Approximation of Nonlinear Model Predictive Control Using Mixture Density Networks

Morimasa Okamoto a, Jiayang Ren a, Qiangqiang Mao a and Yankai Cao a,1

a Department of Chemical and Biological Engineering, The University of British Columbia, Vancouver, BC, Canada

Abstract
Model Predictive Control (MPC) is an advanced control method broadly applied to chemical processes. However, the prohibitive online computation time limits its application to nonlinear systems. Although the approximation of the MPC control law via deep neural networks (DNNs) has been studied in recent years, this approach cannot be applied to nonlinear systems if the optimal control problems have multiple optima. When the MPC control law follows one-to-many mappings, it cannot be effectively approximated via DNNs, which provide one-to-one mappings. In this paper, we propose a mixture density network (MDN)-based approximation method for nonlinear MPC. MDNs approximate the MPC control law through conditional probabilities by mixing several estimated Gaussians and then generate several control inputs with the highest probabilities, which means that the network can realize the one-to-many mappings. We also investigate a case study of a nonlinear benchmark process, which demonstrates that our proposed scheme exhibits better control performance than the DNN-based approximation method.

Keywords
Model Predictive Control, Mixture Density Networks, Nonlinear systems.

Introduction
Model Predictive Control (MPC) is an advanced control method applied in various industrial processes, such as chemical plants and power systems (Qin and Badgwell, 2003; Xi et al., 2013). Compared with traditional control methods such as PID controllers, MPC typically provides better performance while satisfying a set of constraints. It is realized by optimizing a finite time horizon control problem and implementing the first control input. However, the prohibitive online computation time limits its real-world applications, especially for nonlinear systems. To achieve fast online control, explicit MPC is proposed by pre-computing the explicit optimal control law using parametric programming techniques (Tøndel et al., 2003). In this way, online computations are reduced to a simple function evaluation. However, offline computing scales poorly with the number of constraints and prediction horizon. Moreover, the application of this approach to nonlinear systems is not straightforward.

In recent years, machine-learning and deep-learning methods have proliferated in many fields, such as image recognition (Krizhevsky et al., 2012) and system identification (Wu et al., 2019). Inspired by these advances, realizing fast online control while maintaining scalability in offline computing by approximating the complex MPC control law via deep neural networks (DNNs) has attracted many focuses. Karg and Lucia (2020) showed that a neural network with rectifier units could exactly approximate the MPC control law of linear time-invariant systems. They further demonstrated that DNNs with several hidden layers could robustly approximate the nonlinear MPC control law (Lucia and Karg, 2018). Considering the time series nature of the process states, Kumar et al. (2018) proposed utilizing Long Short-Term Memory (LSTM) networks to approximate the MPC control law more accurately.

While these DNN-based approximation methods can significantly accelerate online control with acceptable offline computing costs, they cannot address set-valued optimal inputs for nonlinear systems (Cao and Gopaluni, 2020; Li et al., 2022). It is possible that a system with complex nonlinear dynamics has multiple global optima. Therefore, the calculated optimal MPC control laws are usually one-to-many mappings, one state corresponding to several optimal control inputs (which is also called set-valued optimal inputs). Additionally, since the lack of efficient global optimal solvers for the nonlinear MPC problems, when generating the training datasets, we often rely on the local optimal solvers such as Ipopt (Wächter and Biegler, 2006). In that case, the solver computes one of the multiple local optima; thus, the computed MPC control laws also follow one-to-many mappings. The aforementioned DNN-based approximation methods can only approximate one-to-one mappings and are not suitable for one-to-many mappings (as demonstrated in Example 1).

To address the potential set-valued optimal inputs, we propose a mixture density network (MDN)-based approximation
method (MDN-based MPC) for nonlinear MPC. MDNs with several Gaussian components in the output layer can predict the conditional probabilities of possible control inputs by mixing multiple estimated Gaussians, then generate several control inputs with the highest probabilities (Bishop, 1994). Therefore, the network can implement the one-to-many mappings and approximate the nonlinear MPC control law with set-valued optimal inputs.

This paper is organized as follows. Section 2 introduces the DNN-based MPC that has already been proposed and studied. Section 3 describes the proposed MDN-based MPC. We investigate a case study of a nonlinear benchmark process in section 4, which demonstrates that our proposed scheme exhibits better control performance than conventional DNN-based MPC. Finally, section 5 concludes this paper.

### Deep Neural Network-based Model Predictive Control (DNN-based MPC)

MPC is a method that determines control inputs by solving optimization problems repeatedly at each sampling time (Rawlings et al., 2020). Given the initial state $x_0$ and the prediction horizon $N$, the optimal control problem $\mathcal{P}(x_0)$ can be formulated as following

$$
\begin{align*}
\min_{u(k)} \quad & \sum_{k=0}^{N-1} I(x(k), u(k)) + V_f(x(N)) \\
\text{s. t.} \quad & x(k+1) = f(x(k), u(k)), \quad x(0) = x_0 \\
& x(k) \in X, \quad x(N) \in X_f, \quad u(k) \in U, \quad \forall k \in \mathbb{T}
\end{align*}
$$

where $x \in \mathbb{R}^n$ is the state vector of the dimension $n$, $u \in \mathbb{R}^m$ is the control input vector of the dimension $m$, $I(\cdot)$ is the stage cost function, $V_f(\cdot)$ is the terminal penalty function, $k$ is the sampling time, $f(\cdot)$ is the function representing process dynamics, $X$ is the state constraint set, $X_f$ is the terminal state constraint set, $U$ is the input constraint set, and $\mathbb{T} := \{0, 1, \cdots, N-1\}$ is the prediction step set.

We denote the optimal control inputs for $\mathcal{P}(x_0)$ as $u^*(0; x_0) = (u^*(0; x_0), u^*(1; x_0), \cdots, u^*(N-1; x_0))$. Based on the receding horizon scheme, MPC applies the first step input $u^*(0; x_0)$ of the optimal control input vector. Therefore, the MPC control law $\kappa_N(\cdot)$ is a mapping from the initial state to the control input, and is defined as $\kappa_N(x_0) := u^*(0; x_0)$.

DNN-based MPC approximates these mappings in MPC control law via DNNs. Here, DNNs are neural networks with multiple hidden neurons and layers, which are formulated as

$$h^{(l)} = g^{(l)}(W^{(l)}h^{(l-1)} + b^{(l)})$$

where $l$ is the layer number, $g^{(l)}$ is the activation function for the layer $l$, $W^{(l)}$ is the weight of the layer $l$, $h^{(l)}$ is the hidden variables of the layer $l$, and $b^{(l)}$ is the bias of layer $l$ (Goodfellow et al., 2016). $h^{(0)}$ corresponds to the system states of the training samples. By learning the parameter set composed of $W^{(l)}$ and $b^{(l)}$, DNNs can represent complex relationships between the system states and the control inputs as nonlinear approximation functions. Therefore, DNNs can approximate the MPC control law $\kappa_N(\cdot)$ by the pairs of the initial states $x_0$ and the optimal control inputs $u^*(0; x_0)$. In this way, DNN-based MPC can dramatically reduce online computation time because we can obtain the control inputs predicted directly by pre-trained DNNs.

One drawback of the DNN-based MPC is that it can only approximate one-to-one mappings between the system states and the control inputs. When multiple optimal control inputs exist for some system states (called set-valued MPC control law), it will fail to give an accurate approximation. The following example shows a simple nonlinear optimal control problem introduced by Cao and Gopaluni (2020) and Li et al. (2022), in which a set-valued MPC control law is approximated.

#### Example 1: Approximating set-valued MPC control law of the optimal control problem.

$$\begin{align*}
\min_{u(k)} \sum_{k=0}^{1} x(k)^2 \\
\text{s. t.} \quad x(1) = x(0)^2 - u(0)^2, \quad x(0) = x_0
\end{align*}$$

The prediction horizon of this problem is one step. We do not consider the state and input constraint. We generated 10000 training samples by solving the above optimal control problem for 10000 randomly generated initial system states $x_0$. We also generated 10000 test samples in the same way. Since this problem obviously has a set-valued MPC control law $\kappa_N(x_0) := \pm x_0$, two optimal inputs for one initial state $x_0$, the optimization solver selects any one of two optimal control inputs based on the randomly chosen initial guess.

Figure 1 illustrates the MPC control law and the approximated DNN control law with 10 hidden neurons, 1 hidden layer and $tanh$ activation function. From the figure, it seems that the DNN control law is approximately the average of the multiple inputs in the training samples and is close to $\hat{u}(x_0) = 0$, where $\hat{u}$ is the control input predicted by the DNN. It is clear that the trained DNN control law deviates far from the original MPC control law. That is because we try to fit the one-to-many mapping via the DNN, which can only provide the one-to-one mapping.

Table 1 shows the training and test errors of the trained DNN controller with increased hidden layers. To consider the set-valued results, we calculate the mean squared errors...
is the lower limit and \( \bar{\alpha} \) and \( \bar{\sigma} \) are the upper limit. This function project \( z \) to the range between \( \bar{\alpha} \) and \( \bar{\sigma} \). The function \( \sigma_i(z) \) is often sigmoid function with a variable range transformation as the activation function \( g^{(L)}_{\sigma_i} \) for the diagonal elements of the covariance matrix:

\[
g_{\sigma_i}(z) = z + \frac{\bar{\sigma}_i - \bar{\alpha}_i}{1 + \exp(-z)}
\]

where \( \bar{\alpha}_i \) is the lower limit and \( \bar{\sigma}_i \) is the upper limit. This function project \( z \) to the range between \( \bar{\alpha}_i \) and \( \bar{\sigma}_i \). Also, for the non-diagonal elements of the covariance matrix, the activation function is often \( \tanh \).
Problem Description

Numerical Case Study

Problem Description

In the case study, we consider a nonlinear system based on a benchmark problem introduced by Morari and Maeder (2012). The plant model is

\[ x_1(k+1) = 1.05x_1(k) - 0.25x_1(k)x_2(k) + x_2(k) \]  
\[ x_2(k+1) = 0.7x_2(k) + u(k) \]

where \( x_1 \) and \( x_2 \) are elements of the system state and \( u \) is the control input which has a constraint \(-10 \leq u \leq 10\). We change the coefficient in the first term of Eq. (9) from 0.25 to 0.7. Since this system converges from any states to the terminal state \((0,0)\), we set the terminal cost \( V_f(x(N)) \) to 0. The stage cost is

\[ l(x(k), u(k)) = x_1(k)^2 + x_2(k)^2. \]

The prediction horizon \( N \) is 20.

Training and Evaluation

Data Generation: To check the control abilities of DNN and MDN-based MPC from various system states, we divide the state range between \(-10\) and 10 into 321 sub-ranges, such as \(-10, -9.9375, \ldots, 10\). Then, we solve the corresponding optimal control problem for all 103041 initial states and store all pairs of system states and control inputs as the dataset. Since this problem has lots of local optima for one state, the optimization solver will generate different control inputs for one system state if we solve the optimal problem multi-times with random initial guesses. Therefore, we store up to three best inputs with a difference of more than one. Consequently, the number of the stored pairs is 158225, which has 10163 states with three control inputs, 34858 states with two control inputs, and 58020 states with one control input. To compare the difference in performance between DNNs and MDNs, we divided the stored pairs into a training dataset (40\%), a validation dataset (30\%), and a test dataset (30\%).

Training Setting: The settings for training DNNs and MDNs are shown in Table 3. We evaluated different settings for both DNNs and MDNs on the validation dataset and then report the best settings. We limit \( \sigma_u \) to the range between 0.3 and 0.9 by adjusting \( \tilde{z} \) and \( \tilde{z} \). \( \sigma_{ij} \) does not exist because the dimension of \( u \) is one.
Evaluation: We evaluate the performance from two perspectives. The first one is to evaluate the deviation of the predicted control input from the optimal MPC control input. We use the tailored MSE in Eq. (4) to compute the training and test errors. The second one is to evaluate the overall closed-loop performance of the controller. Specifically, different controllers are simulated in the closed-loop system over 20 time steps, starting from every state in test sets. Then we calculate the average values of the cost function. The average cost of ideal MPC is 256.3. Here, because of the input constraints, the predicted inputs from DNNs and MDNs are all projected to be between −10 and 10.

Simulation Result

Since each state may have more than one local optimal input, we prepare the dataset in two ways.

In the first case, for each state, we randomly select one control input from up to 3 best local optimal inputs. This is to simulate the case when we use local solvers to solve optimal control problems only once for each state. Figure 4 shows the MPC control law, and the DNN approximation, and the MDN approximation, when \( x_1 \) is fixed to 0. As we expected, the training and test errors of the MDNs reduce as the number of Gaussians increases. In addition, while the DNN cannot approximate the MPC control law when a state has a unique control input (\( x_2 \in [-5, 4] \)), the DNN tends to have errors at the point where the control law is discontinuous over three segments. We observe that the MDN can approximate the discontinuity quite accurately by combining 3 Gaussians while the DNN tends to have errors at the point where the control law changes significantly. Additionally, as shown in Table 4, MDN-based MPC with several Gaussians exhibits better control performance than DNN-based MPC. These results imply that MDN-based MPC with adequate Gaussians outperforms DNN-based MPC unless MPC control laws are regularly lined up.

Finally, in terms of online computation time, which is critical for online control, one step computation time of MDN-based MPC with 3 Gaussians is 0.00265 (sec) in Python environment. Compared with the other methods in the same condition, it was slower than the DNN-based MPC, 0.00083 (sec), but faster than the ideal MPC, 0.05513 (sec). The runtime gap between MDN-based MPC and ideal MPC will further expand as the prediction horizon \( N \) of the optimization problem increases.

Conclusions

MPC is a control strategy widely applied to a variety of industrial processes. It is challenging for DNNs to approximate the MPC control law of nonlinear systems with set-valued optimal inputs. We propose the approximation of the MPC control law via MDNs to enable approximating the one-to-many mappings. In the case study of the nonlinear benchmark process, MDN-based MPC exhibits high control performance for general nonlinear processes with multiple

<table>
<thead>
<tr>
<th>Loss function</th>
<th>DNNs</th>
<th>MDNs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activation (output)</td>
<td>Linear</td>
<td>Mixed</td>
</tr>
<tr>
<td>Activation (others)</td>
<td>tanh</td>
<td>tanh</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam</td>
<td>Adam</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>Epoch</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>Mini-batch</td>
<td>512</td>
<td>512</td>
</tr>
<tr>
<td>Hidden neuron</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Hidden layer</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Number of Gaussians</td>
<td>—</td>
<td>1, 2, 3</td>
</tr>
</tbody>
</table>

Table 3: Training setting for DNNs and MDNs.

<table>
<thead>
<tr>
<th>Model</th>
<th>DNN</th>
<th>MDN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td>—</td>
<td>1</td>
</tr>
<tr>
<td>Training error</td>
<td>2.819</td>
<td>2.999</td>
</tr>
<tr>
<td>Test error</td>
<td>2.810</td>
<td>2.974</td>
</tr>
<tr>
<td>Average cost</td>
<td>426.5</td>
<td>480.4</td>
</tr>
</tbody>
</table>

Table 4: Training and test errors of DNN and MDN trained by the dataset with one randomly selected local optimal input for each state.
Table 5: Training and test errors of DNN and MDN trained by the dataset including one input with the smallest cost for each state.

<table>
<thead>
<tr>
<th>Model Gaussian</th>
<th>DNN</th>
<th>MDN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training error</td>
<td>0.463</td>
<td>0.464</td>
</tr>
<tr>
<td>Test error</td>
<td>0.453</td>
<td>0.455</td>
</tr>
<tr>
<td>Average cost</td>
<td>275.6</td>
<td>276.3</td>
</tr>
</tbody>
</table>

Although MDNs with several Gaussians are highly expressive and can represent complex MPC control laws, our proposed method has its limitations. One limitation is that the training of MDNs can be computationally expensive. Although this issue is not fatal for the case study because of its simplicity, it will be very expensive if the dimension of the control inputs is numerous. That is because neurons in the output layer increase as the dimension of the control input increases, which means that the networks have numerous parameters to be learned. In the future, we need to investigate efficient approaches to train MDNs with a large network.

Acknowledgment

Y. C. acknowledges funding from the discovery program of the Natural Science and Engineering Research Council of Canada under grant RGPIN-2019-05499. The authors also gratefully acknowledge the computing resources provided by SciNet (www.scinethpc.ca) and Digital Research Alliance of Canada (www.alliancexcan.ca).

References


