DEEP REINFORCEMENT LEARNING OF CLOSED LOOP POWERTRAIN CONTROLLERS

- MATHWORKS REINFORCEMENT LEARNING TOOLBOX

Vivek Venkobarao and Gautham T. Sidharthan

External
Emission laws getting stringent with faster development time

Need to develop and customize the new generation of control functions. (Like AI based)

Place of AI/ML in new generation control functions

MATLAB Reinforcement learning toolbox really helped in fast prototyping
AI BASED MODEL BASED CONTROL FUNCTION
WHY AI IN POWERTRAIN CONTROL SYSTEMS

> Challenges in powertrain control

> Designing accurate and scalable mathematical models for vehicle powertrain components which have highly complex and non-linear behavior in real-time operation.

> The real-time environment of a vehicle is very stochastic in nature making control and parameterization of these components very cumbersome using conventional control algorithms (e.g. PID, PI, etc.)

> Environmental concerns of climate change and the resulting ultra low emission requirements set by legislation demand accurate control algorithms with shorter product development lifecycles.

> Need of the hour

> Parametrization space is increasing and the effort to cover all operation conditions with a traditional approach leads to an exponential increasing effort.

> The usage of data driven approaches opens up plenty of possibilities and requires well evaluated decisions about architecture and potential side effects or disadvantages

> The control problem and the requirements must be translated into mathematical optimum criteria for a numerical optimization.
CHALLENGES OF THE CONVENTIONAL RL
PAIN OF AN AI DEVELOPER

> Why Reinforcement Learning

> Stochastic and non perfect nature of environment

> Reinforcement learning algorithms maintain a balance between exploration and exploitation

> Reinforcement Learning algorithms like DDPG, a combination of value-based algorithms and deep learning exist for a wide variety of problem classes.

> This approach is driven by the objective to reduce significantly human parametrization effort.

> Challenges in implementation

> The policy that maps the selected actions based on the observations from the environment. DNN is used.

> The learning algorithm continuously updates the policy parameters based on the actions, observations, and rewards.

> Realizing DDPG Agent
The goal of reinforcement learning is to train an agent to complete a task within an uncertain environment.

The agent receives observations and a reward from the environment and sends actions to the environment.

The reward is a measure of how successful an action is with respect to completing the task goal.

A DDPG agent is an actor-critic reinforcement learning agent that computes an optimal policy that maximizes the long-term reward.
CASE STUDY – POWERTRAIN CONTROLLER

ACCELERATED PROTOTYPING

> Conventional Method

Air path model → Engine Model → Exhaust gas model → State feedback control/PID Control

Fuel Model

> Proposed Method

Air path model → Engine Model → RL controller → Environment Model

Fuel Model
# Choice in Implementation

## Reinforcement Learning Realization Options

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Traditional Methods</th>
<th>MATLAB/SIMULINK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time series Models</td>
<td>Need to derive time vector</td>
<td>Simulink to rescue</td>
</tr>
<tr>
<td>Solve the ODE</td>
<td>Numerical methods to be introduced</td>
<td>Solvers</td>
</tr>
<tr>
<td>Environment Models</td>
<td>Need to be recreated with time series</td>
<td>State of Art plant models</td>
</tr>
<tr>
<td>Action, State</td>
<td>Needs to be derived and implemented</td>
<td>Toolbox generates</td>
</tr>
<tr>
<td>Observations, Reward</td>
<td>Needs to be derived with time vector</td>
<td>Simulink to rescue</td>
</tr>
<tr>
<td>A2C (Actor and Critic)</td>
<td>Can use library</td>
<td>DL API</td>
</tr>
<tr>
<td>DDPG Agent</td>
<td>Can use library</td>
<td>Actor and Critic -&gt; RL toolbox</td>
</tr>
<tr>
<td>Component Integration</td>
<td>Complex – Time series and cross sectional data</td>
<td>RL toolbox with good flexibility of playing with parameters</td>
</tr>
</tbody>
</table>
ACCELERATED PROTOTYPING
STUFF CAN BE SIMULINK

> Observations

> Rewards

> Stop Criterion – when to stop

> Environment Model
ACCELERATED PROTOTYPING
STUFF WITH REINFORCEMENT LEARNING TOOLBOX

> Critic network

```python
statePath = [
    imageInputLayer(numObservations 1 1, 'Normalization', 'none', 'Name', 'State')
    fullyConnectedLayer(256, 'Name', 'criticStateFC1')
    reluLayer('Name', 'criticRelu1')
    fullyConnectedLayer(256, 'Name', 'criticStateFC2')];
actionPath = [
    imageInputLayer(numActions 1 1, 'Normalization', 'none', 'Name', 'Action')
    fullyConnectedLayer(225, 'Name', 'criticActionFC1')];
commonPath = [
    additionLayer(2, 'Name', 'add')
    reluLayer('Name', 'criticCommonRelu')
    fullyConnectedLayer(1, 'Name', 'criticOutput')];
criticNetwork = layerGraph();
criticNetwork = addLayers(criticNetwork,statePath);
criticNetwork = addLayers(criticNetwork,actionPath);
criticNetwork = connectLayers(criticNetwork,'criticStateFC2','add/1/in1');
criticNetwork = connectLayers(criticNetwork,'criticActionFC1','add/2/in2');
```

> Actor network

```python
actorNetwork = [
    imageInputLayer(numObservations 1 1, 'Normalization', 'none', 'Name', 'state')
    fullyConnectedLayer(400, 'Name', 'ActorFC1')
    reluLayer('Name', 'ActorRelu1')
    fullyConnectedLayer(400, 'Name', 'ActorFC2')
    reluLayer('Name', 'ActorRelu2')
    fullyConnectedLayer(1, 'Name', 'ActorFC3')
    tanhLayer('Name', 'ActorTanh')
    scalingLayer('Name', 'actions', 'Scale', 4, 'Bias', 0.5)];
```

> DDPG network

```python
agentOpts = rLDPPGAgentOptions(...
    'SampleTime', Ts,...
    'TargetSmoothFactor', 1e-3,...
    'DiscountFactor', 1.0, ...%
    'ExperienceBufferLength', 1e5, ...%
    'DiscountFactor', 0.99,...
    'MiniBatchSize', 128,...
    'ResetExperienceBufferBeforeTraining', false,...
    'SaveExperienceBufferWithAgent', true);%
agentOpts.NoiseOptions.Variance = 0.3;
agentOpts.NoiseOptions.VarianceDecayRate = 1e-5;
agent = rLDPPGAgent (actor, critic, agentOpts);
```

> Simulation of RL

```python
maxepisodes = 500;
maxsteps = ceil(Tf/Ts);
maxsteps = 200;
trainOpts = rLTrainingOptions(...
    'MaxEpisodes', maxepisodes, ...,
    'MaxStepsPerEpisode', maxsteps, ...%
    'Verbose', true, ...%
    'Plots', 'training-progress', ...,%
    'StopTrainingCriteria', 'EpisodeCount', ...
    'SaveAgentCriteria', 'EpisodeCount', ...
    'SaveAgentValue', 103);
```

Agents can be saved in a .mat file.
> Advantages of Reinforcement based control study

> Good transient response of the controller

> With a good environment model the RL based control provides good recommendations of the control variable

> Simulink also allows limitation for component protection

> The trained control model can be used across various operating zones

> With the usage of toolbox I could focus more on accuracy of environment model

Couldn’t honor the setpoint as it was violating component protection limits
Saturation block is used to limit the action based on the current operating point
GOLDEN WORKFLOW BASED ON EXPERIENCE

RECOMMENDATION FOR FASTER DEVELOPMENT
Excellent Technical support from Mathworks with dedicated calls with the Expert

Great documentation from MATLAB with algorithms and examples on Reinforcement learning

Manually coding the Reinforcement learning and optimization is difficult with current development timelines.

MATLAB allows us to use Simulink to effectively so that state of art plant models can be imported.

Usage of MATLAB Reinforcement toolbox considerably reduced the development time.

The toolbox gave an amazing quick and fast prototyping to realize the generation of agents and optimization.

The usage of data-driven approaches opens up plenty of new possibilities by using AI

Deep Reinforcement Learning based algorithms are able to solve advanced control problems.
LETS DISCUSS