分類型ニューラルネットワークを用いたスパース最適制御器の近似

○松永 那瑠, 井上 正樹（慶應義塾大学）

Approximation of Sparse Optimal Controller Using Classification Neural Network

∗N. Matsunaga and M. Inoue (Keio University)

Abstract− Optimal control, which determines the control actions based on the solution to an optimization problem, has great potentials to contribute to many research and application fields. However, due to its high computational burden, the optimal control cannot be implemented to fast real-time systems with low-spec microprocessors. To overcome the drawback, there have been tried the approximations of the control law by utilizing neural networks (NNs). This manuscript addresses the NN-based approximation of the optimal control law, in particular, focusing on the maximum hands-off control, which is a control method promoting the sparsity in the control actions. Unlike most of the conventional works, the Classification NN is applied for the approximation rather than the Regression NN. The usefulness is verified in numerical simulations.

Key Words: Deep neural network, Sparse optimal comtrol, Classification problem

1 はじめに

1.1 研究背景

今日の社会は膨大な量のデータで溢れており，そのデータの中から本質のみを抜き出すスパースモデリングという考え方が重要とされ，幅広い分野で近年注目されている．スパース化されたデータは，元のデータと比較して大幅に少ない特徴量のデータが抽出されることで情報量が圧縮される．例えば，地球科学分野では地学現象に関連する様々な観測データから地球の構造や地学現象を逆問題として抽出する研究 [1)]，医学分野では，圧縮センシングと呼ばれるスパースモデリングの応用技術を用いてより少ない時間で MRI の画像を撮影する研究 [2)] があり，スパースモデリングの有効性が示されている．

工学的分野では，スパースモデリングを自動車をはじめとした動的システムに対して応用する研究 [3)] があり，動的スパースモデリングと呼ばれている．特に，制御分野においては，制御操作を加える時間を最小化する Maximum hands-off 制御 [4)] が提案されている．操作時間を最小化することで，アクチュエータの動作時間を減らしエネルギーや故障率を低減できることが期待される．

スパース性を測る指標としては，ベクトルの非零の要素数を示す L^0 ノルムが基本的に用いられる．しかし，L^0 ノルムを含んだ評価関数のもと最適な制御操作を導く L^0 最適制御問題は組合せ最適化問題となり，中規模以上の問題では解くことが極めて難しい．そこで，特定の条件下において L^0 最適制御との等価性が知られている L^1 最適制御 [4)] に着目する．L^0 ノルムの代わりに L^1 ノルムを評価関数に使用した L^1 最適制御問題は凸最適化問題であり，L^0 ノルムを用いる場合と比較して効率的に動的スパースモデリングが実現可能である．

しかし，最適化の実行時間が限られている実時間スパース最適制御を想定した場合，L^1 最適制御には依然として課題が残されている．L^1 最適制御問題は陽に制御則が記述されておらず，継続的に制御をおこなうには繰り返し計算が必要である．そのため，大規模な制御システムを対象とする場合や計算能力の低いマイクロコンピュータに制御器を実装する場合では限られた時間内に処理を実行できず，リアルタイムでの運用が困難である．この点において，スパース最適制御を広く応用展開するためには大きな課題が残されている．

最適制御器の計算負荷低減のための取り組みとして，ニューラルネットワーク（NN）を用いて制御器を近似する研究が 1980 年代より盛んにおこなわれてきた [5)6)]．NN モデルにより制御器を記述することで，リアルタイムでの繰り返し計算を不要として処理を簡略化することができる．近年でも，学習アルゴリズムの高速化や計算機能力の向上を背景として精力的に研究がされている [7)8)9)10)]．本研究でも，NN モデルを用いてスパース最適制御器を近似する問題に取り組む．

1.2 本研究の動機付け

ここではスパース最適制御器の NN モデルによる近似をおこない，その有用性とともに本研究で解決すべき課題を例示する．詳細な設定の説明は 4 節での説明にゆずることとして，ここでは結果のみ紹介する．

スパース最適制御器を回帰型の NN モデルによって近似した．最適制御器による制御操作と，近似制御操作を Fig.1 に示す．上図はスパース最適制御問題を求解して得られた制御操作，下図は回帰型 NN を用いてそれを推論した近似制御操作を示している．また，本稿ではスパース性を評価する尺度としてシミュレーション時間において入力が $0 \pm 1\%$ の時刻が占める割合を入力休止率（pause rate）と定義する．Fig.1 で示されたシミュレーション結果について，実行時間と入力休止率を Table1 に示す．

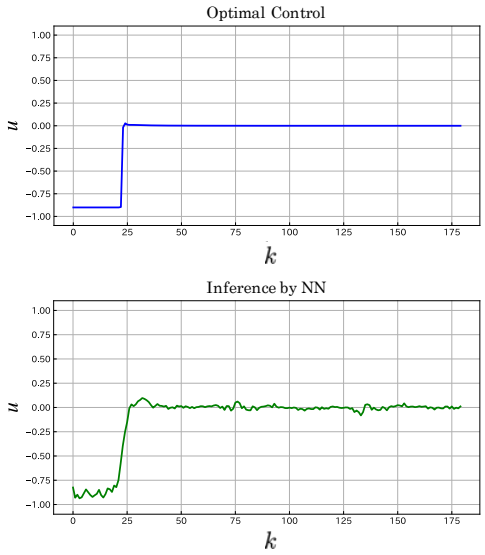

Fig. 1: Regression NN lost sparsity.

Table 1: Comparison of sparsity between Optimizer and Regression NN

	Optimizer	Regression NN
Execute time [s]	0.78	0.024
Pause rate [%]	85.5	37.8

Table1 では，実行時間が 1/10 以下に削減できたことが読み取れる．これより，計算負荷を軽減する点において最適制御近似の有効性が確認できた．さらに，最適制御器による制御操作が，25 ステップ近辺から先のステップでは 0 となっていることが Fig.1 から読み取れる．すなわち，スパースな制御操作になっているといえる．一方，回帰型 NN による近似制御操作では値が完全に 0 になっていない時刻が多く存在することが読み取れる．Table1 に示した NN による近似では制御操作の休止率は最適制御器による制御操作の休止率の半分以下に減少している．これより，NN によって得られた近似制御操作は元々の制御操作が有していたスパース性を失っていることがわかる．

このような問題が発生する原因について，使用した NN の種類が関係していると考えられる．シミュレーションで利用した NN は回帰型 NN という種類であり，連続値を最適解として出力する．筆者らの知る限り，最適制御器を近似する先行研究のほとんどが回帰型の NN モデルを用いている [5)6)]．一方，通常は画像認識などの分野で利用されている分類型という NN が存在する．分類型 NN からは，正解の候補として設定した要素の中から出力が決定するため，最適制御器の近似に利用すると，得られる制御操作の種類を制限することが可能である[1]．

[1]例えば，0〜9 の数字が描かれた画像を入力し，画像に描かれた数字がいくつであるかを当てる問題に適用することができる．この例では，正解の候補を 0〜9 と設定するのが適切である．

そこで本稿では，スパース性の表現には回帰型 NN よりも上述の特徴を有した分類型 NN の方が適しているという仮説を立て，分類型 NN を用いたスパース最適制御器の近似を提案する．

2 スパース最適制御問題の定式化

本節では，スパース最適制御問題の定式化をおこなう．記述を簡単にするため，本稿で扱う制御対象は線形時不変システムであるとし，次の離散時間状態方程式でモデル化されるものとする．

$$x_{k+1} = Ax_k + Bu_k \tag{1}$$

ここで，$k \in \{0, 1, \ldots\}$ は離散時刻，x_k は時刻 k の状態ベクトル，u_k は時刻 k の制御操作であるとする．

これより，制御対象 (1) の状態 x_0 が観測されたとして，おこなうべき制御操作の列 $U = [u_0, \ldots, u_{N-1}]^\top$, $U \in \mathbb{R}^N$ を求める問題を定式化する．ここで U の各要素 $u_0, \ldots, u_{N-1}$ は各時刻 $k \in \mathcal{I} := \{0, \ldots, N-1\}$ でおこなう制御操作をあらわしている．以下では，制御ホライズンを N とする．すなわち，ある時刻での観測結果をもとに将来 N 時刻先までの制御操作を決定し操作することになる．

状態 x_k のレギュレーションをスパースな制御操作列 U で実現することをねらって，次の評価関数 $J(X, U)$ を定義する．

$$J(X, U) = x_N^\top P x_N + \sum_{k=0}^{N-1} \{x_k^\top Q x_k + |u_k|\} \tag{2}$$

ただし，$X = [x_0^\top, \cdots, x_N^\top]^\top$ である．また，P, Q はそれぞれ半正定，正定行列である．評価関数 (2) の第一項は終端コスト，第二項以降はステージコストを表している．ステージコストのうちで特に $\sum_{k=0}^{N-1} |u_k|$ は，制御操作列 U をスパース化させる役割がある．また，入力および状態に関する制約が関数 $h(x_k, u_k)$ で示されているとすると，スパース最適制御問題は次の (3) 式で定式化できる．

$$\begin{aligned} \min_U \quad & J(X, U) && \text{(3a)} \\ \text{sub.to} \quad & x_{k+1} = Ax_k + Bu_k,\ k \in \mathcal{I} && \text{(3b)} \\ & h(x_k, u_k) \le 0,\ k \in \mathcal{I} && \text{(3c)} \end{aligned}$$

Fig.2 に，(3) 式で定式化したスパース最適制御器の入出力の関係をまとめている．最適制御器は状態 x_0 をもとに最適制御問題の求解をおこない，その最適解である制御操作 $U^* = [u_0^*, \ldots, u_{N-1}^*]^\top$ を出力する．しかしながら前節で示したように，計算負荷の大きさのためにリアルタイムな制御が難しいことがある．次節ではその計算負荷を軽減するための手法として本稿で提案する，分類型 NN を用いた最適制御器の近似手法について説明する．

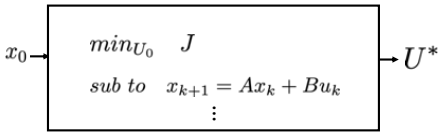

Fig. 2: Optimal control U^* is generated by Optimizer.

3 分類型 NN による最適制御器の近似

3.1 分類型 NN の構造

本節では，(3) の最適制御問題の求解で得られる x_0 と U^* の入出力関係を学習し，最適解 U^* の近似値を得る分類型 NN を提案する．

はじめに分類型 NN の出力の設定を与える．分類型 NN が出力するデータは離散値であるため，制御操作列 U の要素 u_k が取りうる値を離散値に制限したモデルを構築する．本研究では特に，分類型 NN で出力される制御操作を $u_k \in \{-1, 0, 1\}$ と制限する．すなわち，分類型 NN は 3 種類の制御操作 $\{-1, 0, 1\}$ の中から最適な操作を選択する問題を解く制御器とする．なお，このような制御をバンバン制御と呼び，L^1 最適制御がバンバン制御となるならばそれは L^0 最適解 (スパース最適解) にもなることが知られている [11]．

この設定のもと，分類型 NN モデルの構造を与える．ここでは，最適制御操作列である U^* を

$$U^\dagger = M(Y) \tag{4}$$
$$Y = f_{\mathrm{NN}}(x_0) \tag{5}$$

から求められる $U^\dagger := [u_0^\dagger, \ldots, u_{N-1}^\dagger]^\top \in \mathbb{R}^N$ で近似することを考える．ただし，$Y \in \mathbb{R}^{3N}$ は推論のための中間変数で M，f_{NN} は関数である．

関数 f_{NN} は x_0 から Y を出力する．$U^\dagger \in \mathbb{R}^N$ であるのに対して，$Y \in \mathbb{R}^{3N}$ であることに注意されたい．上で設定したように $U^\dagger$ の各要素 $u_k^\dagger$ が取り得る値は $\{-1, 0, 1\}$ の 3 値のみに限定されている．この 3 値を表現するために各 Y_k を $Y_k \in \mathbb{R}^3$ としている．制御操作列 $U^\dagger \in \mathbb{R}^N$ であるため，

$$\begin{aligned} Y &= [Y_0, \ldots, Y_k, \ldots, Y_{N-1}]^\top \\ &= [[y_0^{-1}, y_0^0, y_0^1]^\top, \ldots, [y_{N-1}^{-1}, y_{N-1}^0, y_{N-1}^1]^\top]^\top \end{aligned} \tag{6}$$

の Y 全体では $Y \in \mathbb{R}^{3N}$ である．

ここで得られた Y は (4) に示すとおり，関数 M によって $U^\dagger$ に変換される．M は，Y の各要素 Y_k について $\{-1, 0, 1\}$ のうち最適なものを割り当てる処理をおこなう関数である．分類型 NN による推論では，各要素が最適である確率が出力データ Y として得られる．例えば，$Y_k = [y_k^{-1}, y_k^0, y_k^1]^\top$ のうち最大の要素が y_k^1 である場合には，最適な制御操作 u_k は $u_k = 1$ となる．そのため，対応する三要素 $[y_k^{-1}, y_k^0, y_k^1]$ の中で最も数値が大きい要素を制御操作として取り出す操作が関数 M である．このような関数 M は

$$M(Y) = [m(Y_0), \ldots, m(Y_{N-1})]^\top \tag{7}$$

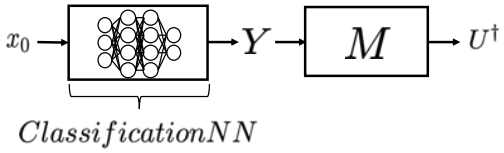

Fig. 3: Clasification NN estimates $U^\dagger$.

のように記述される．ただし，$m(Y_k)$ は

$$m(Y_k) = \begin{cases} -1 & \text{if } \max\{Y_k\} = y_k^{-1} \\ 0 & \text{if } \max\{Y_k\} = y_k^0 \\ 1 & \text{if } \max\{Y_k\} = y_k^1 \end{cases} \tag{8}$$

で表され，これを利用することで各要素が $\{-1, 0, 1\}$ の値をとるベクトル $U^\dagger$ が得られる．

状態 x_0 から近似制御操作 $U^\dagger$ を求める分類型 NN の構造は Fig.3 で示される．このうち，x_0 から Y を求める関数 $Y = f_{\mathrm{NN}}(x_0)$ が分類型 NN で得られる．さらに，(7),(8) にしたがい $\{-1, 0, 1\}$ のうち最適な操作を N 時刻分決定する関数 M によって近似制御操作 $U^\dagger$ が得られる．

3.2 分類型 NN による数値実験および考察

分類型 NN の学習及び推論を，実際の数値を用いておこなった．本項ではその結果を示した上で，分類型 NN を用いたスパース最適制御器の近似の有効性について考察する．

制御対象としてはつぎの連続時間システム

$$\dot{x} = A_c x + B_c u \tag{9}$$

を考える．ここで，

$$A_c = \begin{pmatrix} 0 & -1 \\ 2 & 0 \end{pmatrix},\ B_c = \begin{pmatrix} 2 \\ 0 \end{pmatrix} \tag{10}$$

である．スパース最適制御のために，(9) を離散時間 $t_d = 0.05$ で離散化することで (1) のモデルを得る．また，設計パラメータとして

$$P = Q = 1/20 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{11}$$

を使用した．制御ホライズン $N = 60$ とし，制御操作列 U に含まれる u_k に対して

$$-1 \le u_k \le 1,\ k \in \{0, \ldots, 59\} \tag{12}$$

と不等式制約を課した．なお状態の制約条件はないものとした．

深層学習をおこなうために必要な訓練データについて説明する．はじめに，20000 通りランダムに決定した状態

$$x_{0(j)} = [x_{01(j)}\ x_{02(j)}]^\top\ j \in \mathcal{J} = \{1, \ldots, 20000\} \tag{13}$$

を用意した．ただし，学習の効率化を図るため

Table 2: Settings of Classification NN

Structure	$2 \times 64 \times 64 \times 180$
Loss function	BCEWithLogitsLoss
Optimizer	Adam
Batch size	100
Activation function	ReLU
Activation function (output layer)	Identity function
Number of training [epoch]	10

$x_{01(j)}$ $x_{02(j)}$ はともに閉区間 $[-2, 2]$ から決定した．状態 $x_{0(j)}$ に対して (3) の最適化問題を解き，

$$U^*_{(j)} = [u^*_{0(j)} \ \cdots \ u^*_{k(j)} \ \cdots \ u^*_{59(j)}]^\top, \ j \in \mathcal{J} \tag{14}$$

を得た．さらに，$U^*_{(j)}$ の各要素 $u^*_{k(j)}$ をもとに

$$[y^{-1}_{k(j)} \ y^0_{k(j)} \ y^1_{k(j)}] = \begin{cases} [1,0,0] & \text{if } -1 \le u^*_{k(j)} < -0 \\ [0,1,0] & \text{if } u^*_{k(j)} = 0 \\ [0,0,1] & \text{if } \ 0 < u^*_{k(j)} \le 1 \end{cases} \tag{15}$$

の規則にしたがい

$$Y^*_{(j)} = [Y^\top_{0(j)} \ \cdots \ Y^\top_{k(j)} \ \cdots \ Y^\top_{59(j)}]^\top \tag{16}$$

$$Y_{k(j)} = [y^{-1}_{k(j)} \ y^0_{k(j)} \ y^1_{k(j)}] \tag{17}$$

のように $Y^*_{(j)} \in \mathbb{R}^{180}$ を求めた[2]．そして，$x_{0(j)}$ および $Y^*_{(j)}$ から構成される訓練データ

$$D_{(j)} = \{x_{0(j)}, Y^*_{(j)}\}, \ j \in \mathcal{J} \tag{18}$$

を作成した．

得られた訓練データ $D = \{D_{(1)}, \ldots, D_{(20000)}\}$ を利用して，分類型 NN の深層学習をおこなった．学習にあたっては Python のオープンソース機械学習ライブラリである Pytorch を使用し，Table2 に示す条件を設定した．学習の完了後，学習済み NN の推論をおこなった，その結果として得られた $U^\dagger$ の入力休止率，状態の L^2 ノルムを計算し性能を評価した．

学習済み分類型 NN に状態 $x_0 = [1.28, -1.89]^\top$ を入力して得た $U^\dagger$ を Fig.4 内の緑線で示す．ここでは，図内青線で示されている最適制御列である U^* との比較をおこなっている．$k = 20$ 以前では制御操作 $u_k = -1$，それ以降では $u_k = 0$ であることが確認できる．また，60 ステップ分の入力休止率は最適制御器による最適制御操作 U^* が 56.7 %であるのに対して分類型 NN による近似制御操作 $U^\dagger$ は 60.0 %であった．したがって，スパース性能の観点から評価すると $U^\dagger$ は適しているこ

[2](15) 式について，数値計算上 $u^*_{k(j)} = 0$ となることはごく稀である．そのため，実際には $-0.1 \le u^*_{k(j)} \le 0.1$ である場合について $[y^{-1}_{k(j)} \ y^0_{k(j)} \ y^1_{k(j)}] = [0,1,0]$ とした．

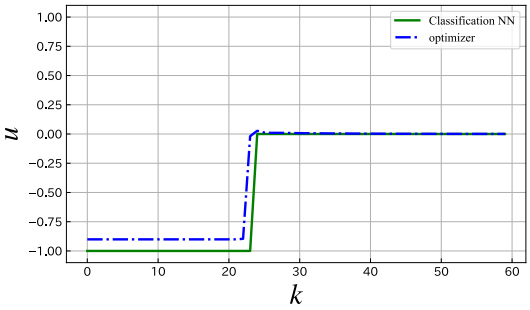

Fig. 4: Classification NN can keep sparsity.

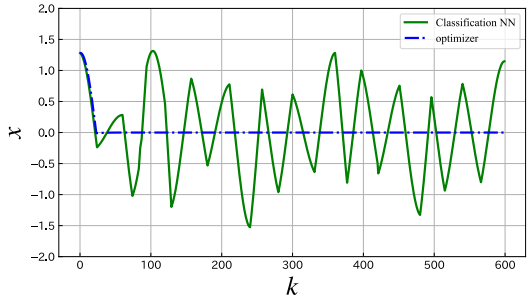

Fig. 5: States cannot converge on 0 by using Classification NN.

とがわかる．

さらに，繰り返し制御操作を推論することで制御をおこなった．制御対象の挙動として，状態 x_k の第一要素のみ Fig.5 に示す．近似のない最適制御では状態は目標値 0 に留まり続ける．しかし，分類型 NN による推論の結果得た近似制御操作を利用しても，状態を 0 付近で留めることができなかった．状態の変動の大きさを評価するためにその L^2 ノルムを計算すると，最適制御では 20.19 であったのに対し，分類型 NN による制御は 367.96 となった．したがって，分類型 NN による近似では状態の制御性能が明らかに悪化したと結論づけられる．

原因の一つとして，分類型 NN を利用する上で $u_k \in \{-1, 0, 1\}$ のように制御操作の数値を 3 値に制限したことが挙げられる．そのため，連続値を取れる最適制御と比較して，精密な操作がおこなえなかったと考えられる．そこで以上の欠点を補うべく，分類型 NN を利用した学習モデル Dual-Net を次節で新たに提案し検証する．

4 Dual-Net

4.1 Dual-Net の構造

3.2 項で検証したように，分類型 NN から生成された制御操作 $U^\dagger$ は狙い通りスパース性を有することを確認した．しかし，制御操作の数値を離散値 $\{-1, 0, 1\}$ に制限したために精密な操作を加えられなくなり，状態の制御性能が劣化した．そこで本節では，スパース性に加えて精密な制御操作を可能とする新たな学習モデルである Dual-Net を提案する．

Dual-Net の構造模式図を Fig.6 に示す．Dual-Net は，

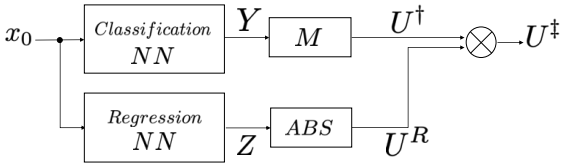

Fig. 6: Dual-Net: it estimates signs and their magnitude separately.

3 節で提案した分類型 NN に回帰型 NN を加えた構造をもつ．二つの NN モデルにそれぞれ役割をもたせているところに Dual-Net の特徴がある．最適解 U^* の符号を分類型 NN モデルにより，その大きさを回帰型 NN モデルを用いてそれぞれ推論し，掛け合わせることで近似をおこなう．

Dual-Net で生成する制御操作列を $U^\ddagger \in \mathbb{R}^N$ とおく．さらに，分類型 NN モデルの出力を 3 節と同じように $U^\dagger$ と，回帰型 NN モデルの出力を新たに $U^R = [u_0^R \cdots u_{N-1}^R]^\top \in \mathbb{R}^N$ とそれぞれ書くことにする．このとき，Dual-Net はつぎのように記述される．

$$U^\ddagger = U^\dagger \circ U^R \tag{19}$$

ただし，記号 $\circ$ はベクトルの要素積である．すでに述べたように $U^\dagger$ は (4), (5) 式にしたがい分類型 NN から得ることができる．また，U^R は

$$\begin{aligned} U^R &= \mathrm{ABS}(Z) \\ &:= [|z_0|, \ldots, |z_k|, \ldots, |z_{N-1}|]^\top \qquad (20) \\ Z &= g_{\mathrm{NN}}(x_0) \qquad (21) \end{aligned}$$

で生成される．ここで ABS は各要素ごとに絶対値をとるベクトル値関数，g_{NN} は回帰型 NN を表す．

分類型 NN 部分については，3 節で提案した近似制御操作 $U^\dagger$ を利用している．$U^\dagger$ の各要素がとりうる値は，3.1 項で制限したように $\{-1, 0, 1\}$ のいずれかである．ここで，U^R の要素 u_k^R は全て正であるとすると，$u_k^\ddagger$ の符号は必ず $u_k^\dagger$ の符号に等しくなる（$u_k^\dagger = 0$ の時は必ず $u_k^\ddagger = 0$ である）．したがって，(4), (5) の分類型 NN モデルがもつスパース化の性質をこの (19) の Dual-Net でも引き継ぐことが期待できる．また，Dual-Net のうちの回帰型 NN モデル部分にも注目する．すでに述べたように $g_{\mathrm{NN}}(x_0)$ がニューラルネットワークを表現している．1.2 項で利用した回帰型 NN によって得られた近似制御操作は，この Z を利用したものである．

次項では，Dual-Net を利用したスパース最適制御器の近似の数値実験をおこない，もとの最適制御器や 3 節で与えた分類型 NN モデルによる近似結果と比較する．

4.2 Dual-Net による数値実験および考察

本項では，(19) で示した Dual-Net を用いてスパース最適制御器の近似をおこない，その有用性を検証する．また，(3) の最適制御器だけではなく (4), (5) で与えた分類型 NN モデルや (21) の回帰型 NN モデルとも比較する．特に，生成される制御操作列のスパースや状態抑制の制御性能の評価をおこなう．

Table 3: Settings of Regression NN: part of Dual-Net

Structure	$2 \times 64 \times 64 \times 60$
Loss function	MSELoss
Optimizer	Adam
Batchsize	100
Activation function	ReLU
Activation function (output layer)	Identity function
Number of training [epoch]	10

分類型 NN やもとのスパース最適制御器との比較のため，制御対象の状態方程式，最適制御問題の設計パラメータ，制約条件はそれぞれ (9)〜(11) と同じものとした．また，予測ホライズン N 及び離散時間も 3.2 節における数値実験と同じく $N = 60$，$t_d = 0.05$ とした．

Dual-Net の分類型 NN 部分については，3.2 項で設定したものと同一のものを利用したため，本項での説明は割愛する．ただし，異なる点として，学習に使用した訓練データは 10000 個とした．これは，訓練データの量について分類型 NN や回帰型 NN を単独で利用した場合の実験結果との公平性を保つためである．

Dual-Net の回帰型 NN 部分の学習について説明する．はじめに，学習に使用した訓練データについて述べる．使用した訓練データ E_j は，

$$E_{(i)} = \{x_{0(i)}, U_{(i)}\},\ i \in \{1, \ldots, 10000\} \tag{22}$$

で表される．このうち，$x_{0(i)}$ は (13)，$U_{(i)}$ は (14) に示す手順で同様に得られるが，こちらも使用する訓練データは 10000 個としている．

$E = \{E_1, \ldots, E_{10000}\}$ の収集後，それらを利用して回帰型 NN 部分の深層学習をおこなった．その際，学習モデルには Table3 に示す条件を設定した．本項で構築した回帰型 NN と 3.2 項で構築した分類型 NN からそれぞれ U^R と $U^\dagger$ を計算し，(19) から $U^\ddagger$ を求めた．さらに，$U^\ddagger$ の入力休止率や制御対象の状態の L^2 ノルムを計算して，Dual-Net の性能を評価した．

Dual-Net を利用した制御シミュレーションの結果を Fig.7 に示す．状態 $x_0 = [1.28, -1.89]^\top$ を学習済の Dual-Net 及び最適制御器に入力した場合に得られた制御操作を示している．Dual-Net から生成される $U^\ddagger$ が図内赤線，最適制御操作である U^* が図内青線で示されている．$k = 25$ 以降で，$u_k^\ddagger$ は $u_k^\ddagger = 0$ となっており，最適制御操作 u_k^* の挙動を近似できていることが確認できる．

また，学習済み Dual-Net を繰り返し利用し，制御をおこなった．制御対象の挙動として，状態 x_k の第一要素のみを Fig.8 に示す．3.2 項で示したように，図内緑線で示している分類型 NN を単独で使用した結果得られた $U^\dagger$ では，状態 x_k が 0 に留まっておらず大きく変

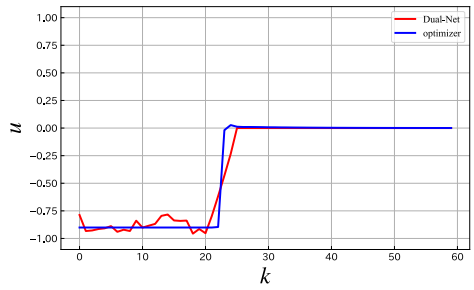

Fig. 7: Control actions generated by Dual-Net and sparse-optimal control.

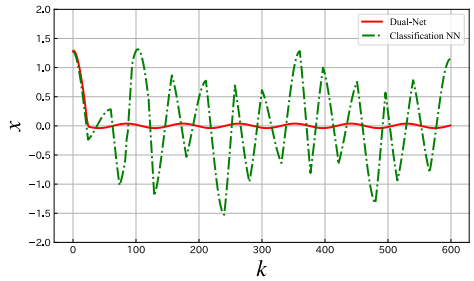

Fig. 8: State trajectory achieved by Dual-Net and Classification NN.

Table 4: Dual-Net achieved both keeping sparsity and following performance

	Pause rate [%]	L^2 norm
Optimizer	56.7	20.19
Regression NN	21.7	20.47
Classificartion NN	60.0	367.96
Dual-Net	58.3	20.96

動している．一方，赤線で示す Dual-Net から生成した $U^{\ddagger}$ を用いた場合には，状態を 0 近傍に留めることができている．

続いて，入力休止率及び状態の L^2 ノルムを計算することで，定量的な評価をおこなう．スパース最適制御器，分類型 NN，回帰型 NN および Dual-Net のそれぞれの制御シミュレーションの結果について，入力休止率と状態の L^2 ノルムを一覧として Table4 にまとめた．シミュレーションした 3 種類の学習モデルのうち，Dual-Net では両方の指標でもとの最適制御器と近い値であることを読み取れる．したがって，Dual-Net では休止率と状態抑制性能の両立が可能であった．

5 おわりに

本稿では，スパース最適制御器のニューラルネットワークモデル（NN モデル）による近似に取り組んだ．標準的な方法のように回帰型 NN モデルではなく，分類型 NN モデルを用いることを提案した．さらにスパース最適制御器の近似精度を高めるために，分類型 NN モデルに回帰型 NN モデルを併用する Dual-Net を提案し，そのモデル構造や学習での工夫などを述べた．最後に数値実験をおこない，Dual-Net によりもとの最適制御器のようにスパースな制御操作による高精度な状態抑制性能を達成できることを示した．

提案した Dual-Net において，分類型 NN 部分は制御操作列の符号を表現していると解釈できる．より広く捉えると分類型 NN 部分は動的な量子化器として振る舞うものである．Dual-Net の解釈を進めることが今後の課題である．特に回帰型 NN モデルに静的な量子化器を付与する場合との比較検討が必要である．単純な比較では，量子化器を追加するだけで容易に実装可能である回帰型 NN に量子化器を付与したモデルに軍配が上がる．そのため，深層学習を通して制御操作のスパース性を実現する分類型 NN 独自の特徴を生かして，静的な量子化器との差別化をはかりたい．

謝辞

本論文の執筆にあたり，原啓太氏をはじめ井上研究室の皆様から大変有意義な助言を頂きました．心より感謝を申し上げます．本研究は，JSPS 科研費基盤研究 (B) 20H04473 の支援を受けたものです．

参考文献

1) 桑谷立，中田令子，岡田真人，堀高峰：スパースモデリングの地球物理学への応用，電子情報通信学会誌，**Vol.99** No.5, 406/410 (2016)
2) 岡田知久, 山本憲, 伏見育崇, 山本貴之, 富樫かおり：スパースモデリング——医用 MRI 画像への応用——, 電子情報通信学会誌, **Vol.99** No.5, 434/438 (2016)
3) 永原：動的スパースモデリングの理論と応用，システム／制御／情報，**Vol.61** No.4，146/151 (2017)
4) M. Nagahara, D. E. Quevedo, and D. Nesic: Maximum hands-off control: A paradigm of control effort minimization, IEEE Transactions on Automatic Control, **Vol.61**, No.3, 735/747 (2015)
5) D. Psaltis, A. Sideris, and A. A. Yamamura: A multilayered neural network controller, IEEE Control Systems Magazine, **Vol.8**, No.2, 17/21 (1988)
6) T. Parisini and R. Zoppoli: A receding-horizon regulator for nonlinear systems and a neural approximation, **Vol.31**, Issue.10, 1443/1451 (1995)
7) 森安竜大, 上田松栄, 池田太郎, 永岡真, 神保智彦, 松永彰生, 中村俊洋: 機械学習によるディーゼルエンジン吸排気系の実時間 MPC 設計, 計測自動制御学会論文集, **Vol.55**, No.3, 172/180 (2019)
8) B. Karg and S. Lucia: Stability and feasibility of neural network-based controllers via output range analysis, in Proceedings of the 59th IEEE Conference on Decision and Control, 4947/4954 (2020)
9) H. Hu, M. Fazlyab, M. Morari and G. J. Pappas: Reach-SDP: Reachability analysis of closed-Loop systems with neural network controllers via semidefinite programming, in Proceedings of the 59th IEEE Conference on Decision and Control, 5929/5934 (2020)
10) X. Zhang, M. Bujarbaruah, and F. Borrelli: Safe and near-optimal policy learning for model predictive control using primal-dual neural networks, in Proceedings of the 2019 American Control Conference, 354/359 (2019)
11) T. Ikeda and M. Nagahara: Maximum hands-off control without normality assumption, in Proceedings of the 2016 American Control Conference, 209/214 (2016)