h can be exchanged, allowing for instance to compute a reconstruction of y given x and h.



ARCHITECTURE

- Mathematical Framework
- Neural Network Implementation

ARCHITECTURE: Mathematical Framework

Representation of action is given by $q_t = f(s_t, m_t, c_t)$

- q_t: the command at time t (e.g. torques, Cartesian or joints velocities)
- s_t: the state of the system at time t (e.g the current joint positions)
- m_t: a memory of past states
- c_t: the current goal of the system (e.g a symbolic representation of the pursued goal)

Mathematical Framework: A robotic writing task

They control the end-effector of a robot in the three dimensional Cartesian space to write different digits.

$$q_t = f(s_t, m_t, c_t)$$

- q_t: encodes the desired Cartesian velocities $\dot{x}_t$ of the end-effector
- s_t: the current Cartesian position x_t of the end effector,
- m_t: stores some memory about past positions
- c_t: a symbolic representation of the action, a, being performed (i.e. the digit being written), represented by a boolean action vector a containing only zeros, except for the desired digit for which the corresponding bit is set to one.

$\Rightarrow q_t = f(s_t, m_t, c_t) \approx \dot{x}_t = f(x_t, m_t, a)$

Mathematical Framework: A robotic writing task

They further factorize $\dot{x}_t = f(x_t, m_t, a)$ introducing an instantaneous goal function, $g(m_t, a)$, which depends on :

- a: the action being performed (the "overall goal")
- m_t: its current progress

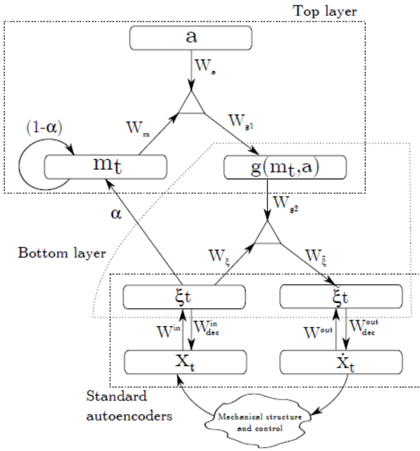
$$\Rightarrow \dot{x}_t = f(x_t, g(m_t, a))$$

- Reduces the dimensionality of the input when the dimensionality of the output of g is smaller than the dimensions of m_t and a.
- Allows function f to learn synergies which can be re-used by any of the actions from the repertoire.

The Neural Network Architecture

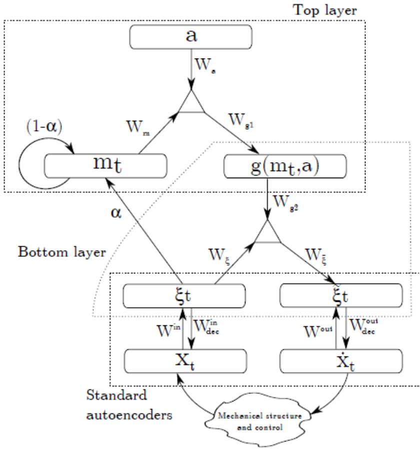
The proposed network contains two sublayers:

- **Top layer**: Corresponds to the computation of g which takes m_t and a as input.
- **Bottom layer**: Corresponds to the computation of f and takes x_t and the output of the first sub-network as input.



The Neural Network Architecture

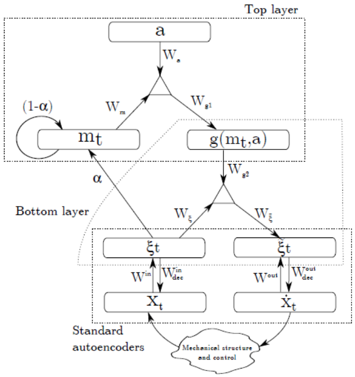
- The positions x of the end-effector are first encoded by an autoencoder.
- Similarly, Cartesian velocities are encoded by a second autoencoder.
- m_t is related to the past positions by a simple exponential window:
  - $m_t = (1-\alpha)m_{t-1} + \alpha\xi_t$



The Neural Network Architecture

Each sub-network uses gated connections, which results in a factorization of representations at each layer. Thus, g and h are computed as:

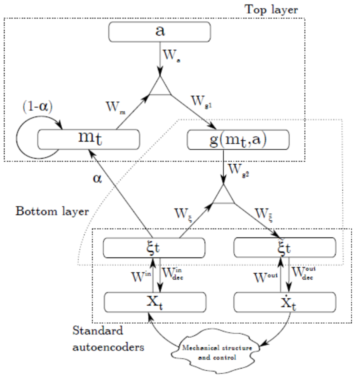
- $g(m_t, a) = \sigma(W_{g_1}((W_m m_t) * (W_a a)))$
- $\tilde{\xi}_t = \sigma(W_{\tilde{\xi}}(W_\xi \xi_t) * (W_{g_2} g(m_t, a)))$



The Neural Network Architecture

- Training this architecture consists in learning all connectivity matrices W_* by minimizing the reconstruction error of predicted Cartesian velocities $\hat{\dot{x}}$ given the positions and the vector a:

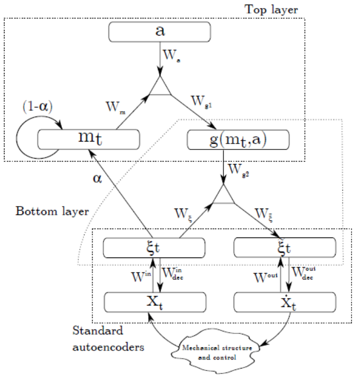
$$\sum_t ||\hat{\dot{x}}_t - \dot{x}_t||^2$$



The Neural Network Architecture

Each sub network is pre-trained independently.

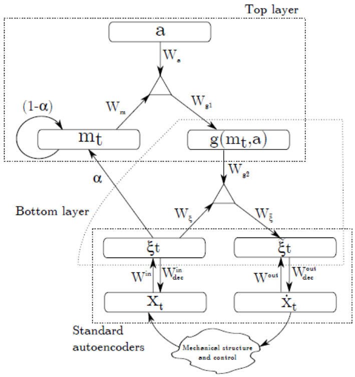
- First, the autoencoders are trained to learn a representation ξ of $x$and a representation $\tilde{\xi}$ of $\dot{x}$ by minimizing the reconstruction error of x and $\dot{x}$ respectively.



The Neural Network Architecture

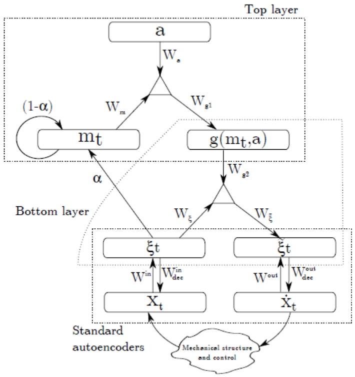
Then, the lower level gated network is trained to learn a representation g_t, given ξ_t and $\tilde{\xi}_t$, by minimizing the distance with their reconstruction ξ_t^{recons} and $\tilde{\xi}_t^{recons}$:

- $g_t = \sigma(W_{g_2}^T(W_\xi \xi_t * W_{\tilde{\xi}}^T \tilde{\xi}_t)))$
- $\xi_t^{recons} = \sigma(W_\xi^T(W_{g_2} g_t * W_{\tilde{\xi}}^T \tilde{\xi}_t)))$
- $\tilde{\xi}_t^{recons} = \sigma(W_{\tilde{\xi}}(W_\xi \xi_t * W_{g_2} g_t))$



The Neural Network Architecture

- Then, the upper level gated network is trained to infer the intermediate goal g_t provided by the lower level network, given m_t and a, by minimizing the difference between g_t and $g(m_t, a)$
- Finally, a global gradient descent involving all matrices is performed to fine-tune the network to minimize the global prediction error of $\dot{x}_t$.



Experiments

They test the architecture on a writing task with the iCub humanoid robot.

- First, the network learns to generate trajectories for each of the ten digits.
- In a second experiment, they illustrate the parametrization of actions by teaching the network to draw rotated digits.

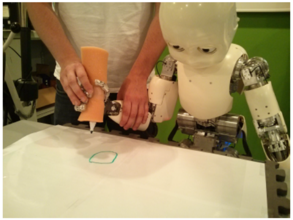


Fig. 3. Experimental setup. The robot is controlled in zero-torque mode, while a human operator moves its arm to write the digits between 0 and 9. The three dimensional Cartesian positions of the end-effector are recorded at approximately 100Hz.

Experiments

- They record the arm trajectories of the robot being manipulated by a human operator to write the digits between 0 and 9
- Each digit basically corresponds to a different action from a repertoire.
- A total of 76 trajectories for each digit are recorded and used to train the network.

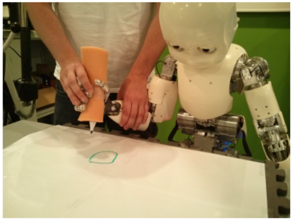


Fig. 3. Experimental setup. The robot is controlled in zero-torque mode, while a human operator moves its arm to write the digits between 0 and 9. The three dimensional Cartesian positions of the end-effector are recorded at approximately 100Hz.

Experiments

- For each trajectory, the initial position is considered as the origin of the three dimensional Cartesian frame.
- Each trajectory then consists of the sequence of Cartesian positions sampled at approximately 100Hz.
- The memory m_t is set to 0 at the beginning of each trajectory.

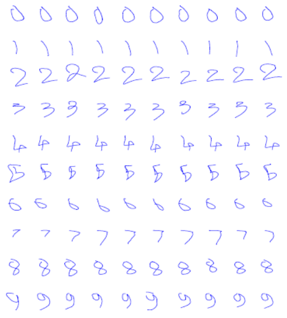


Fig. 4. Samples of recorded trajectories from the iCub robot. Each trajectory is plotted in a 12x8 cm box. Only two dimensions are plotted, the third being constant for most digits, except for 4 and 5 which necessitate to lift the pen.

Experiments: Parametrization of actions

- The network is trained on the recorded trajectories using mini-batches of 1000 points, i.e. 1000 Cartesian positions along with the corresponding Cartesian velocities.
- The network is then tested to generate trajectories by iterating for a number of time steps equal to the mean duration of recorded trajectories for the considered digit.
- They close the loop between positions x_t and predicted velocities $\hat{\dot{x}}_t$ by simulating the movement of the end-effector according to:

$$x_{t+1} = x_t + 0.001 \times \eta \times (\hat{\dot{x}}_t + \nu)$$

Experiments: Repertoire of actions

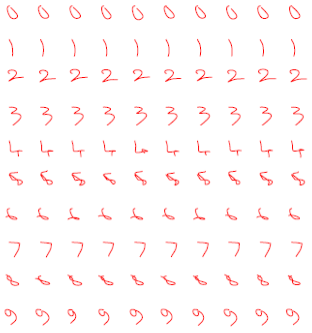


Fig. 5. Trajectories generated by the network. Each trajectory is plotted in a 12x8 cm box. Noise is added during the generation of each trajectory to evaluate the robustness of the network.

Experiments: Parametrization of actions

- The network is trained to learn a set of different primitives based on a symbolic representation of actions.
- The network also handle a parametrization of such actions
  - Generate new trajectories from the recorded dataset, by rotating one sample of each digit by an angle
  - Apply the corresponding rotation matrices to all points of the trajectories
  - Provides a total of 50 trajectories for the training set, 5 trajectories for each digit. (Previously, 760 demonstrations)

Experiments: Parametrization of actions

- The rotation parameter is added to the top layer a as two additional units taking values
- The resulting top layer a is thus composed of 10 binary units to represent the digits, and two real-valued units representing the rotation angle.

All the other parameters are the same as for the previous experiment.

Experiments: Parametrization of actions

- After training, the network is used to generate trajectories with rotation angles
- No noise is added for the generation

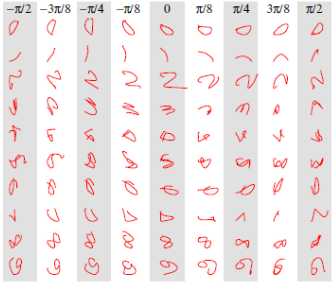
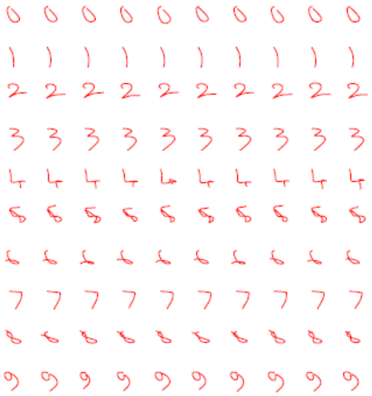


Fig. 6. Trajectories generated by the network for different rotation angl
Digits on gray background correspond to rotation angles which were p
sented to the network during training (only one trajectory for each (di
rotation angle) pair was provided for training). Other trajectories show
generalization capabilities of the network.

Discussion

In this work, the memory layer allows the network to produce different desired velocities for a same current position x , without explicitly modeling features:

- such as corners (e.g. digits 2 and 9)
- or cross points (e.g. digits 4, 6 and 8).



Discussion

- In a DMP approach, this would require either carefully chosen parametrized primitives, or a segmentation of such trajectories into several segments
- However, this approach does not use prior knowledge, it can prevent the network from learning very specific features. (e.g digit 5)
- Improving such trajectories would necessitate either an optimization framework such as reinforcement learning to drive the network towards better trajectories, or a more sophisticated memory able to extract and store relevant information about important via-points.

Discussion

This approach aims at endowing control architectures with the powerful dimensionality reduction capabilities of deep networks:

- It separates the lowest level of control from the high level trajectory planning by introducing the intermediate goal representation $g(m_t, a)$.
- Using gated connections, it factorizes knowledge from which we can expect an increasingly faster learning of new trajectories when the amount of knowledge increases.

Future Work

- Wider architecture to provide a compact and meaningful representation of perception using dimensionality reduction capabilities of deep networks. learning.
- Learning the recurrent weights from the memory layer, possibly involving long short term memory units to make the network more generic.

References

- J. Piaget, The Origins of Intelligence in Children. WW Norton & Co, 1952.
- Alain Droniou, Serena Ivaldi, Olivier Sigaud. Learning a Repertoire of Actions with Deep Neural Networks. Joint International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob), Oct 2014, Italy. 6 p., 2014. <hal-01065741>