Summary Report of the Paper 'Deep Reinforcement Learning from Human Preferences'

Jayaram Kuchibhotla
University at Buffalo
jayaramk@buffalo.edu

Reinforcement Learning can be applied to large scale problems which can have a well specified reward functions.

But ,the complex tasks of robots involve goals that are not well defined and hard to communicate.

The complex tasks for robots could be such as 1. Scramble an egg 2.Cleaning up the dirt. The reward function for the goals in these tasks is a function of sensors of the robot which can be specified in an approximate manner

As the reward function is not exactly defined, when in working it can be optimized to get the needed values but the behavior of the robot would not be as expected which is a concern to be addressed by communicating the objectives of the task to it using a reliable method

The exact reward function for a task can be determined by applying 'Inverse reinforcement learning' on the demonstrations of the task generally made by human beings. Apart from that, imitation learning can be used to learn the required behavior from these demonstrations. But, training using these demonstrations can be useful only in the case of humanoid robots in terms of degrees of freedom and not the robots which have a non-human morphology and possess high degrees of freedom.

Human feedback can be used as reward function for training process. This approach is tedious as training systems using Reinforcement learning(RL) require hundreds or thousands of hours. It would be exhaustive for human to sit through this duration and provide feedback even though it is in layman terms. Hence, systems have to be designed in such a way that the duration and amount of human feedback needed is very less.

The team of researchers have tried to apply the basic method of using reward function, designed using human feedback, to deep RL systems which helped them to know about the challenges invlovled. So,they came up with efficient solution, that doesn't need a well specified reward function for sequential decision problems.This solution has the following features:

- It works for training the robots for tasks that do not need or lack the human demonstrations. However,the desired behavior in these tasks have to be recognized by humans and robots correctly
- People who do not have knowledge related to robotics can also train them by using simple instructions which can be understood by any other human and execute the task.
- It can be scaled for large scale problems.
- Minimal feedback in terms of number of instructions and time utilized by human in providing them while training the robot

The algorithm of this solution can be mentioned in the following steps:

- The RL algorithm uses the approximate reward function supplied as input initially.It predicts the action in the environment.
- The action is observed and a feedback is sent to the reward predictor and the RL algorithm
- The RL agorithm uses this feedback as a reward and updates the values and actions for all the states in every iteration towards forming an optimal policy
- The different behaviors of the agent recorded by video clips are observed by the human and compared to provide a score which is the human provided feedback.

The reward predictor uses the feedback provided by the environment along with the above mentioned feedback to predict a reward function which will be used in the first step for the next iteration.

The above mentioned steps are repeated to obtain an optimal policy and an optimized reward function. The feedback provided by humans while training helps to achieve the goal of the task by the robot which was otherwise difficult when only approximate reward function was used.

The researchers have conducted experiments in two domains :
-Atari games in the Arcade learning environment
-Robotics tasks in the physics simulator Mujoco

They have used the human feedback ranging from 15 minutes to five hours for various RL tasks which do not have clear reward function.For the tasks such as driving with the flow and back flipping ,an hour of human feedback is used to achieve the goal

They have analyzed work related to human feedback from 3 categories :
-Absolute score values are provided as input for reinforcement learning problems
-Relative scores as input for reinforcement learning problems
-Usage of relative scores for problems belong to non-reinforcement learning category

Comparison between the present work and others' research

Important Differences

S.No	Work done in the present research paper	Work by Akrour et al. (2012) and Akrour et al. (2014)
1	Reward Function is linearly dependent on features which could be of the form w1*E[f1] + w2*E[f2] +w3*E[f3] + ...wn*E[fn] where E[f1] ,E[f2]...E[fn] are expectations of the features. Features are extracted by deep learning based methods	Reward Function is same as mentioned in the left cell. But, the features in the reward function are extracted using the hand coded method
2	Multiple degrees of freedom	Four degrees of freedom
3	Human feedback is provided for short clips of the trajectory motions	Feedback is provided for the whole trajectory motion of movement of robot's parts intended for the task

Similarities :
-Both of them use continuous and small discrete domains.
The parts of the robots move in a trajectory for the intended task . This motion can be be either
Discrete : The motion is interspersed with brief intervals of movements and pauses
Continuous : The motion is free from brief pauses

Other differences :

-If 'n' is the number of comparisons for human feedback is provided in the other work and x is the unit magnitude of such comparisons, then the number of comparisons for the present work is n +2x
-If 't' is time required for training in the other work, the training duration is 2t in the present work
-Both the approaches use different RL algorithms ,different reward functions
-Ensembling of models and asynchronous training is used in the present work

Asynchronous training : This could be the following

Consider The two events a. Viewing video clips and providing numerical score as feedback after comparison and b. Optimization of existing reward function based on the feedback, action of robot based on the new reward function 'a' and 'b' could not be happening one after the other synchronously.

Rather, a priority queue could have been maintained to collect the numerical scores, high score could mean high priority optimization and correction of the action. All the numerical scores, one for each video clip comparison, are inserted into priority queue and delivered as asynchronous events to the reward predictor. Before the reward predictor can get a new score and optimize the reward function, the agent continues to act on the old reward function. It does not need to wait to act for the human to provide the feedback

Following could have been done by Wilson et al. (2012)

-They use the same feedback mechanism as used in the present work i.e. providing discrete numerical scores within an interval of values
-The reward function is the distance (could be either euclidean ,manhattan or Frobenius norm) between function representing the current policy and the function for target policy(set of states and actions to achieve the goal) .
-Synthetic feedback drawn from the Bayesian network is used instead of human feedback
-Reward function is a linear combination of weights and features.It can be given as w1f1 + w2f2 + ...wnfn
-Bayesian inference,which uses feedback,is applied to fit the reward function i.e to calculate the coefficents w1 ,w2 .....w3
-By using the optimized reward function ,the current policy or esitmate of target policy is made using the Maximum a posteriori (MAP) technique rather than RL algorithm

MacGlashan et al. (2017), Pilarski et al. (2011)

-They use reinforcement learning algorithms to predict the target policy ,but consider only human feedback to optimize the reward function but not the feedback from the environment. This approach is not feasible for the present work since the agent is trained for lot of hours in environments such as Atari games

Knox and Stone (2009), and Knox (2012)

-They also use RL algorithms but consider simpler settings where a policy can be learnt quickly. The present work has relatively more complex settings

Hadfield-Menell et al.(2016)

-They use inverse reinforcement learning to learn the reward function. The robot acts as per the optimized reward function in the environment. Human interacts with the environment after the robot has made an action, to optimize the reward function
-The present work is similar to this work except that human interaction is only by means of giving feedback through numerical scores

The team of researchers have studied the other works where the reward function is learnt using deep learning to train the agent in acquiring more complex behaviors. In the present work, they scaled human feedback and used the above technique which is on the lines of deep reinforcement learning

Method followed:

In traditional reinforcement learning, the agent receives observations from the environment and acts. For the action made, it receives reward from environment.

There are two possible scenarios for the above activity to occur:

1. An agent acting in the video game or robot simulation, where the environment is divided into a grid of positive states (agent could get bonus points which is a positive reward) and negative states (negative reward such as end of game). The agent moves to the future states (acts) which are towards the goal based on the values of those states (negative states have low values and positive states have high values). Agent can access a limited number of future state values surrounding the current state. Getting information about the future states is acquiring observations from the environment.

2. A robot or real word robot acting in the real world. Observations are the data gathered by its proximity sensors about the goal in the environment. Action is the torque exhibited in the arms of the robot to produce the needed motion. If the motion is towards achieving the goal, the agent acquires the positive reward. Otherwise, it receives a negative reward. Again, the proximity sensors help to get this information.

A 'Trajectory' is the set of observations and actions that are made from the start state to goal state. The subset of observations and actions is called 'Trajectory segment' which can be given as

$\sigma = ((0_0, a_0), (0_1, a_1), \dots, (0_{k-1}, a_{k-1})) \in (\mathcal{O} \times \mathcal{A})^k$.

In the current work, apart from using the reward signal from the environment, the score provided by the human observer is also used as feedback. These feedback values are used to optimize the reward function and to determine the optimal policy

$\sigma^1 \succ \sigma^2$ indicates that the human observer has preferred trajectory segment σ^1 to trajectory segment σ^2. Eventually, the agent has to determine a trajectory, which is as expected by human, with as minimum feedback as possible.

Evaluation of algorithms:

Qualitative :

In this approach a reward function, which is external to the RL algorithm being used, is applied to compare the performance of two trajectory segments .

We say that preferences $\succ$ are generated by a reward function $r : \mathcal{O} \times \mathcal{A} \to \mathbb{R}$ if

$$((0_0^1, a_0^1), \dots, (0_{k-1}^1, a_{k-1}^1)) \succ ((0_0^2, a_0^2), \dots, (0_{k-1}^2, a_{k-1}^2))$$

whenever

$$r(0_0^1, a_0^1) + \cdots + r(0_{k-1}^1, a_{k-1}^1) > r(0_0^2, a_0^2) + \cdots + r(0_{k-1}^2, a_{k-1}^2) .$$

This feedback is called synthetic feedback, which shall explained in detail further.

Quantitative:

Reward Function is not used in this approach. Rather, the goal of the sub task in the trajectory segment described using natural language and the video clip of the trajectory segment are shown to the human observer to provide a score based on how well the goal is accomplished

This feedback is called human feedback, which shall also be explained later

The method of comparing trajectory segments is similar to that used by the work in Wilson et.al (2012). After achieving the goal in the current trajectory segment and proceeding to the next one, the state values in the latter work are reset to arbitrary values to introduce randomness in achieving the goal and reward maximization. In the present work, these states are reset with default values.

For learning the reward function efficiently, the state values are reset in both the cases.

Resetting the state values can cause a problem where in the trajectory segments of two different training instances for comparison look similar to the human observer. Actually, they differ in accomplishing the goal –one might be lot better or worse than the other. The observer would give similar score values which do not differ a lot .Hence the situation is not well assessed. The algorithm used in the present work could compensate for this error

The Method

'Policy' is a function which takes 'state' or 'observation' as input and outputs 'action' as output parameter which can be represented as $\pi : \mathcal{O} \rightarrow \mathcal{A}$. The reward function estimate is denoted as $\hat{r} : \mathcal{O} \times \mathcal{A} \rightarrow \mathbb{R}$. Both of them are implemented using a deep neural network.

These two neural networks are updated by the following processes:

1. The policy network predicts an action for an observation. The set of actions form a trajectory and for multiple training iterations we get a set of trajectories $\{\tau^1, \ldots, \tau^i\}$, one for each iteration . To optimize the policy, the parameters of the policy network are updated by reinforcement learning algorithm. The predicted reward values, one for each segment $r_t = \hat{r}(0_t, a_t)$ in this trajectory, are added to get a value. This value is maximum among values calculated for all other trajectories in a similar manner

2. From the above mentioned set of trajectories, two of them are considered. A corresponding trajectory segment one from each of these two trajectories ,form a pair which is denoted by (σ^1, σ^2). This pair is given to human for comparison and providing feedback

3. The parameters of the second neural network, which implement the reward function are optimized to minimize the cross entropy loss between the predictions made by the reward function and the human labels for the trajectory segment comparison mentioned in step2

Output of process1 - set of trajectories is used as input in process2.
Output of process2 - human feedback is used as input in process3
Output of process3 - optimized reward function is used as input in process1

However, all these processes are asynchronous. The above processes execute for multiple times to get an optimal policy and best reward function which would help in getting an optimal trajectory.

Optimizing the Policy

As mentioned in step1 , a RL algorithm is used to optimize the policy .In the current method, the reward function is optimized by human feedback .Hence it is not-stationary. In such a scenario, the policy gradients methods are best suited .

For Robotic simulation tasks, TRPO (Trust Region Policy Optimization) method is used,
while AAC (Advantage Actor Critic) method is used for Atari games

Trust Region Policy Optimization (TRPO):

In task or path planning, trajectory starts from start state to goal state with intermediate states. As the policy is optimized, the trajectory changes, thus changing the visitation frequency of each state. Due to a bad update, an effective policy is overwritten by an ineffective policy and a good update improves the existing policy. The expected sum of discounted rewards $\eta(\pi)$ is a measure of effectiveness of a policy, it is higher for a better policy and low for a not so good policy.

TRPO is a method, which guarantees monotonic improvement for a stochastic policy. $\eta(\pi)$ is monotonic which means it is neither completely non-increasing nor non-decreasing.

It can be explained further as below:
If $(S, \mathcal{A}, P, r, \rho_0, \gamma)$ is a infinite horizon discounted Markov Decision Process (MDP) where
S is a finite set of states
$\mathcal{A}$ is a finite set of actions ,
P : S × A × S → R is the transition probability distribution.
$r : S \to \mathbb{R}$ is the reward function,
$\rho_0 : S \to \mathbb{R}$ is the distribution of the initial state s_0
$\gamma \in (0,1)$ is the discount factor.

Let π denote a stochastic policy $\pi : S \times \mathcal{A} \to [0, 1]$,

A deterministic policy outputs action without associating a probability value. For e.g. If there is a state which has 4 Q values for 4 actions , then the policy outputs an action which has the maximum Q value.

A stochastic policy is usually a function which takes state value as input and is implemented by a neural network .It predicts actions with probability values ranging between [0,1] .

$\eta(\pi)$ denote its expected discounted reward:

$\eta(\pi) = E_{s_0, a_0, \ldots} [\sum_{t=0}^{\infty} \gamma^t r(s_t)]$, where

The term $\sum_{t=0}^{\infty} \gamma^t r(s_t)]$ is discounted sum of rewards for any given state computed over multiple instances of time , is represented by a continuous random variable as it can be any value in the set of real numbers.

The expectation is calculated using this as random variable and the joint probability density function which takes all the state and action variables $s_0, a_0 \ldots\ldots$ in the sets S and $\mathcal{A}$ respectively

$s_0 \sim \rho_0(s0)$, $a_t \sim \pi(a_t|s_t)$, $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$

Above 3 notations represent the following in order :

1.Initial state s_0 is sampled from the probability distribution of initial states $\rho_0(s0)$,
2. The action is a_t is sampled from stochastic policy function or network and
3. Probability of future value of the same state s_{t+1} is sampled from the transition probability distribution using the current state and action s_t, a_t values as input parameters

Following are the standard definitions of the state - action value function Q_π, the value function V_π, and the advantage function A_π:

$Q_\pi(s_t, a_t) = E_{s_{t+1}, a_{t+1}, \dots}[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l})]$

$V_\pi(s_t) = E_{a_t, s_{t+1}, \dots}[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l})]$,

Both $Q_\pi(s_t, a_t)$and $V_\pi(s_t)$ are calculated by using the discounted sum of rewards over multiple instances of time for a single state as a continuous random variable.

In $Q_\pi(s_t, a_t)$ for the current state and action pair s_t, a_t , the expectation value is calculated using the joint probability density function which takes the future values (the succeeding training instances or trajectories) of the same state and action pair. This way of calculating the expectation value is same for $V_\pi(s_t)$ except that the current action value a_t is also considered as input parameter for the joint probability density function

The advantage function $A_\pi(s, a)$ can be given as $A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)$

The value $V_\pi(s)$ is calculated in two ways:

1. Using the expectation of discounted sum of rewards as mentioned in the above equation
2. The Q_π values for every action are calculated and the maximum among them is the V_π value

Instead of using the V_π values in the above said two methods to find the difference, the Q_π values in the second method and the V_π values in the first method are used . The objective of finding this difference is to compare the utility values computed by these two methods.

The below identity gives the expected value of discounted sum of rewards $\eta(\tilde{\pi})$ for policy $\tilde{\pi}$ in terms of the advantage over π, accumulated over time steps (introduced by Kakade & Langford (2002))

$\eta(\tilde{\pi}) = \eta(\pi) + E_{s_0, a_0, \dots \sim \tilde{\pi}}[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t)]$ (1)

$\eta(\pi)$ is the expected value of discounted rewards for policy π
The policy $\tilde{\pi}$ is optimized over π.
It can be observed in the above identity that advantage A_π is calculated using states and actions from policy π , while the expectation is calculated by using the density function which takes the states and actions as input parameters , these are sampled from policy $\tilde{\pi}$.

Using the identities

$E_{XY}(RV) = \sum_x P(X) \sum_y P(\frac{Y}{X}) RV$

X Y are the probability distributions over which expectation is calculated for the Random Variable

And $\mathrm{E}\left(\sum_{i=0}^{N} X_i\right) = \sum_{i=0}^{N} E(X_i)$

If X is considered to be Random Variable s that can take the states $s_0, s_1 \ldots . s_n$ and
Y is a that can take actions $a_0, a_1 \ldots \ldots . a_n$ then
Equation (1) can be written as below

$$\eta(\tilde{\pi}) = \eta(\pi) + \sum_{t=0}^{\infty} \sum_s P(s_t = s|\tilde{\pi}) \sum_a \tilde{\pi}(a|s) \gamma^t A_\pi(s,a)$$
$$= \eta(\pi) + \sum_s \sum_{t=0}^{\infty} \gamma^t P(s_t = s|\tilde{\pi}) \sum_a \tilde{\pi}(a|s) A_\pi(s,a)$$

If ρ_π is the (unnormalized) discounted visitation frequencies

$$\rho_\pi(s) = P(s_0 = s) + \gamma P(s_1 = s) + \gamma^2 P(s_2 = s) + \cdots,$$

Here $s_0, s_1 \ldots . s_n$ indicate the values of a single state at multiple training instances t= 0,1,2, ……. ∞ . This is different from multiple states $s_0, s_1 \ldots . s_n$ mentioned above. In reality, the training instances are limited. But here ∞ is considered as these calculations are being done for infinite horizon Markov Decision Process(MDP) .

then

$$= \eta(\pi) + \sum_s \rho_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a|s) A_\pi(s,a) . \quad (2)$$

In the above expression, if the below term

$\sum_a \tilde{\pi}(a|s) A_\pi(s,a)$ >0, then the performance of policy $\tilde{\pi}$ improves compared to π, as $\eta(\tilde{\pi})$ increases.
= 0, it remains the same as $\eta(\tilde{\pi})$ would not change in this case
< 0, performance of policy degrades as $\eta(\tilde{\pi})$ decreases

$\rho_{\tilde{\pi}}(s)$ is dependent on $\tilde{\pi}$ which makes Equation (2] difficult to optimize directly (The state visitation frequencies of a policy would be updated only after the present policy is updated). Instead, the following local approximation to discounted sum of rewards η is introduced:

$$L_\pi(\tilde{\pi}) = \eta(\pi) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A_\pi(s,a) . \quad (3)$$

It can be noticed that this approximation is dependent on ρ_π instead of $\rho_{\tilde{\pi}}$.

If there is a parameterized policy π_θ, where $\pi_\theta(a|s)$ is a differentiable function of the parameter vector θ, then L_π matches η to first order (see Kakade & Langford (2002]). That is, for any parameter value θ_0,

$$L_{\pi_{\theta_0}}(\pi_{\theta_0}) = \eta(\pi_{\theta_0}) ,$$

$$\nabla_\theta L_{\pi_{\theta_0}}(\pi_\theta)|_{\theta=\theta_0} = \nabla_\theta \eta(\pi_\theta)|_{\theta=\theta_0} . \quad (4)$$

Equation (4·] implies that a sufficiently small step $\pi_{\theta_0} \rightarrow \tilde{\pi}$ (slight improvement in the policy) that improves $L_{\pi_{\theta_0}}$ will also improve η, but it does not tell clearly about the step size.

To solve the above mentioned problem, a policy updating scheme called 'conservative policy update' is proposed by , Kakade & Langford in 2002 , it can be given by the following equation :

$$\pi_{new}(a|s) = (1-\alpha)\pi_{old}(a|s) + \alpha\pi'(a|s) \quad (5)$$

where

π_{new} is the new policy,
π_{old} is the current policy
π' is the best among the π_{new} policies
α is given by $\sum_a |\pi_{new}(s,a) - \pi_{old}(s,a)|$
$\pi_{new}(s,a)$ is the joint probability value between state 's' and action 'a' for new policy
$\pi_{old}(s,a)$ is the joint probability value between state 's' and action 'a' for old policy

The above policy update has to be done so as the right side of the below bound is maximized as much as possible

$$\eta(\pi_{new}) \geq L_{\pi_{old}}(\pi_{new}) - \frac{2\varepsilon\gamma}{(1-\gamma)^2}\alpha^2$$

so as approximate expected value of discounted sum of rewards is closer to the expected value of discounted sum of rewards $\varepsilon = \max |E_{a\sim\pi'(a|s)}[A_\pi(s,a)]|$,the maximum among the expected values calculated by using advantage as a random variable and by using the density function which takes action 'a' as a parameter that is sampled from π' which is the best of all the policies (which has the maximum discounted sum of rewards)

Total variation divergence between two discrete probability distributions p and q can be given by

$$D_{TV}(p\|q) = \frac{1}{2}\sum |p_i - q_i|$$

Maximum total variation divergence between 2 different policies π_{old} and π_{new} , is given by

$$D_{TV}^{\max}(\pi_{old}, \pi_{new}) = max_s \frac{1}{2}\sum_a |\pi_{old}(a|s) - \pi_{new}(a|s)|$$

s represents every possible state and a represents every possible action

If α is substituted by maximum total variation divergence between old policy and new policy instead of just the absolute difference, then the bound would be as below

$$\eta(\pi_{new}) \geq L_{\pi_{old}}(\pi_{new}) - \frac{4\varepsilon\gamma}{(1-\gamma)^2}\alpha^2 \text{ where } \varepsilon = max_{s,a}|A_\pi(s,a)|$$

In the above equation α can be calculated using KL divergence instead of total variation divergence
As mentioned in equation (5), the policy update is done for multiple iterations, for each iteration the right hand side of above mentioned either of the conditions or bounds is maximized which makes $\eta(\pi_{new})$ non-decreasing.

Initially π_{old} , π' , π_{new} both are initialized with value π_0 which is a policy with default random parameters
for each iteration until convergence i.e. $\eta(\pi_{new})$ does not decrease
$\pi_{new}(a|s) = (1-\alpha)\pi_{old}(a|s) + \alpha\pi'(a|s)$
$\pi' = \arg max_{\pi_{new}} [L_{\pi_{old}}(\pi_{new}) - C\alpha]$ $where\ \alpha = D_{KL}^{\max}(\pi_{old}, \pi_{new})$
(π_{old} is π_{new} in previous iteration , C = $\frac{4\varepsilon\gamma}{(1-\gamma)^2}$)
$\eta(\pi_{new})$ is calculated and compared with $\eta(\pi_{old})$
to check loop terminating condition

The above mentioned procedure is theoretical algorithm for TRPO ,for practical implementation following optimizations are needed

1. If $\alpha = D_{KL}^{\max}(\pi_{old}, \pi_{new})$ and C = $\frac{4\varepsilon\gamma}{(1-\gamma)^2}$, then the policy updates will happen in small steps.
But, if α is considered in such a way that it is always less than or equal to δ and C >= $\frac{4\varepsilon\gamma}{(1-\gamma)^2}$ then the updates will be in larger steps

2. Due to large number of constraints , maximum KL divergence is substituted by average KL divergence which can be given by $\overline{D}(\theta_1, \theta_2) := E_{s\sim\rho}[D_{KL}(\pi_{\theta_1}(\cdot\,|s) \,\|\pi_{\theta_2}(\cdot\,|s))]$.

3. Instead of constraint based optimization, sample based estimation is used for $L_{\pi_{old}}(\pi_{new})$. Let $\theta_{old} = \pi_{old}$ $\theta = \pi_{new}$
The surrogate objective function is maximized
$maximize_\theta\ L_\theta(\theta_{old}) = maximize_\theta\ (\eta(\theta) + \sum_s \rho_{\theta_{old}}(s) \sum_a \pi_\theta(a|s) A_{\theta_{old}}(s,a))$
subject to $\overline{D}(\theta_1, \theta_2) <= \delta$

In the above equation for the second half part on the RHS : $maximize_\theta\ (\sum_s \rho_{\theta_{old}}(s) \sum_a \pi_\theta(a|s) A_{\theta_{old}}(s,a))$,three changes are made
1. $\sum_s \rho_{\theta_{old}}(s)\,[\ldots]$ is replaced by the expectation $\frac{1}{1-\gamma} E_{s\sim\rho_{\theta_{old}}}[\ldots]$.
2. The advantage values $A_{\theta_{old}}$ are replaced by the Q-values $Q_{\theta_{old}}$
3. The sum over the actions is replaced by an importance sampling estimator. q denotes the sampling distribution.

Importance sampling:

If f(x) is a PDF, then E(f(X) = $\int f(X)P(X)dx \cong \lim_{N\to\infty} \frac{1}{N}\sum_{x\sim P} f(X)$ .If it is not possible to draw samples from the distribution P ,then the samples can be drawn from another distribution Q and calculate the approximate Expectation value as below

E (f(X) $\cong \lim_{N\to\infty} \frac{1}{N}\sum_{x\sim Q} \frac{P(X)}{Q(X)} f(X)$ such that if Q(X) = 0 ,then P(X) = 0

After these changes the expression changes as below

$maximize_\theta\ E_{\ s\sim\rho_{\theta_{old}},\ a\sim q}[\frac{\pi_\theta(a|s)}{q(a|s)} Q_{\theta_{old}}(s,a)]$ subject to $E_{s\sim\rho_{\theta_{old}}}[D_{KL}(\pi_{\theta_{old}}(\cdot\,|s) \,\|\pi_\theta(\cdot\,|s))] \le \delta$. (Eqn. A)

This substitution helps in practical implementation and faster execution

The above sampling estimate is made in two ways:

1. Single Path:
The states are collected by sampling initial state distribution $s_0 \sim \rho_0$. The policy $\pi_{\theta_{old}}$ is used for some number of time steps to generate a trajectory $s_0, a_0, s_1, a_1, \ldots, s_{T-1}, a_{T-1}, s_T$. $\pi_{\theta_{old}}$ represents the sampling distribution $q(a|s)$ which is used for sampling out the actions. The sampled states are used in $\pi_\theta(a|s)$, sampled states and actions are used to calculate the Q-values $Q_{\theta_{old}}$.

2. Vine:
In this method, the states are sampled $s_0 \sim \rho_0$. A set of trajectories are generated using the policy π_{θ_t}. From these trajectories a set of states are collected which is called a roll out set (set of branched short trajectories).
K actions are sampled for each state in this roll out set as $a_{n,k} \sim q(\cdot\,|s_n)$.
The sampling distribution $q(\cdot\,|s_n)$
$= \pi_{\theta_i}(\cdot\,|s_n)$ works well on continuous problems, such as robotic locomotion .
= uniform distribution which works well on discrete tasks, such as the Atari games .

$\hat{Q}(s_n, a_{n,k})$ is calculated for each action $a_{n,k}$ sampled at each state s_n.

TRPO Summary:
1.One of the methods 'Single Path' or 'Vine' is to be used to collect the state action pairs and estimate the Q values using Monte Carlo estimation.
2. Use the estimated Q-values ,sampled state and action values in Eqn.A apart from the KL-divergence constraint.
3. The objective function (Eqn. A is part of it) is optimized to get the policy's parameter vector θ. The authors use the conjugate gradient algorithm along with line search.

Theoretical explanation of Advantage Actor Critic (A2C) Algorithm:

An agent has to take actions and move across the states from start state to goal state in such a way that the total reward sum is maximized.
In A2C , there is a single neural network that takes state as input and output the following respectively :
1. The action to be taken by the agent to move to the next state – when this is role is played by the agent, it is called actor.
2. The state value which is the estimate of total rewards that can be acquired from the present state to the goal state – when the agent makes such an estimation it is called critic

For every state , the agent maintains five values
1. Estimation of state value
2. Action taken
3. Reward received
4. Actual State value
5. Error, the difference between the actual state value and estimation of the state value

The above data can be imagined as table which contains five columns and one row corresponds to one state

The agent fills the first three values for every state transition while the last two values are filled for multiple visited states when it makes a reflection

The rewards are discounted to get the average sum of rewards while policy gradients are not discounted so as the agent does not need to move fast during the initial states or slow during the ending states , it has to move at uniform speed . This is typically the case when the agent has to navigate along long paths.

In A2C, several agents are trained to make batches of observations (e.g. 3 in a batch) in parallel ,all the agents use a single shared neural network –the predictor of a state value .This is done to prevent the autocorrelation between the error values of observations. The presence of autocorrelation leads to incorrect prediction of the value of a state.

For a given input state, the policy network predicts all the possible actions with confidence values, the action with high confidence value is chosen. Instead of the above method ,action can also be sampled from the probability distribution, this action could be a mediocre and not the best always. This method might help to maximize the total sum of rewards. The prediction of actions is not deterministic but probabilistic .

Actions can be sampled from the action distribution, if the standard deviation between the probability values of samples is high ,then the entropy is low and choice of an action from these samples is discouraged .Otherwise, if the standard deviation is low ,then the entropy is high and choice of an action from these samples is encouraged.

The true state value is computed while reflection by the agent. It uses the preceding estimated state values. The difference between true value and estimate value of the state is called the Error or Advantage which is used by both actor and critic.

The total loss used to train the neural network
Total loss = Action Loss + Value Loss - Entropy

Value loss is the Error or Advantage.
Action loss is the difference between the predicted action and the actual action in the ground truth training sample

Thus, a single deep learning model is used by agent (which acts as both actor and critic) to predict the state values and action for a given state input using the advantage, eventually maximizing the total sum of rewards

In the present paper, TRPO is used for robotics task and A2C was used for Atari games.

In TRPO, to determine the policy within the trust region ,adequate exploration has to be done. This is not possible as the reward function is changing. To compensate this problem , the parameter 'Entropy bonus' is adjusted. This parameter is usually added with the reward function

Preference Elicitation:

The human observer is shown 2 clips of 1 or 2 seconds long each containing the video of trajectory segment. The observer has to make a comparison, this comparison output is stored in the database in the following format
$(\sigma^1, \sigma^2, \mu)$
σ^1 and σ^2 are the labels of two segments
μ is a rectangular distribution over {1, 2}
If σ^1 is preferred over σ^2 , then probability value at 1 is higher than probability value at 2 and vice versa for the other case
If both are equally preferred then μ is a rectangular distribution.
If σ^1 and σ^2 are not comparable ,then the output is not recorded in the database

Fitting the reward Function:

The preferences of trajectory segments can also be made using the reward function. There are 2 outputs for a classifier i.e. if preference $\sigma^1 > \sigma^2$ and $\sigma^2 > \sigma^1$. The softmax function computation for these outputs can be given as follows

$$\hat{P}[\sigma^1 \succ \sigma^2] = \frac{\exp \sum \hat{r}(0_t^1, a_t^1)}{\exp \sum \hat{r}(0_t^1, a_t^1) + \exp \sum \hat{r}(0_t^2, a_t^2)}. \qquad \hat{P}[\sigma^2 \succ \sigma^1] = \frac{\exp \sum \hat{r}(0_t^2, a_t^2)}{\exp \sum \hat{r}(0_t^1, a_t^1) + \exp \sum \hat{r}(0_t^2, a_t^2)}.$$

In the above section, if the human observations for the 2 choices are considered as ground truth and above two values are predictions of a classifier, then the cross entropy loss between the predictions and this data can be given as :

$$\text{Loss } (\hat{r}) = -\sum_{(\sigma^1, \sigma^2, \mu) \in D} \mu\,(1) \log \hat{P}[\sigma^1 \succ \sigma^2] + \mu(2) \log \hat{P}[\sigma^2 \succ \sigma^1].$$

Further changes made to the above work:

1. There are multiple ('D' number of) human comparisons made for two trajectories and 'D' number of predictors (ensemble of deep neural networks) which implement the reward functions separately are used to predict the preferences. Loss functions are computed separately and the parameters of the respective models are optimized.

After training, all the parameters of deep learning models in the ensemble are averaged to give final set of parameters for the reward function estimate

2. 1/e of the total data is used for validation for each of the model in the ensemble. L2 regularization is used and the regularization coefficient is adjusted such that validation loss is between 1.1 times to 1.5 times of the training loss. Drop out can also be used for regularization.

3. As human observers can make errors in making the comparison of trajectory segments, to compensate that the softmax function is used with adjustment.

Selecting Queries:

Each model in the ensemble approximates reward function estimate, are trained using the human observations and the predictions. After this training process is complete, the new pairs of trajectory segments are fed as input (one pair at one time) to each model to get prediction of preferences. The pair, which has high variance in predicted preference output, is selected for human feedback.

Experimental Results:

The algorithm is implemented in Tensor Flow. The environment MuJoco (Mutli Joint Dynamics with contact) is used for Robotics while Arcade Learning Environment is used for the Atari games .Both of these are interfaced through Open Ai Gym

Reinforcement Learning Tasks with Unobserved Rewards:

There is no concrete reward function beforehand to calculate the rewards. It is approximated using the human feedback. The goal is to input the queries to human observer which are as minimum as possible.

Method to obtain human feedback:
1. One to two sentence description of the task is provided to human observers to provide feedback.
2. Then the observers are shown 1-2 seconds long video clips of the trajectory segments corresponding to the task mentioned in point 1.
3. The response is provided by the observers within a time gap of 3-5 seconds.

4. In the above said manner, several hundreds to thousands of trajectory segments are shown thereby incurring a total human time ranging between 30 minutes to 5 hours.

For Comparison following experiments are done in addition :
1. A synthetic reward function is used to predict the preferences of trajectory segments instead of human feedback.
2. Use a pre-available reward function to provide rewards for the trajectory segments
The aim of the work in the present paper is to shape the reward function, without actually accessing a pre-available reward function, using only a little human feedback. The reward function shaped this way can always outperform the reward function mentioned in Point2

Simulated Robotics:

There are 8 robotics (walker, swimmer, hopper, cheetah, ant, reacher, double pendulum, pendulum) tasks that were used with modifications. Originally, the session would end if the agent falls or few parameters go out of control. After modifications for the present work , the agent would be given a penalty.

The reward function used in this tasks is a linear function of distances, velocities and positions and quadratic in features.

For comparison, the cart pole(inverted pendulum task where the pole has to balanced while moving the cart) task is also performed

With 700 queries to human observers, the performance of the algorithm where the reward function is optimized using feedback is almost equal to that of the RL algorithm where a reward function is already present. Also, in the former case, the variance in the output was high

With 1400 queries the performance of the algorithm is greater than the RL algorithm, the reward function is better shaped due to large number of queries.

Human feedback method is slightly less effective than the method where synthetic feedback is used. On the 'Ant' task, the human feedback is effective than synthetic feedback because the observers were told to prefer those segments higher where the agent is in standing position (the 'Ant' agent has to be in upright position in this task) and the shaping of the reward function formed better.

Similarly in the RL algorithm, bonus parameter - if the 'Ant' agent was upright this parameter would help to get higher reward, is used in the reward function but this was not much useful

Atari Games:

In this task, 7 Atari games (beamrider, breakout, pong, qbert, seaquest, spaceinvaders, enduro) were considered in the Arcade Learning Environment . The comparisons are made between the cases, 5500 human queries, 350 ,700 or 1400 synthetic queries and the RL algorithm with real award.
The chosen trajectory segment pairs with high variance in predicted output:
When given as input to the synthetic reward function they are called Synthetic Queries.
When rated by humans they are called Human Queries.

1. The algorithm in the present in the work could not match the RL algorithm method except in few cases .In few other cases the performance of the former exceeds the latter.
2. In Beam Rider and Pong games, when 3300 synthetic queries are used to shape the reward function, the performance comes closer to the RL algorithm.
3. In Seaquest and Qbert games, synthetic feedback performs close to the level of RL but learns the reward function slowly
4. In Space Invaders and breakout games, synthetic feedback never matches RL but the agent crosses the first level in space invaders always while in Breakout ,the score is 20 or 50 with enough queries

5. In these games, the performance of human feedback based method is equal or slightly worse than the synthetic feedback based method with same number of queries for both. Most of the times, synthetic feedback queries are 40% lesser than human feedback queries. The reasons for the difference in performance are :

a. Incorrect labelling (comparison) of trajectory segments by human observers.
b.Ideally, all trajectory segments have to be labelled by all the observers. But, in practical situation, few of them would have labelled by variable number of observers thereby causing uneven rate of labelling.
c. In a trajectory, few segments are labelled by the observers many times while others are scarcely labelled.
These problems could be solved by improving processes in labelling.

In Q-bert game, the human feedback did not help to go through even the first level of the game. The difficulty in evaluation of the short clips of this game videos was responsible for this problem

Enduro game was difficult to learn by both:
a. The RL method that uses the A3C algorithm and b. Synthetic Feedback method .
Interestingly, the human feedback method learns better than the above 2 methods and this method surpasses the A3C method in performance.

Novel behaviors:

The human feedback method could learn new complex behaviors such as:

1. The Hopper robot was trained to do backflip, land upright and repeat these steps using 900 queries in less than an hour.

2. With 800 queries in an hour, the Half Cheetah robot was trained to move forward standing on just one leg.

3. In the Enduro game, for another specific task where the car has to be alongside the other cars, the agent (car) has been trained using 1300 human feedback queries, which are selected from 4 million frames of game videos. The car was alongside other cars for most of the time. It got confused in the instances where the background of the game changed.

Ablation studies:

This study involves removing or replacing certain important methods of the algorithm and testing those changes.

1. As mentioned earlier, the queries, which have high variance in the output predicted by the reward function under shaping, are given to human observers for providing the feedback. Instead of this method, the queries are selected randomly and given as input to the observers.

2. Instead of ensemble of reward predictors, there is only one reward predictor used along with random queries.

3. Training is done only using the human feedback queries only in the beginning of the training and not though the entire duration. (offline reward function estimate training)

4. Dropout is used instead of L2 regularization.

5. Trajectories of unit length (i.e. which have only one trajectory segment) are used for robotics tasks only.

6. Instead of using the human feedback for fitting the reward function estimate, the reward predictions from an oracle(feedback from relatively highly skilled human observer) are used. Also, instead of using the cross entropy loss, mean squared loss between the predicted reward values and true reward values by the oracle is used for training.

In the offline reward predictor training method, due to the stationarity (mean, autocorrelation ,variance parameters do not change with the time) of the occupancy distribution (distribution of the offline queries which are collected at the beginning of the training) can train the reward function in a manner that it gives a partial reward ,which when maximized results in undesirable behavior . The true behavior can be detected using the online training to get the true reward.

E.g Offline training in Pong game can lead the agent into losing the points instead of winning . Also, it can result in volleying for a long duration which is undesired. To get the right behavior, RL training along with human feedback based method has to be used in combination.

Comparing trajectory segments was found to better rather than awarding absolute scores to them because human observers can do the former task better than the latter especially for the tasks related to robotics.

Also, another reason for the above preference is that for the continuous control tasks, when absolute scoring is used, the high variation in scale of rewards complicates the regression problem. This variation in scale is smoothed significantly when only comparisons have to be predicted.

In the Atari tasks, for the absolute scoring method, the rewards are clipped and effectively only the sign is predicted which avoids the difficulty of complicating the regression problem. In robotics tasks, the magnitude of rewards is also important and this method of clipping the rewards and predicting the sign cannot be followed.

The performance of the 'absolute scoring method' is almost equal to the 'preference comparison method'

Frame comparison requires many comparisons in number compared to the clip comparison

Comparison of longer clips was helpful but the 'subtle details observation' that could have happened in frame comparison was missed in this process.

The observers have taken a little time for comparing the shorter clips. For longer clips, the evaluation time is linear function of clip length. The clip length for longer clips is chosen in such a way that evaluation time is linear function of clip length and also it is as short as possible.

In Atari games, comparing longer clips was better than shorter clips comparison as the former provided more context.

Discussions and conclusion:

The method of human feedback is expensive when compared to the method of using a reward function. But, the authors have shown that by careful selection (high variance or random selection) the number of queries that are given for rating by observers are reduced a lot(complexity is reduced by 3 orders of magnitude). Even with such few queries, the reward function was optimized to produce rewards for the required performance. Thus deep RL agents can be trained using human preferences. Also, the total sum of rewards diminishes in further sample complexity environments, here the cost of computation using the RL algorithm is almost same to the cost of computation of method which uses human feedback. Hence, for the practical environments the latter is suitable as good as the former.

Previously, a lot of work was done on topics of preference elicitation using human feedback and getting the reward from an unknown reward function. The present work is done on a large scale and can be applied to many practical RL systems to do complex tasks on real world

Future work would be to improve the efficiency of learning from human feedback and apply it to other domains

In the long run, the authors expect that learning a task using human feedback should not be more difficult than learning using programmatic reward signal, which can help to apply this method for tasks that are more complex rather than for just only to simple tasks

References:

Main Paper:

[1] https://arxiv.org/abs/1706.03741
[2] https://deepmind.com/blog/learning-through-human-feedback/
[3] https://blog.openai.com/deep-reinforcement-learning-from-human-preferences/

TRPO

[1] https://www.youtube.com/watch?v=CKaN5PgkSBc
[2] https://arxiv.org/pdf/1502.05477.pdf

A2C

[1] https://hackernoon.com/intuitive-rl-intro-to-advantage-actor-critic-a2c-4ff545978752
[2] https://www.youtube.com/watch?v=PpVhtJn-iZI&feature=youtu.be&t=33m38s
[3] https://www.cengage.com/resource_uploads/downloads/0030348064_54176.pdf