# Human Movement Prediction and Modelling with Reinforcement Learning : Challenges and Directions for Research. A Personal Perspective.

Pankayaraj. P

## Summary

While the developments over the years in Reinforcement learning (RL) have lead to agents achieving human level performance and beyond in games [1] [2], when it comes to modelling human like movement via RL still remains a challenge due to several factors such as complexity, delay, constraints in practicality etc. As a part of my graduate school application in this work on an abstract level I will present and explore my research motivation about the potential directions that are open for exploration when it comes to modelling human-like movement using Reinforcement learning. This article is structured in a way such that in the first section 1 I will, within my knowledge, first explore the ***limitations*** that human movement modelling faces when using RL (in an environmental level). In the subsequent section 2 I will present the ***potential research direction*** that can be explored in order to tackle these problems and the additional complications these methods induce. And finally, in the ***research Question*** 3 section the potential ideas are finalized as set of questions and answers. In order to keep the flow of the article some ideas that complements the suggestion here are explained in the ***Appendices*** 3

## Contents

# 1 Environment Level Challenges

The challenges I present here for the most part mentioned in the work [3].

## 1.1 Challenges in Motor Level Control

1. **Curse of Dimensionality** : Classic Mujoco and Roboschool environments use torque forces to control the human skeleton model which is free of friction. The number of torque control needed is relatively low (17) which is lower than if we are to have a control over a muscular actuator (induces the curse of dimensionality in the action space)

2. **Control Delay** : If we are to use muscle tendon actuators these will be subjected to sensor delays and control signal delays. There is a general delay between control signal and torque eventually getting generated on the bones . Though it can be mitigated by existing data it won't fit in new scenarios thus a robust system should be there to model them.

3. **Muscle actuation to force mapping** : Different factors are involved in transmitting force from muscle to bones. Firstly the way muscles generate force is dependent on their length, velocity and activation level. There is also an interface to transmit these forces from those muscles to the bone which is a compliant tendon which can further complicate the muscle mapping.

4. **Computational cost** : The cost of simulation is higher thus sample efficiency is preferred.

5. **Higher Cost of Breakage** Muscular actuators can have a certain limit or safe operational range which must be observed in order to prevent wear or damages which can be quite expensive in real life.

# 2 Potential Solutions and Research Directions

## 2.1 Control in Lower Level

### 2.1.1 Addressing the control delays

There are two types of delays that can happen in the case of humanoid control in RL. One is observation delay and the other one is action delay given that the reward is something that is computed instantaneously. In case of the ***observation delay***, that is the current state of the humanoid is only realized after some delay, we would need the past actions since the currently available state to the controller in order to make the decision process MDP [4]. This would ***make the reward function stochastic*** even when the initial one is not, as now we don't the actual current state of the body for sure and even with the added information of past actions we still have to face the stochasticity in the environment which would translate into the reward. Similarly in case of the ***action delay*** as well we need to argument the state information with available state and past actions but in contrary to the previous case it ***won't make the reward function stochastic***.

There are some recent developments [5] which propose as Delay-Correcting Actor-Critic which can be off policy thereby being sample efficient. There is a potential for these to be explored or used in an application standpoint for our setting of interest.

### 2.1.2 Generalization of Muscle actuation to force mapping

When it comes to finding a proper mapping from muscle actuators to the forces that are being exerted on the bones generalization becomes vital because ***not all muscles will be exposed to movement*** during the learning process thus some less used muscles will be starved off data. The generalisation could be done in two ways. Firstly, this can be treated as a continual or a multi task learning in a supervised setting. That is to use an embedding function to map the actuator control to less complex embedding. This function can be used in a continual learning setting thus some generalisation can be made from the previously learned knowledge from more exposed muscles. But, this does require the representation and dimensions of actuator control to be similar.

Another way is to break the motor control problem into two hierarchies. As a background, it is indeed easier for an algorithm to learn movement when the humanoid model is presented as

just torques acting on the skeleton. But the reason why this model can result in some unrealistic movements is because some torque configurations are not plausible to be exerted by the muscle actuators.

Here we can break the learning into two hierarchies (given we have access to the simpler torque model of the same humanoid) one with the simpler model governed by torque (acting as a higher tier) another is a problem of muscle actuators proving the necessary configurations to generate the required torques (acting as a lower tier). Now we can treat the lower tier problem as a continual RL problem where the policy is to generate muscle actuator configuration that would result in the torque required by the higher tier policy. We can either maintain ***a singular neural network or multiple ones*** based on the Pareto optimality [6]. Pareto optimality is a situation where a given system can't be further improved to perform better for a set of problems without worsening the other. In our setting for a better generalisation we do need to have different neural networks rather than a single one if we want to keep the generalised performance across all the muscle actuators. One potential avenue to explore the variability in the tasks so that we can decide weather to generalize it with the current task or to go with a different neural network for it is ***via the use of the distance between the steady state distribution*** induced by the optimal policy in each task individually. An ***explanation*** about the steady state distributions estimation and potential can be accessed in the ***Appendices*** 3 of the article. But a caveat of this direction is that we will be introducing further computational costs as we need the optimal policy so an exploration for other effective methods is also warranted.

Here an ***argument*** can be made that the resulting higher level ***policy won't be as unrealistic*** as the one that is completely trained on just torques on a skeleton as eventually it is these muscle actuators and the compliant tendons that generate the torque within their limitation thus it can't go into arbitrary unrealistic values as the lower tier policy will reach a fail state before giving the necessary configuration to generate those torques (given the limitations are implied in the reward structure or stated as an explicit constraints in the optimization)

### 2.1.3 Role of Constraint Optimization on muscle actuation

When muscle actuation is done in general it does pose some limits in a real system where tears can happen over a certain extension (over extension) etc. If a control is to be derived from the reinforcement learning it is paramount for us to have limitations on the range of action permitted for the actuators that can turn this into a constraint RL problem.

When it comes to constrained reinforcement learning I do primarily see the problem as something that needs to be seen in two perspectives. **1.** One is the need to maintain the constraints imposed at all times (for instance scenarios where certain limitations should be observed in order to avoid damage to the robot). **2.** While the other being scenarios where the bounds need to be observed only on the testing phase. When it concerns the latter I believe that any kind of structure that imposes constraint as a subordinate reward would suffice. But when it comes to the former the computational cost of bringing the non optimal feature (action in most cases) to an optimum at every stage and computation of the gradient with regards to that process seems to be higher. (In most cases involves finding the inverse of larger matrices etc) But in general that extra computational cost in the former is unavoidable if we are to never breach the constraints on training even. But there are some ways to reduce the computational cost which can be elaborated as below.

If the constraints are convex in the function space (neural network in our case) we can play a ***two player Lagrangian dual primal game*** and achieve the optimal constraint (achieve a pure Nash equilibrium since the min-max theorem holds) with a first order process who's gradient can be computed similar to what's done in model agnostic meta learning[7]. 1 What's more interesting is the non convex setting where the constraints are not only non convex but also the gradients are not available at all times (due to discontinuity).There are some recent developments in mathematics that do propose a proxy constraints [8] in these non convex settings. It is an open area to explore ***how these proxy constraints can fit into the above mentioned narrative.***

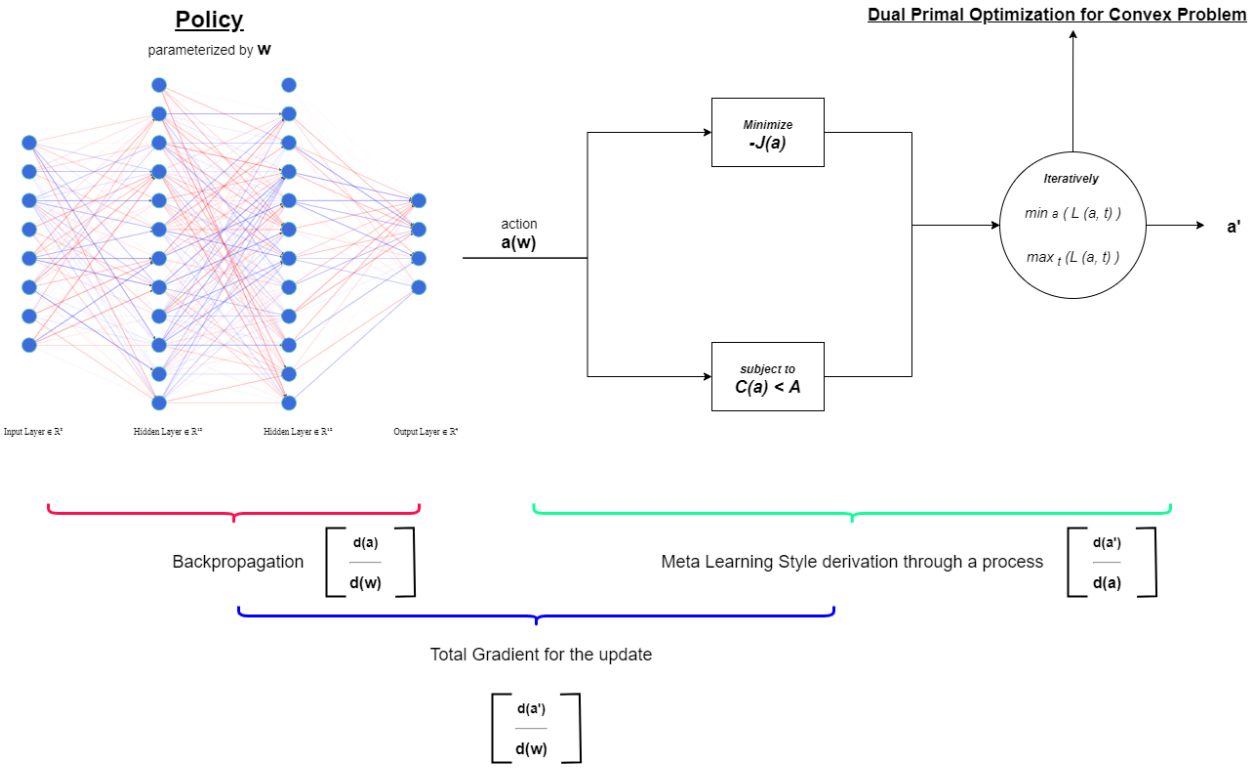


Figure 1: **Constraint Reinforcement Learning** This diagram elaborate the process of imposing constraints on the policy while having a pathway to compute it's gradient with respect to the policy parameter for the update. Here $a$ denotes the action $w$ the neural network parameter, $s$ the state representation, $t$ the dual variable, $J(w)$ the RL objective, $L(a,t)$ Lagrangian of the constraint optimization problem and $C(w), A$ are the constraint functions. For the Dual Primal problem to result in an optimal we do need both the function and the constraint to be convex in terms of the policy space. In this case the return $J(w)$ is already convex in the policy space. Thus this method would result in optimal solution (strong duality / pure Nash Equilibrium) when the constrains are convex. This is where we have to explore on how well can the proxy constraints [8] fit into this narrative so this method can be generalized for non convex (including non differentiable) constraint settings

## 2.2 Control in Higher Level

### 2.2.1 Hierarchical and Inverse RL in higher level control

This is a much more complex scenario where we need both motor level control (how to move) and a higher level control (where to move and at what speed) . This is something that creates an opening in ***hierarchical reinforcement learning*** . Also ***imitation*** can be an essential part when it comes to training these muscular humanoids.

While the imitation can provide better performance in a focused task (with ample of data), in general for to build a system that can be smart about it's exploration in an unique environment with the bootstrapped knowledge from imitation, we need the agent to have some kind of understanding of the sub skills it had learned via hierarchical imitation. Storing this understanding becomes a problem when the number of available sub skills grows. This warrants a need for a compact representation which can in fact be replaced by learning these skills as rewards. This brings us into the problem of ***inverse reinforcement learning***. ***The combined use of hierarchical and inverse RL*** can increase both performance accuracy and computational efficiency of the problem which can be elaborated via an example below.

We can for example take inspiration on how babies learn. For instance when a baby who wants a sweet jar that's on a higher elevation and wants to access it but don't have the necessary skills for it, for example, use a ladder (An example used in "Computational theories of curiosity-driven learning" [9] by Pierre-Yves-Oudeyer on a different context) the baby would see a human behaving in a different context and learns his sub goals (for example, a human using a chair to get something

that's higher) and would try to imitate his action (this imitation process can become more robust if the baby is to learn with the inferred rewards from the human [a higher reward when the elevation increases] rather than just copying them) and finally, try to explore getting to the jar by exploring in lines with the sub tasks learned from the human (using a chair to climb).

This can be a great context to explore human robot collaboration in general. Particularly, as an inverse hierarchical reinforcement framework ( as this can provide the learner with tasks as robust reward function).Also when we talk about either inverse learning or behavioural cloning in a hierarchical framework the hierarchical aspect ***alleviates the complexity issue*** which for the most part acts as a barrier in these fields.

As an extension, if the modelled system is to be used to teach non expert agents (as a learning aide) , this sort of a hierarchical inverse learning framework is something that we need to explore. Because, when it comes to robots teaching humans the scenario generation based on curriculum is essential ( in order to teach humans). While GAN based task generation is something we can pursue with this regard, unless hierarchies are formed the generation process itself can become increasingly complex.

### 2.2.2 Increased conditional stochasticity in the reward structure in the face of hierarchy

The goal of how fast should the humanoid move does pose a contradiction with the reduced metabolic cost reward that is used for the motor control thus now we need to impose a conditional on this reward. That is the low level control has to be modelled as a ***stochastic reward which is conditional on the higher level action*** chosen by the agent. This in itself poses some type of complexity and uniqueness in the hierarchical setting.

### 2.2.3 Complexity of constraints in the face of hierarchy

There is also a unique problem when we talk about constraints in a hierarchical setting that is now the ***constraints have to be passed bottom up***. That is for instance if the higher level control requires a human to perform a certain action at an unusual pace, this cannot be implemented in the lower policy due to the constraints in them (actuator limitation). This does open up feedback from the lower tier to the higher one. Something similar to a ***penalized reward structure*** which would be passed upwards so it can be supplemented with the instantaneous reward the higher tier is supposed to get for an action. Here a simpler penalization may be enough for the higher level policy as the immediate choice of an action (goal for the lower tier policy) would not cause any immediate harm in the motor level so we can afford to skip the strict constraint requirement (first case in the constraint RL section) while avoiding extra computational costs. .

An simpler and effective measure of these penalties would be the measure of ***how much the constraints are breached*** inorder to perform the action. Then again this would create another issue of use needing to allow the agent to breach constraints (second case as mentioned in the Constraint RL section). Though this would be plausible in a stimulation it will not be ideal in a real life setting where the cost of damage is expensive. If not another measure that may come into mind is ***how much does the agent fall short of the goal*** that is requested by the higher tier agent. This in its current form can induce the lower tier to not learn at all as it can now fall short of the goal because it didn't learn and then convey it as a penalty on the agent. So if we are to go down this sort of a penalty then indeed we need to find a balance that would incentivise a lower level learning and convey a penalty without breaching the constraints.

## 3 Research Questions

In this section as a set of research questions the ideas for potential research directions in Human movement modelling are summarized.

1. **What are the ways in which we can create a better mapping from muscle actuator to force ?**

   This can be done using the methods in the paradigm of continual learning or in multi task learning. These methods can either be employed in an embedding function that does this mapping in a supervised manner or on a lower tier reinforcement learning method in an

hierarchical reinforcement learning setting. The latter provides a better structure to the solution. 2.1.2

2. **How can we solve the issue of complexity that would arise when we need to have much more complex control (not just the basic human like movements) with the humanoid ?**

   Hierarchical Reinforcement Learning can in some sense break down the complexity would allow us to reduce the complexity. In addition to the having an inverse RL aspect in the loop can immensely help us when we go into imitation and exploration from the bootstrapped knowledge (reduce the complexity in storage of knowledge). This is discussed in 2.2.1.

3. **What additional complications does the hierarchical framework bring into the equation?**

   Along with an added need for constraints to be passed from the lower level control ( answered in the questions below) it does bring about a stochasticity in the reward structure of the motor level control 2.2.2.

4. **How to we impose level of safety for the humanoid?**

   When it comes to safe Reinforcement learning there are lots of ways to impose them. One such instance is consider the variance in the RL return (too much noise) as a metric and try to minimize or ignore policy samples beyond a variance [10]. Even though these methods does have some promise they wont be as theoretically sound as us trying to explicitly impose the desired constraints based off our physical model. Thus a constraint optimization can be formalized as mentioned in the 2.1.3, Even though the method proposed in it is at it's current form only applicable for convex constraints, the recent works on proxy constraints [8] does open up a room for generalisation.

5. **How does the addition of an explicit constraint in the Hierarchical Reinforcement Learning impact the overall process ?**

   The addition of constraints at the lower level does warrants a need for us to pass the constraints into the higher level control policy in a bottom up manner. This is discussed in 2.2.3.

# References


[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–503, 2016.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013.

[3] L. Kidzinski, S. P. Mohanty, C. F. Ong, Z. Huang, S. Zhou, A. Pechenko, A. Stelmaszczyk, P. Jarosik, M. Pavlov, S. Kolesnikov, S. Plis, Z. Chen, Z. Zhang, J. Chen, J. Shi, Z. Zheng, C. Yuan, Z. Lin, H. Michalewski, P. Milos, B. Osinski, A. Melnik, M. Schilling, H. J. Ritter, S. F. Carroll, J. L. Hicks, S. Levine, M. Salathé, and S. L. Delp, "Learning to run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments," *ArXiv*, vol. abs/1804.00361, 2018.

[4] K. Katsikopoulos and S. Engelbrecht, "Markov decision processes with delays and asynchronous cost collection," *IEEE Transactions on Automatic Control*, vol. 48, no. 4, pp. 568–574, 2003.

[5] S. Ramstedt, Y. Bouteiller, G. Beltrame, C. Pal, and J. Binas, "Reinforcement learning with random delays," 2021.

[6] K. Khetarpal, M. Riemer, I. Rish, and D. Precup, "Towards continual reinforcement learning: A review and perspectives," 2020.

[7]

[8] A. Cotter, H. Jiang, and K. Sridharan, "Two-player games for efficient non-convex constrained optimization," 2018.

[9] P.-Y. Oudeyer, "Computational theories of curiosity-driven learning," 2018.

[10] J. García and F. Fernández, "A comprehensive survey on safe reinforcement learning," vol. 16, p. 1437–1480, jan 2015.

[11] O. Nachum, Y. Chow, B. Dai, and L. Li, "Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections," 2019.

[12] I. Kostrikov, O. Nachum, and J. Tompson, "Imitation learning via off-policy distribution matching," 2019.

[13] M. D. Donsker and S. R. S. Varadhan, "Asymptotic evaluation of certain markov process expectations for large time. iv," *Communications on Pure and Applied Mathematics*, vol. 36, no. 2, pp. 183–212, 1983.

[14] C. Villani, "Optimal transport: Old and new," 2008.

# Appendices

## A Efficient steady state distribution Computation

Here I will shortly summarize the recent developments in the efficient computation of the distance between the steady state distribution induced by a policy and another which can be formulated as just the replay buffer in A.2. In A.4 I will explore the other extensions to it along the same line.

### A.1 Notation

The Reinforcement learning problem is addressed as a policy search problem in a Markov Decision Process(MDP) which is defines by a tuple $(\mathcal{S}, \mathcal{A}, p, r)$.The state space $\mathcal{S}$ and action space $\mathcal{A}$ are assumed to be continuous, and the state transition probability $T : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, \infty)$ represents the probability density of the next state $\mathbf{S}_{t+1} \in \mathcal{S}$ given the current state $\mathbf{s}_t \in \mathcal{S}$ and action $\mathbf{a}_t \in \mathcal{A}$. The environment returns a reward $r : \mathcal{S} \times \mathcal{A} \to [r_{\min}, r_{\max}]$ on each transition. We will use $d_\pi(\mathbf{s}, \mathbf{a})$ to denote the stationary distribution of the Markov chain that is induced by the policy $\pi$. If the Markov chain is ergodic and finite sate then this stationary distribution is unique. Let $p_0(s)$ define the initial state distribution.

### A.2 Distance between stead state stationary distributions

When it comes to learning the steady state distribution between a behaviour agnostic buffer of transitions belonging to a single or multiple of policies $d^D$ and a current policy $d^\pi$ is computed by [11] by solving the problem of

$$\min_{x:S\times A\to C} J_1(x) := \frac{1}{2} E_{(s,a)\sim d^D}\left[x(s,a)^2\right] - E_{(s,a)\sim d^\pi}[x(s,a)] \tag{1}$$

The dependence of the second term on $d^\pi$ which is not accessible was solved via a change of variable as follows.

$$\begin{aligned}
E_{(s,a)\sim d^\pi}[x(s,a)] &= E_{(s,a)\sim d^\pi}\left[\nu(s,a) - \gamma E_{s'\sim T(s,a),a'\sim\pi(s')}\left[\nu\left(s',a'\right)\right]\right] \\
&= (1-\gamma)\sum_{t=0}^{\infty}\gamma^t E_{s\sim p_0(s),a\sim\pi(s)}\left[\nu(s,a) - \gamma E_{s'\sim T(s,a),a'\sim\pi(s')}\left[\nu\left(s',a'\right)\right]\right] \\
&= (1-\gamma)\sum_{t=0}^{\infty}\gamma^t E_{s\sim p_0(s),a\sim\pi(s)}[\nu(s,a)] - (1-\gamma)\sum_{t=0}^{\infty}\gamma^{t+1} E_{s\sim\beta_{t+1},a\sim\pi(s)}[\nu(s,a)] \\
&= (1-\gamma)E_{s\sim\beta,a\sim\pi(s)}[\nu(s,a)].
\end{aligned} \tag{2}$$

By using that they were able to estimate the ratio between steady state distribution $\frac{d^D}{d^\pi}$. [12] expanded upon this and applied the same change of variable on the Donsker-Varadhan definition of KL divergence [13]

$$-D_{\mathrm{KL}}\left(d^{\pi}||d^{\exp}\right) = \min_{x:S\times\mathcal{A}\to R} \log E_{(s,a)\sim d^{\exp}}\left[e^{x(\alpha,a)}\right] - E_{(s,a)\sim d^{\pi}}\left[x(s,a)\right] \tag{3}$$

in order to estimate the KL divergence between $d^D, d^\pi$.

## A.3 Shortcoming

One obvious shortcoming to the aforementioned KL Divergence measure is that it is a divergence. As such it won't hold either the triangle inequality or the symmetry which can be a issue if we are trying to use it as a measure in an extensive manner.

## A.4 Potential Changes

One way to resolve the issue is to explore other metrics rather than a divergence as they by definition would hold the symmetry and the triangle inequality.

### A.4.1 Wasserstein Metric

A metric with a definition 5 similar in form to the Donsker-Varadhan definition is the Wasserstein metric. So using the aforementioned change of variable method we can similarly compute the Wasserstein metric using 5.

Intuitively, wasserstein metric is the measure of the mass that needs to be moved from distribution $\mu$ to $\nu$ in order to make both distributions similar. Wasserstein Metric between two probability distributions $\mu, \nu$ is given as

$$W_1(\mu,\nu) := \inf_{\gamma\in\Gamma(\mu,\nu)} E_{(x,y)\sim\gamma}[\|x-y\|] \tag{4}$$

where $\Gamma(\mu,\nu)$ denotes the set of all joint distributions $\gamma(x,y)$ whose marginals are respectively $\mu,\nu$.By Kantorovich-Rubinstein duality [14] we can write a dual of wasserstein metric as

$$W_1(\mu,\nu) = \sup_{\|f\|_{L\leq 1}} E_{x\sim\mu}[f(x)] - E_{x\sim\nu}[f(x)] \tag{5}$$

Here $\|f\|_{L\leq 1}$ denotes the need for the function $F$ to be a 1-Lipschitz function. An ***added cost*** of this method is the necessity for us to now maintain the neural network 1-Lipschitz.