Deep Neural Network Approximation of Nonlinear Model Predictive Control

Yankai Cao and Bhushan Gopaluni
Department of Chemical and Biological Engineering
University of British Columbia
yankai.cao@ubc.ca
Nonlinear Model Predictive control
- Repeatedly solve the optimal control problems online with different $x_0$

\[
\min_{x(k),u(k)} \sum_{k \in \mathcal{T}} l(x(k), u(k)) + V_f(x(N)) \\
\text{s.t.} \quad x(k + 1) = f(x(k), u(k)) \\
x(0) = x_0 \\
x(k) \in \mathbb{X}, x(N) \in \mathbb{X}_f, u(k) \in \mathbb{U} \\
\forall k \in \mathcal{T}
\]

- Implicit control law $u(0) = \kappa_N(x_0)$, expensive online computational cost

Explicit MPC
- Compute the optimal control law offline as a function of all possible states
- Multi-Parametric optimization
- Negligible online, intractable offline for process with 10+ variables
Two-step / “Optimize-then-train” approach
( Zoppoli 1995, Lantos 2015, Zhanfg 2018, Lucia 2018, Gopaluni 2018)
- **Optimize** control actions for multiple initial states to obtain $\{x_0,s, \kappa_N(x_0,s)\}$
- **Train** the neural network to obtain the control law $\hat{u}(0) = \kappa_N(\pi, x_0)$

**Advantages**
- Negligible online computational cost

**Disadvantages**
- Training error of the NN can lead to sub-optimal or even infeasible action even for training samples
- Those errors would accumulate through time (poor closed loop performance)
- Multiple optimal control actions for the same initial states
- Multiple local optimal control actions
Illustrative Example

\[
\begin{align*}
\min_{x(k), u(k)} & \sum_{k=0}^{1} x(k)^2 \\
\text{s.t.} & \quad x(1) = x(0)^2 - u(0)^2 \\
& \quad x(0) = x_0
\end{align*}
\]
All-in-one/“Optimize-and-train” Approach: Stochastic Optimization

\[
\min_{\pi, x_s(k), u_s(k)} \sum_{s \in S} \sum_{k \in T} l(x_s(k), u_s(k)) + V_f(x_s(N)) \\
\text{s.t.} \quad x_s(k+1) = f(x_s(k), u_s(k)) \\
\quad u_s(k) = \hat{\kappa}_N(\pi, x_s(k)) \\
\quad x_s(0) = x_{0,s} \\
\quad x_s(k) \in \mathbb{X}, x_s(N) \in \mathbb{X}_f, u_s(k) \in \mathbb{U} \\
\forall s \in S, \forall k \in T
\]

All-in-one/“Optimize-and-train” approach
- Solve only one large scale optimization problem
- Decide the control law directly instead of control actions
- Optimize closed loop performance directly
- Constraints are satisfied at least for training samples
- Parallel solvers (e.g. PIPS-NLP) can exploit the structure on HPC (CPUs)
- Links to policy search/reinforcement learning
Illustrative Example
Structured Nonlinear Optimization (Scalable Linear Algebra)

\[
\begin{align*}
\min_{\pi, x_s} \sum_{s \in S} f_s(\pi, x_s) \\
\text{s.t. } c_s(\pi, x_s) &\geq 0, \ s \in S \\
x_s &\in X_s, \ s \in S
\end{align*}
\]

Structured Linear System

\[
\begin{bmatrix}
K_1 & B_1 \\
K_2 & B_2 \\
\vdots & \vdots \\
B_1^T & B_2^T & \ldots & B_S^T & K_0
\end{bmatrix}
\begin{bmatrix}
q_1 \\
q_2 \\
\vdots \\
q_S \\
q_0
\end{bmatrix}
= 
\begin{bmatrix}
r_1 \\
r_2 \\
\vdots \\
r_S \\
r_0
\end{bmatrix}
\]

Schur Decomposition

\[
\left( K_0 - \sum_{s \in S} B_s^T K_s^{-1} B_s \right) q_0 = r_0 - \sum_{s \in S} B_s^T K_s^{-1} r_s
\]

\[
K_s q_s = r_s - B_s q_0, \ s \in S
\]

Cao et al. “Scalable modeling and solution of stochastic multi objective optimization problems.” Computers & Chemical Engineering 99 (2017): 185-197.
"Optimize-and-train"/ All-in-one Approach: Recurrent Neural Network

The stochastic optimization problem can be reformulated as the training of RNN
- Existing packages (e.g. TensorFlow, Flux) can exploit the structure on GPUs
- Simple Input constraints (bounds) can always be satisfied by the design of NN
- Treat state constraints as soft constraints
Implementation

```plaintext
function step_model(x_k, u_k)
    return x_k.^2 - u_k.^2
end
control_law = Chain(
    Dense(1, 20, σ),
    Dense(20, 1))
function loss(x0, setpoint)
    x_k = x0
    x = nothing
    for t = 1:N
        u_k = control_law(x_k)
        x_k = step_model(x_k, u_k)
        x = t==1 ? x_k : vcat(x,x_k)
    end
    return Flux.mse(x,setpoint)
end
data=[(I, zeros(N)) for I in -2:0.1:2]
opt = ADAM()
Flux.@epochs 1000 Flux.train!(loss, Flux.params(control_law), data, opt)
```

Additional 10 lines to implement input/state constraints
Feasibility

**Constraint Violation**

For state constraints:

\[ g(x) \leq 0 \]

\[ C_v(\pi) = \max_{x(0) \in X_0, x(k), u(k)} \| [g(x)]_+ \| \]

\[ \text{s.t.} \quad x(k + 1) = f(x(k), u(k)) \]

\[ u(k) = \hat{\kappa}_N(\pi, x(k)) \]

\[ \forall k \in T \]

If \( C_v(\pi) > 0 \)

- Select the scenario leading to the largest violations
- Add the scenario to the training scenarios, and optimize the control law again
- Or chose tighter state constraints in the optimization/training

If \( C_v(\pi) = 0 \)

- It means the control law can drive any state \( x_0 \in \overline{X}_0 \) to \( \overline{X}_f \) in \( N \) steps
- Assume a local control law \( \mu_f(x_0) = K x_0 \) can stabilize any \( x_0 \in \overline{X}_f \), then DNN control law is stable
- Or we can train the control law to ensure that states in \( \overline{X}_f \) remain in the zone
Uncertainty

\[ \hat{\kappa}_N(\pi, x(0)) \]

\[ \hat{\kappa}_N(\pi, x(1)) \]

\[ \hat{\kappa}_N(\pi, x(N-1)) \]

\[ u(0) \]

\[ u(1) \]

\[ u(N-1) \]

\[ x(0) \]

\[ x(1) \]

\[ x(2) \]

\[ x(N-1) \]

\[ x(N) \]

\[ f(x(0), u(0)) \]

\[ f(x(1), u(1)) \]

\[ d(0) \]

\[ d(1) \]

\[ d(N-1) \]

\[ \min_{\pi, x_s(k), u_s(k)} \sum_{s \in \mathcal{S}} \sum_{k \in \mathcal{T}} l(x_s(k), u_s(k)) + V_f(x_s(N)) \]

s.t.

\[ x_s(k+1) = f(x_s(k), u_s(k), d_s(k)) \]

\[ u_s(k) = \hat{\kappa}_N(\pi, x_s(k)) \]

\[ x_s(0) = x_{0,s} \]

\[ x_s(k) \in \mathbb{X}, x_s(N) \in \mathbb{X}_f, u_s(k) \in \mathbb{U} \]

\[ \forall s \in \mathcal{S}, \forall k \in \mathcal{T} \]

Each scenario has data: \( [x_{0,s}, d_s(k)] \)
Nonlinear Quadtank Problem

\[
\begin{align*}
\frac{dz_1}{dt} &= -\frac{a_1}{A_1}\sqrt{2g(z_1 + x_{1s})} \\
&\quad + \frac{a_3}{A_1}\sqrt{2g(z_3 + x_{3s})} + \frac{\gamma_1}{A_1}(v_1 + u_{1s}) \\
\frac{dz_2}{dt} &= -\frac{a_2}{A_2}\sqrt{2g(z_2 + x_{2s})} \\
&\quad + \frac{a_4}{A_2}\sqrt{2g(z_4 + x_{4s})} + \frac{\gamma_2}{A_2}(v_2 + u_{2s}) \\
\frac{dz_3}{dt} &= -\frac{a_3}{A_3}\sqrt{2g(z_3 + x_{3s})} + \frac{(1 - \gamma_2)}{A_3}(v_2 + u_{2s}) \\
\frac{dz_4}{dt} &= -\frac{a_4}{A_4}\sqrt{2g(z_4 + x_{4s})} + \frac{(1 - \gamma_1)}{A_4}(v_1 + u_{1s})
\end{align*}
\]

Training scenarios:
- 81 initial state scenarios with each state variable discretized by 3 points
- for ideal NMPC, we need to solve 81 \times 20 = 1620 optimization problems (N=20 steps)
- These 1620 data pairs are used in the two-step approach

Test scenarios:
- 256 initial state scenarios with each state variable discretized by 4 points

NN controller:
- Structure 4 - 10 - 2 - 2
- Both hidden layers use the activation function tanh
- To guarantee the satisfaction of input constraints, the last layer projects values in the range of [-1,1] to \([v_{min}, v_{max}]\)
Nonlinear Quadtank Problem

### Performance of different controllers

|                  | training |              | testing |              |
|------------------|----------|--------------|---------|--------------|
|                  | cost     | Cons. Viol.  | cost    | Cons. Viol.  |
| ideal NMPC       | 582.18   | 0            | 488.53  | 0            |
| two-step         | 582.64   | 1.85         | 492.83  | 1.85         |
| all-in-one       | 582.29   | 0            | 488.95  | 0.075        |

### Averaged online and offline computational time

|                  | online (s) | offline (s) |
|------------------|------------|-------------|
| ideal NMPC       | 0.016      | -           |
| two-step         | 4e-5       | 2134        |
| all-in-one       | 4e-5       | 1194        |

### Performance of two-step method with different DNN layers

| # of layers | training |              | testing |              |
|-------------|----------|--------------|---------|--------------|
|             | cost     | Cons. Viol.  | cost    | Cons. Viol.  |
| 2           | 582.64   | 1.8502       | 492.83  | 1.8502       |
| 4           | 584.40   | 0.599        | 490.26  | 1.008        |
| 6           | 583.74   | 1.12         | 491.06  | 1.18         |
| 8           | 586.26   | 0.259        | 492.23  | 1.240        |
| 10          | 582.87   | 0.636        | 489.71  | 0.746        |
| 12          | 586.08   | 0.513        | 492.84  | 1.250        |
Summary

All-in-one Approach

- Still works even if optimal control actions are not unique
- Constraints are satisfied at least for training samples
- RNN reformulation might reduce the training time