Part 1: diffeomorphic matching for imitation learning

Nicolas Perrin perrin@isir.upmc.fr
CNRS / Sorbonne Université - ISIR
Problem: learn a vector field with asymptotic guarantees.

Specifically: global asymptotic stability.

\[ \mathbf{v} = f_{\theta}(\mathbf{x}) \]
Often, $f_\theta$ has the form:

$$f_\theta = \sum_i \alpha_i g_{\theta_i}(x)$$

*Issue*: the sum of asymptotically stable vector fields is not asymptotically stable.
\[ f_\theta = \sum_i \alpha_i g_{\theta_i}(x) \]

*But:* the sum is asymptotically stable if all the \( g_{\theta_i} \)'s are compatible with the same Lyapunov function.
It is difficult to construct complex Lyapunov functions...

...so the seminal approach SEDS [*] was based on quadratic Lyapunov functions, which is quite restrictive.

[*] Khansari-Zadeh et al., "Learning stable nonlinear dynamical systems with Gaussian mixture models", IEEE Trans. on Robotics 2011
More expressive convex sets of Lyapunov candidates have been used (e.g. WSAQF [*]) to turn the search of a Lyapunov function to an optimization problem, but they are still quite restrictive.

We proposed a method based on another property:

A stable vector field remains stable after application of a diffeomorphism.
\[ \mathbf{x} : k_{\rho N}(\mathbf{c}_N, \mathbf{x}) = 0.5 \]
<table>
<thead>
<tr>
<th></th>
<th>( N )</th>
<th>our algorithm</th>
<th>LDDMM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Learning:</strong> average</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>duration of the</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>construction of ( \Phi )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0.25 s</td>
<td>2.78 s</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>0.25 s</td>
<td>14.5 s</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>0.26 s</td>
<td>53.3 s</td>
<td></td>
</tr>
<tr>
<td><strong>Forward evaluation:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>average duration of the</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>computation of ( \Phi(X) )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>3.05 ms</td>
<td>157 ms</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>3.35 ms</td>
<td>804 ms</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>3.72 ms</td>
<td>3130 ms</td>
<td></td>
</tr>
<tr>
<td><strong>Backward evaluation:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>average duration of the</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>computation of ( \Phi^{-1}(Y) )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>29.8 ms</td>
<td>145 ms</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>35 ms</td>
<td>798 ms</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>38.5 ms</td>
<td>3110 ms</td>
<td></td>
</tr>
<tr>
<td><strong>Accuracy:</strong> average</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value of ( \text{dist}(\Phi(X), Y) )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>(3.49\times10^{-3})</td>
<td>(18.2\times10^{-3})</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>(8.32\times10^{-3})</td>
<td>(22.2\times10^{-3})</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>(9.51\times10^{-3})</td>
<td>(22\times10^{-3})</td>
<td></td>
</tr>
<tr>
<td>Dimensionality</td>
<td>2</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>----------------</td>
<td>------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>Training time with SEDS (s)</td>
<td>10.74</td>
<td>26.96</td>
<td>81.14</td>
</tr>
<tr>
<td>Training time with our method (s)</td>
<td>0.482</td>
<td>0.456</td>
<td>0.498</td>
</tr>
</tbody>
</table>
Possible extensions:
Φ maps a circular limit cycle to the Van der Pol limit cycle.

- red trajectory with the DS obtained
- black trajectory with the Van der Pol oscillator
Part 2: a specific approach to reinforcement learning for robotics?

Nicolas Perrin  perrin@isir.upmc.fr
CNRS / Sorbonne Université - ISIR
1. Exploration
2. Feature/structure engineering
3. Composition
Preamble: a few words on RL

Objective: optimize a policy \( \pi_\rho(s) \) to collect reward.

A value function \( V_\theta(s) \) or \( Q_\mu(s, a) \) is also trained to approximate \( E_{\pi^*, s_0=s} \left\{ \sum_i \gamma^i r(s_i) \right\} \). It is used as a substitute to the rewards to get gradients for \( \rho \) updates.

On-policy or Off-policy methods.

Exploration is interesting both to find new solutions and to get good gradients for \( V_\theta \).
1. Exploration

Better exploration may be obtained via motion planning, exploiting resets and determinism in simulation (and maybe the fact that outputs are torques).
Attempts on the learning to run challenge:
Attempts on the learning to run challenge:
Attempts on the learning to run challenge:
Attempts on the learning to run challenge:
2. Feature/structure engineering

A function transforms inputs into outputs:

\[
a = f_\theta(s)
\]

Learning objective: optimize \( \theta \).

A typical structure when \( f \) is a deep neural network for image classification:
Convolutions:

3 grids of neurons in output

one grid of neurons in input

pixel value
Example of convolution filter (Sobel):

Two Sobel filters ($x$ and $y$) applied to an image:
Convolutions are great feature extractors for images.

Can we find something similar when the inputs are either:

- joint angles and velocities + free-floating base position and velocity,
- or position and velocities for the floating base and every rigid body of the robot?

(do we also add torques, some contact points and contact forces?)
In other words: are there robotic control equivalents of convolutions?
Some “canonical” operations or tools in robotic control:

- Going from joint space to Euclidean space, and vice versa.
- PID.s.
- Solving OCPs.
- Euler-Lagrange equation.
- Using simplified dynamical models (=convexifying optimization problems)?
- Relying on important quantities, such as kinetic energy, center of mass, centroidal angular momentum, center of pressure, end-effector positions
- Composing tasks, . . .
**Actor-critic RL:** train both $\pi_{\rho}(s)$ and $V_{\theta}(s) = E_{\pi^*, s_0 = s} \left\{ \sum_i \gamma^i r(s_i) \right\}$.

$\pi_{\rho}$ is trained just because it is hard to compute $\arg\max_a V_{\theta}(s')$ (or $\arg\max_a (Q_{\mu}(s, a)$ in the case of a state-action value function).

In robotics, training $\pi_{\rho}$ may be a bad idea:

- the actions are complex and need to be precise;
- they could probably be obtained analytically because a (relatively) good model of the dynamics is known;
- focusing on learning costs (the value function) may be better because typically costs do not have to be as precise as actions.
\[ a^* = \arg \max_a \{ V_\theta(s') \mid s \rightarrow_a s' \} \]

\[ s' = s + \delta s: \]

\[ \max_{\delta s} V_\theta(s + \delta s) \]

\[ \|\delta s\| \leq K \]

\[ s' = s(t + \delta t) = s + b(s) + M(s)a: \]

\[ \max_a V_\theta(s + b(s) + M(s)a) \]

\[ \|a\| \leq K_a \]
Other potential idea: when learning \( V_\theta(s) \) or \( Q_\mu(s, a) \), put redundancies inside the functions to facilitate the discovery of good correlations (and thus the training of \( V_\theta \) or \( Q_\mu \)):

\[
V_\theta(s) = W_\theta(q, \dot{q}, x(q), \dot{x}(q, \dot{q}), f_c)
\]
3. Composition
SAC-X (Google Deepmind)