2023 年度

卒　業　論　文

題　目　RNN を用いた
SteamGame の接続数の予測に関する研究

学籍番号　20A3058

氏　名　川又　翔

提出月日　2024 年 2 月 5 日

指導教員　二宮　洋

湘南工科大学工学部情報工学科

概要

近年，ゲーム産業の発展より，家庭用ゲーム機や PC も高性能になり，様々なゲームが発売されている．プレイヤー数の増加に伴い，ゲームのサーバーがダウンするなどの問題も増加している．

本研究では，ゲームの値段やプレイヤー数を入力とし，次月のプレイヤーの最大同時接続数の増加率を予測する AI を実装しこの問題の解決を試みる．提案手法では，Popularity of games on Steam[1]というデータセットと SteamDB[2]から取得したデータを，時系列データを扱うニューラルネットワークである Recurrent Neural Network(RNN)で学習する．

RNN と RNN を改善したモデルである Long-short Term Memory(LSTM)との比較を行った．また，中間層の数を増やすと性能が向上することが，一般的に知られるため RNN の中間層を一層と四層で比較実験を行った．実験結果として，LSTM より RNN を用いたモデルの方が，予測精度が良いことが確認できた．また RNN を多層化することで，予測精度が向上することができた．また，RNN を多層化することにより時系列の長さを収縮できることを確認することができた．

本研究では，RNN を用いて次月のプレイヤーの最大同時接続数の増加率を予測する AI の実装を試みた．RNN を多層化することで，入力するデータの月の数を減らしても予測精度を損なわないことを確認した．今後の課題として，より入力するデータの月の数を減らすべく，モデルの構造等の条件を変え，実験することが挙げられる．

目次

第１章　はじめに

近年，ゲーム産業の発展より，家庭用ゲーム機や PC も高性能になり，様々なゲームが発売されている．また，E-sports などのオンラインゲームの大会やインフルエンサーによるゲームの紹介によってゲームプレイヤーの人口が増加している．ゲームプレイヤーの人口増加に伴い，ゲームサーバーがダウンするなどの問題も増加している．よってゲームプレイヤーの増加率を予測することで，サーバーダウン に起因する日付の特徴を見つけることができると考える．

本研究では，ゲームの値段やプレイヤー数を入力とし，次月のプレイヤーの最大同時接続数の増加率を予測する AI を実装しこの問題の解決を試みる．提案手法では，Popularity of games on Steam[1]というデータセットと SteamDB[2]から取得したデータを，時系列データを扱うニューラルネットワークである Recurrent Neural Network(RNN)で学習する．

第 2 章　基盤技術

本章では，本研究で実装する AI の基盤技術となる人工知能，ニューラルネットワーク，ディープラーニング，再帰的ニューラルネットワークについて説明する．

第 2 章　第 1 節　人工知能

人工知能（AI:Articial Intelligence）とは，人間が行う「知的活動」をコンピュータプログラムで実現することである．「知的活動」とは，脳で考えて実行をする活動全般のことである．

人工知能には大きく二つに分かれており，複数の知的活動を行うことが可能な汎用型 AI(AGI:Artifical General Intelligence) と一つの知的活動を専門で行う特化型 AI(ANI:Artificial Narrow Intelligence)に分けられる．現在は，特化型 AI が主に使用されており，「絵を描く」や「翻訳をする」や「ゲームをする」などの用途が限定されている．[1]人工知能のイメージを図 1 に示す．

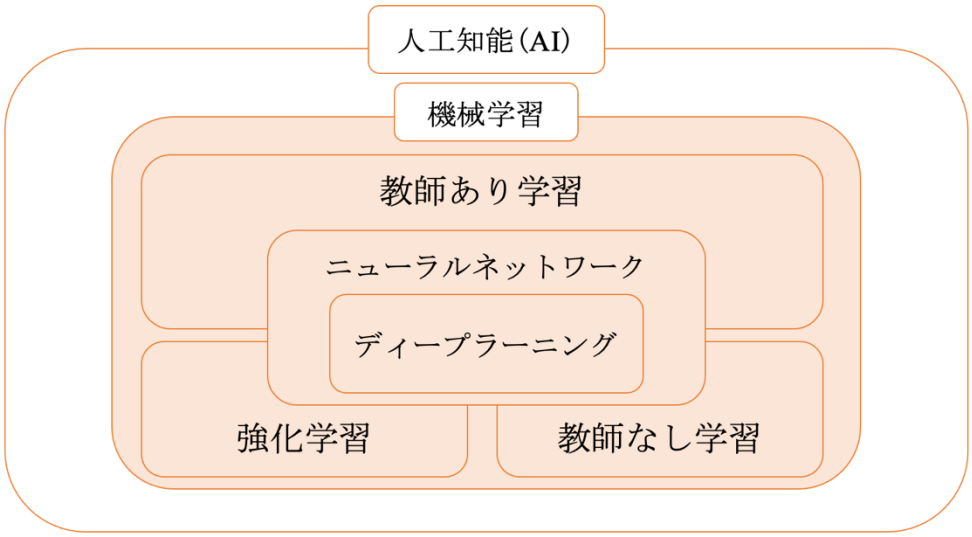

図 1　人工知能のイメージ

人工知能には複数の学習方法が存在する．その中で，様々なデータを用いて機械的に学習を行うことを機械学習と呼ばれる．例として，画像認識や音声認識を用いた翻訳機能が挙げられる．機械学習の学習方法は図 1 のように，「教師あり学習」，「教師なし学習」，「強化学習」の三つに分けることができる．本節では，本研究が回帰するため教師あり学習に絞り説明する．

教師あり学習とは，問題と正解をセットにした教師データから学習を行い，問題から正しく答えを導き出すようにする手法である．正解となるデータで学習を行うため，他の学習方法と比べると学習速度が速く学習精度も高い．しかし正解が存在しない問題に対して，学習を行うことができないことや，用意した問題に正解データを付与するアノテーションなどを行う必要があり，問題と正解のデータを作成する工程に時間がかかってしまう．教師あり学習のイメージを図 2 に示す．

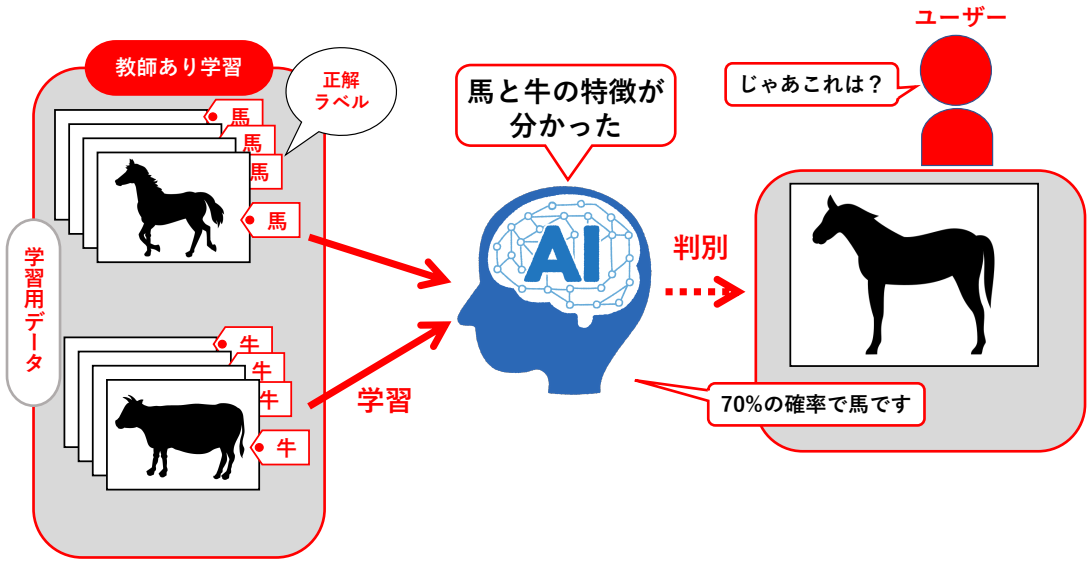

図 2　教師あり学習のイメージ

教師あり学習には，学習用データからどのグループに属するか予測する「分類」と，入力から具体的な数値を予測する「回帰」の二つにタスクを分けることができる．

分類の主な目的は，分析をしたいデータが所属するクラス分けを予測することである．その中で，「はい」や「いいえ」のように予測対象のクラス数が二つの場合を二値分類と呼び，予測対象のクラス数が二つ以上の場合を多値分類と呼ぶ．例としては，ある動物の画像に対して何の動物かを識別するなどが挙げられる．

回帰の主な目的は，教師データとなる連続する値の傾向をもとに予測を行うことである．回帰には，要因となる数値を「説明変数」，結果となる数値を「目的変数」といい，「説明変数」が一つの場合を「単回帰分析」，複数の場合を「重回帰分析」と呼ぶ．例として，今までの株価の情報をもとに未来の株価を予測するなどが挙げられる．[2]分類と回帰のイメージを図 3 に示す．

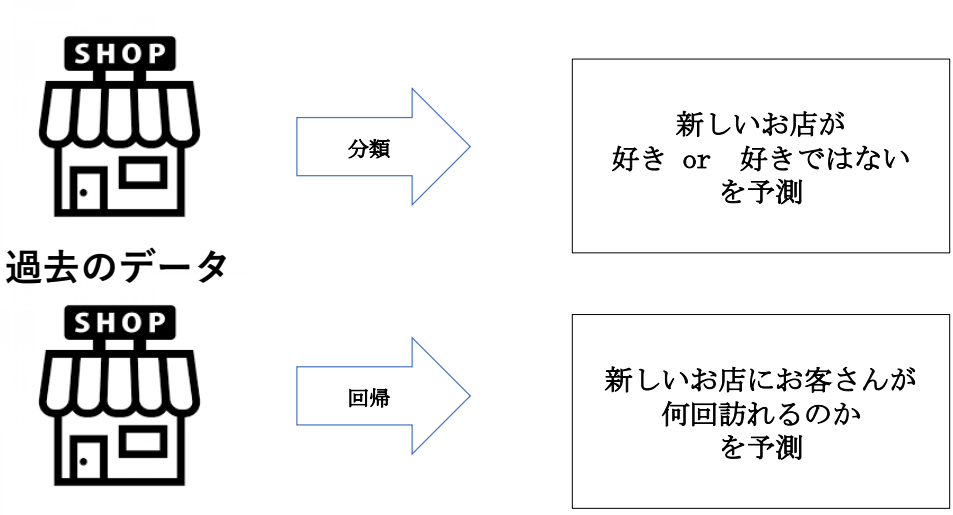

図 3　分類と回帰のイメージ

第２章　第２節　ニューラルネットワーク[3]

生物の脳を構成する神経細胞をニューロンと呼ぶ. シナプスという情報伝達のための接触構造が無数に互いに繋がりあいネットワークを構成している．シナプスは，複数のシナプスから入力を受け，それぞれの入力を評価し，結果をもとに発火するかの決定づけを行う．このような働きを持つシナプスの集まりのネットワーク構造により，脳の学習や記憶といったメカニズムが実現されていると考えられている. ニューラルネットワークは，このような脳のネットワークの働きを数式でモデリングすることで，人の学習や記憶といったメカニズムを再現する技術である.

この脳の神経回路を人工的に再現したアルゴリズムをパーセプトロンと呼ぶ. パーセプトロンのアルゴリズムは，複数の信号を受け取ったときに，一つの信号を出力するアルゴリズムである．パーセプトロンを最小構成したものを単純パーセプトロンと呼ぶ. パーセプトロンの入出力関係を(1) (2), 図 4 に示す.

$$y = \begin{cases} 0(w_1x_1 + w_2x_2 \cdots w_nx_n \leq \theta) \\ 1(w_1x_1 + w_2x_2 \cdots w_nx_n > \theta) \end{cases} \tag{1}$$

$$y = \begin{cases} 0(\theta + w_1x_1 + w_2x_2 \cdots w_nx_n \leq 0) \\ 1(\theta + w_1x_1 + w_2x_2 \cdots w_nx_n > 0) \end{cases} \tag{2}$$

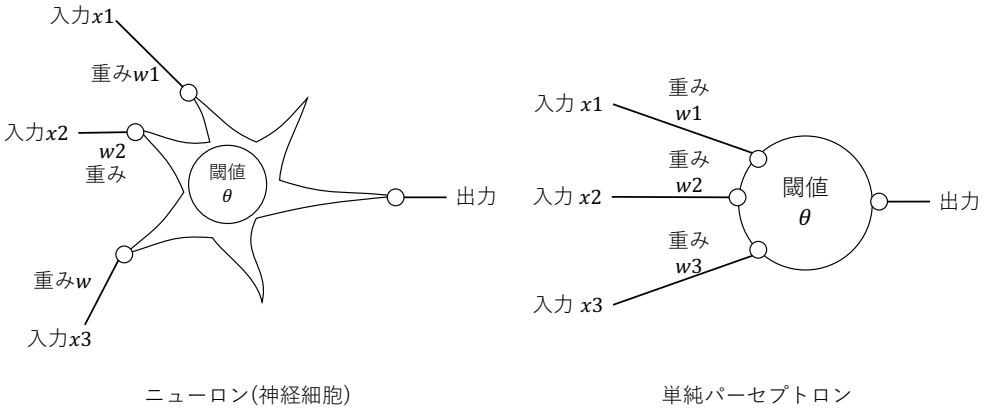

図 4　ニューロンと単純パーセプトロン

ここで，yはパーセプトロンの出力，nはパーセプトロンへの入力の数，x_nはn番目の入力信号，w_nはn番目の入力に対する重み，θは閾値である．重みwとは，入力信号であるxの重要度を決めるパラメータである．θはバイアスと呼び，出力yが 1 を出力する度合いを調節するパラメータであり，重みとは異なった働きをする．

図 9 の単純パーセプトロンには, 出力を任意に変形させるための関数である活性化関数と呼ばれるものが設定されている．パーセプトロンで使用される活性化関数は，ステップ関数である．ステップ関数に，パーセプトロンの出力を入力することでパーセプトロンは発火している，発火していない，の二つの状態を表現していた．ステップ関数を(3)，図 5 に示す．

$$\mathrm{step}(x) = \begin{cases} 1(x > 0) \\ 0(x \leq 0) \end{cases} \tag{3}$$

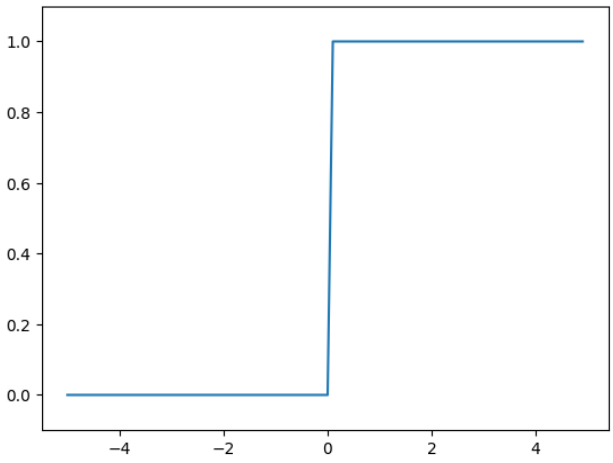

図 5　ステップ関数

単純パーセプトロンには，非線形の問題を解くことができない問題があり，この問題をパーセプトロンの層を重ねることで解決した．二層以上のパーセプトロンを多層パーセプトロンという．多層パーセプトロンを図 6 に示す．

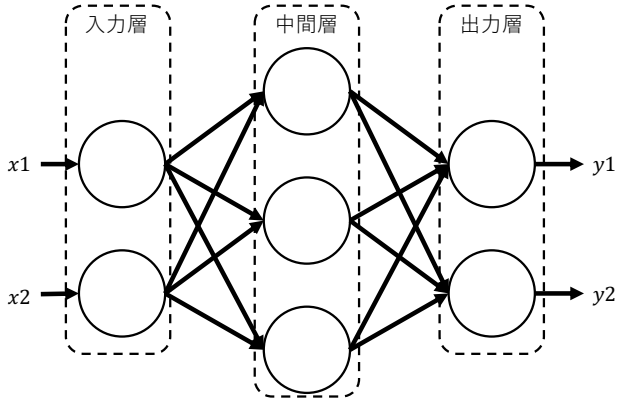

図 6　多層パーセプトロン

ステップ関数は，連続した関数ではないため，微分することができない．また線形関数では，どんなに層を深くしても，同じことを行う「隠れ層のないネットワーク」が必ず存在してしまうためニューラルネットワークの学習をできない．線形関数 $\boldsymbol{h(x) = cx}$を活性化関数として，$\boldsymbol{y(x) = h(h(h(x)))}$を行う計算を三層のネットワークに対応させ考えたときに，別の関数$\boldsymbol{a = c^3}$としたとき$\boldsymbol{y(x) = ax}$の一回の計算で同じことができてしまう．この問題を解決するために，シグモイド関数などの連続の非線形関数を活性化関数として用いる．シグモイド関数を(4)，図 7 に示す．

$$\mathrm{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

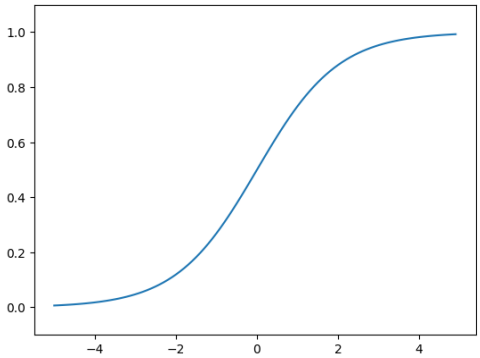

図 7　シグモイド関数

ニューラルネットワークは，複数のパーセプトロンを縦横方向に組み合わせたものである．縦方向に組み合わされた複数のパーセプトロンの入出力関係を(5)，図 8 に示す．また活性化関数のイメージを図 9 に示す．

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b} \tag{5}$$

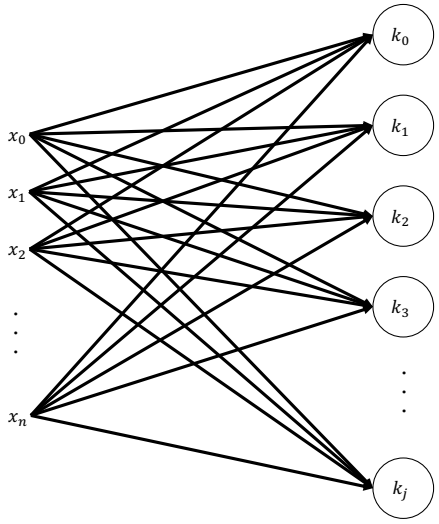

図 8　縦方向に組み合わされた複数のパーセプトロンの入出力関係

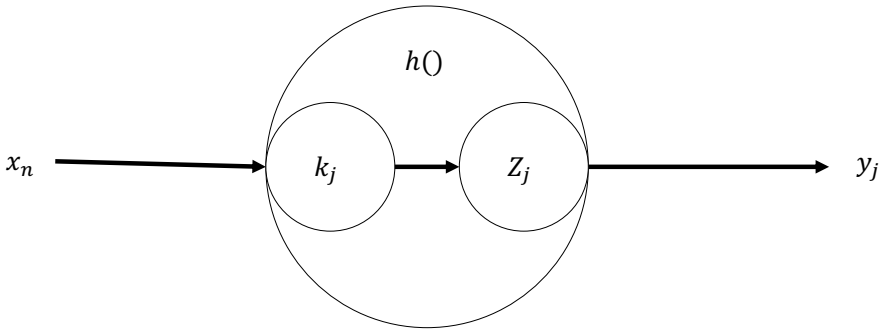

図 9　活性化関数のイメージ

ここで，x_nはn番目の入力，k_jはj番目のパーセプトロン，$h()$は活性化関数，Z_jは活性化関数を通したj番目のパーセプトロン，y_jはj番目の出力，$W \in \mathbb{R}^{j \times n}$は$k_0$から$k_j$のパーセプトロン，$W \in X^n$は入力のベクトル，$b \in \mathbb{R}^j$は$k_0$から$k_j$のパーセプトロンの閾値ベクトルである．図 13 の縦方向にパーセプトロンを組み合わせたものを 1 層とし，この層を何層にも横に重ねたものがニューラルネットワークである. ニューラルネットワークの例とその入出力関係を(6)，図 10 に示す.

$$\mathbf{y} = \mathrm{h}(\mathbf{W}_1 h(\mathbf{W}_0 \cdot \mathbf{x} + b_\mathbf{0}) + b_1) \tag{6}$$

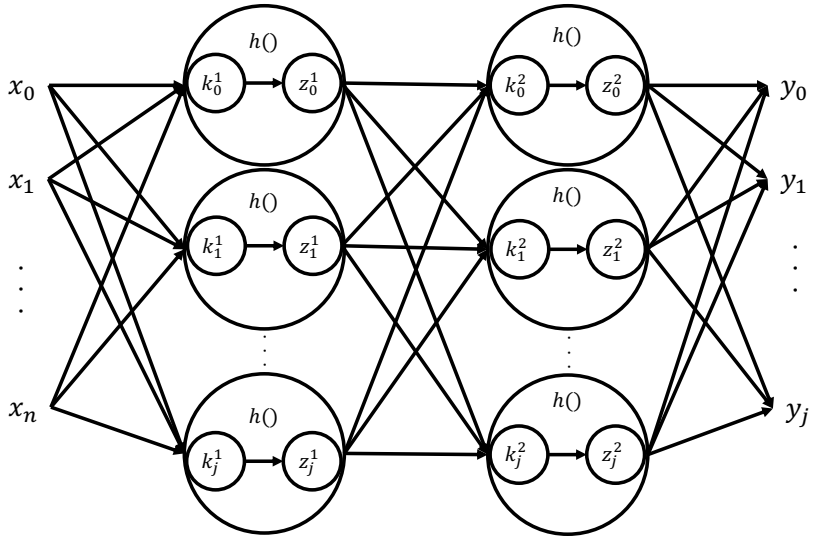

図 10　ニューラルネットワークの例

ここで，jは各層に含まれるパーセプトロンの数，$\mathbf{W}_i$はi層目のパーセプトロンの重み行列，$\mathbf{y} = \mathbb{R}^j$はニューラルネットワークの層の数である．

ニューラルネットワークの層が, 多層で構成されているニューラルネットワークをディープニューラルネットという．

ニューラルネットワークは, 誤差関数を最小化するように重みや閾値を更新することで，学習精度を上げている．代表的な誤差関数である平均二乗誤差を(7)に示す.

$$MSE(\mathbf{W}, x, t) = \frac{1}{n}\sum_{i=0}^{n-1}(t_i - y_i)^2 \tag{7}$$

ここで，$\mathbf{W}$はニューラルネットワークのパラメータのベクトル，xは入力，tはxに対する教師信号, nは入力データの総数, yはニューラルネットワークの予測した推論結果である．この誤差関数を最小化するためのアルゴリズムを学習アルゴリズムという. 代表的な学習アルゴリズムである確率的勾配降下法を(8)，図 11 に示す.

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \eta\frac{\partial \boldsymbol{L}}{\partial \mathbf{W}_t} \tag{8}$$

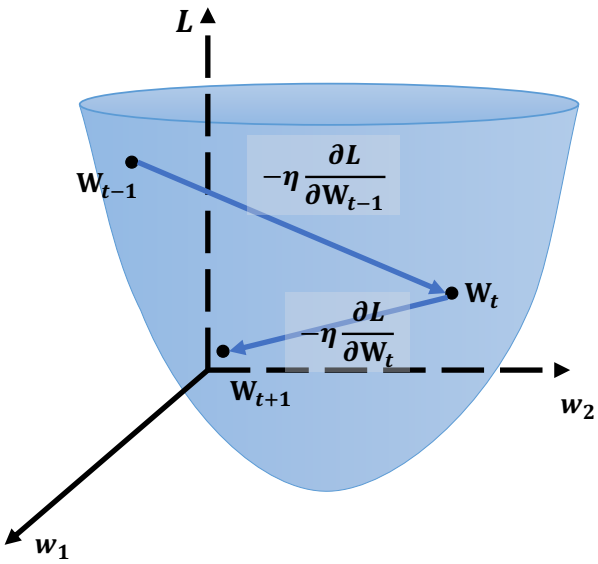

図 11　確率的勾配降下法のイメージ

ここで，$\mathbf{W}_{t+1}$は$t+1$番目のニューラルネットワークのパラメータのベクトル，$\mathbf{W}_t$はt番目のニューラルネットワークのパラメータのベクトル，ηは次の学習係数の更新数，Lは損失関数である．近年では，確率的勾配降下法から発展した学習アルゴリズムである Adam が使用されている．Adam は学習を高速化するモーメント法と，学習率を適応的に更新する RSMprop を組み合わせたアルゴリズムである．

学習を行う際に，学習データ全てを一度に学習するバッチ学習がある．しかし学習データが大きいとデータの特徴量が平均化されてしまい，データの特徴量が失われてしまう可能性がある．そのために学習データを細かく分けて分割し学習していくミニバッチ学習法と学習データを一つずつ学習するオンライン学習法がある．バッチ学習，ミニバッチ学習，オンライン学習のイメージ図を図 12 に示す．

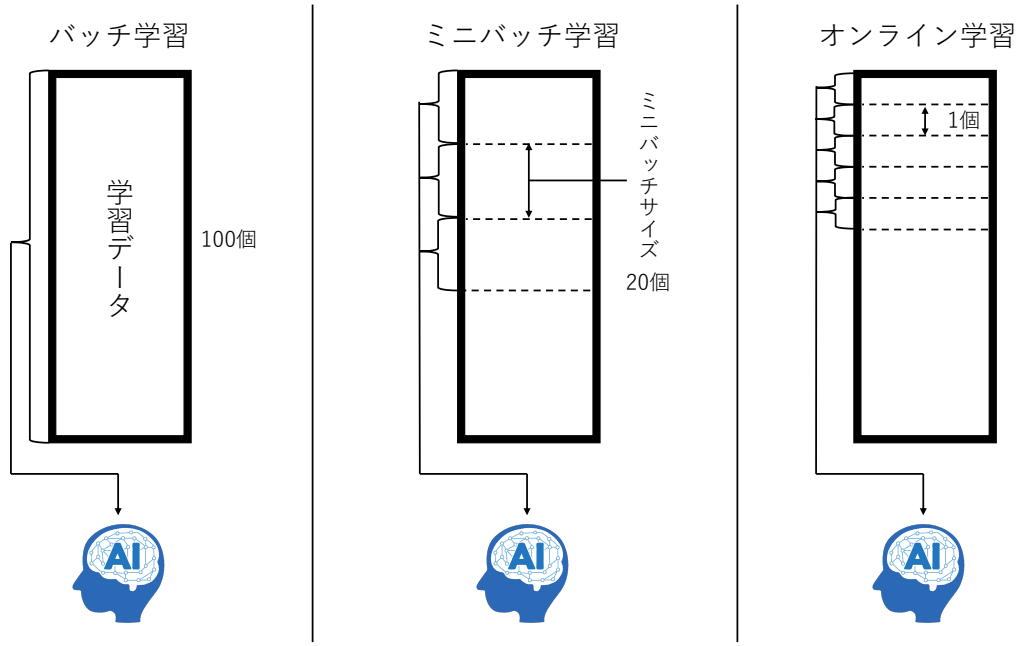

図 12　バッチ学習，ミニバッチ学習，オンライン学習

第２章　第 3 節　再帰的ニューラルネットワーク[4]

再帰的ニューラルネットワーク（Recurrent Neural Network，RNN）は，過去のニューラルネットワークの出力を再帰的にニューラルネットワークに入力する．これにより時系列データの性質を十分に学習することができる．RNN の入出力関係を(9)，図 13 に示す．

$$\mathbf{y}_t = \mathbf{h}_t = \tanh\left((\mathbf{W}_h \cdot h_{t-1} + b_h) \oplus (\mathbf{W}_x \cdot \mathbf{x}_t + b_x)\right) \tag{9}$$

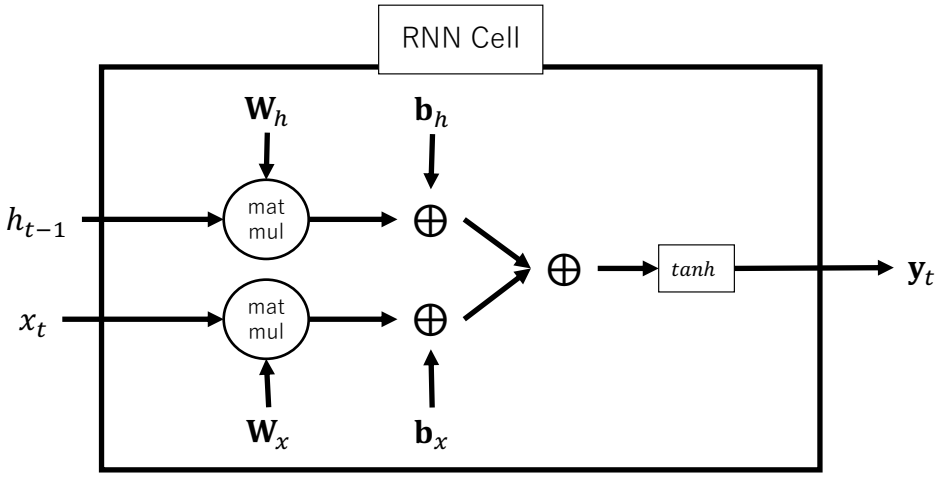

図 13　RNN の入出力関係

ここで，$\mathbf{y}_t$は時刻tにおける RNN における出力，$\mathbf{h}_t$時刻tにおける隠れ層の状態，$\mathbf{x}_t$は時刻tにおける入力，$\mathbf{W}_h$はhに対する重み，b_hはhに対する閾値，$\mathbf{W}_x$はxに対する重み，b_xはxに対する閾値，$\oplus$はベクトルの要素ごとの和である．

RNN は入出力の数によって，構造を三種類に分類することができる．三種類の RNN の構造について図 14 に示す．

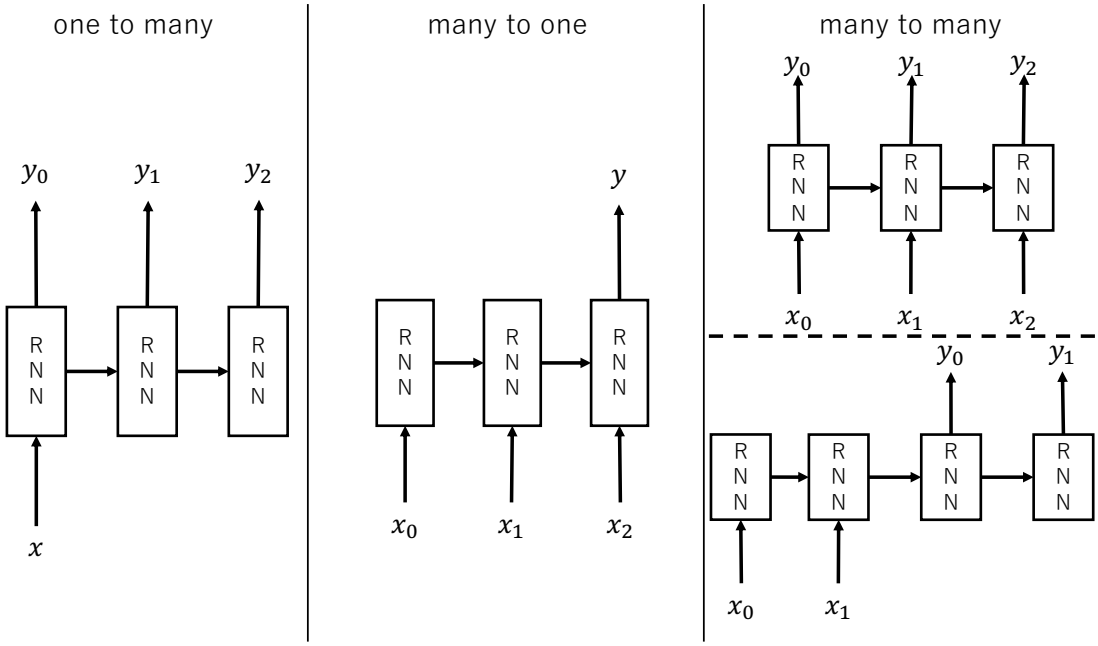

図 14　3 種類の RNN の構造

one to many 構造の RNN は，入力が一つであり，出力が複数である．画像から自然文を出力する画像キャプション等に用いられる．many to one 構造の RNN は，入力が複数あり，出力は一つである．自然言語処理の感情分析等に用いられている．many to many 構造の RNN は，入力と出力が共に複数ある．また遅延モデルと同期モデルの二種類がある．図 14 の many to many 構造の下が遅延モデルであり，言語の翻訳等に用いられる．上の同期モデルは, 動画の各フレームをラベル付けするなど動画分類等で用いられている．

RNN は，過去の情報を加算することで過去の情報を保持している．そのため，長期の過去の情報を保持することができない問題がある．そのため Long-short Term Memory（LSTM）では，ゲートと呼ばれる機構を組み込むことで，記憶の保持量を決めることが可能となった．LSTM には，複数のゲートを組み込みしており，過去の記憶情報をどれだけ忘れるかを判断するゲート（Forget Gate），新たな記憶をどれだけ記憶するかを判断するゲート（Input Gate），記憶している情報をどれだけ出力するか判断するゲート（Output Gate），記憶情報であるセルを RNN に追加することで長期の過去の情報を保持することが可能となった. LSTM の入出力関係を（10.1），（10.2），（10.3），（10.4），（10.5），（10.6），図 15 に示す.

$$\mathbf{f} = \mathrm{sigmoid}\left(\left(\mathbf{W}_h^{\mathbf{f}} \cdot \mathbf{h}_{t-1} + \mathbf{b}_h^{\mathbf{f}}\right) \oplus \left(\mathbf{W}_x^{\mathbf{f}} \cdot \mathbf{x}_t + \mathbf{b}_x^{\mathbf{f}}\right)\right) \tag{10.1}$$

$$\mathbf{i} = \mathrm{sigmoid}\left(\left(\mathbf{W}_h^{\mathbf{i}} \cdot \mathbf{h}_{t-1} + \mathbf{b}_h^{\mathbf{i}}\right) \oplus \left(\mathbf{W}_x^{\mathbf{i}} \cdot \mathbf{x}_{\mathbf{t}} + \mathbf{b}_x^{\mathbf{i}}\right)\right) \tag{10.2}$$

$$\mathbf{o} = \mathrm{sigmoid}\left(\left(\mathbf{W}_h^{\mathbf{o}} \cdot \mathbf{h}_{t-1} + \mathbf{b}_h^{\mathbf{o}}\right) \oplus \left(\mathbf{W}_x^{\mathbf{o}} \cdot \mathbf{x}_{\mathbf{t}} + \mathbf{b}_x^{\mathbf{o}}\right)\right) \tag{10.3}$$

$$\tilde{\mathbf{c}} = \tanh\left(\left(\mathbf{W}_h^{\tilde{\mathbf{c}}} \cdot \boldsymbol{h}_{\mathrm{t}-1} + \mathbf{b}_h^{\tilde{\mathbf{c}}}\right) \oplus \left(\mathbf{W}_x^{\tilde{\mathbf{c}}} \cdot \mathbf{x}_t + \mathbf{b}_x^{\tilde{\mathbf{c}}}\right)\right) \tag{10.4}$$

$$\mathbf{c}_t = \left(\mathbf{f} \otimes \mathbf{c}_{t-1}\right) \oplus \left(\mathbf{i} \otimes \tilde{\mathbf{c}}\right) \tag{10.5}$$

$$\mathbf{y}_t = \mathbf{h}_t = \mathbf{o} \otimes \mathbf{c}_t \tag{10.6}$$

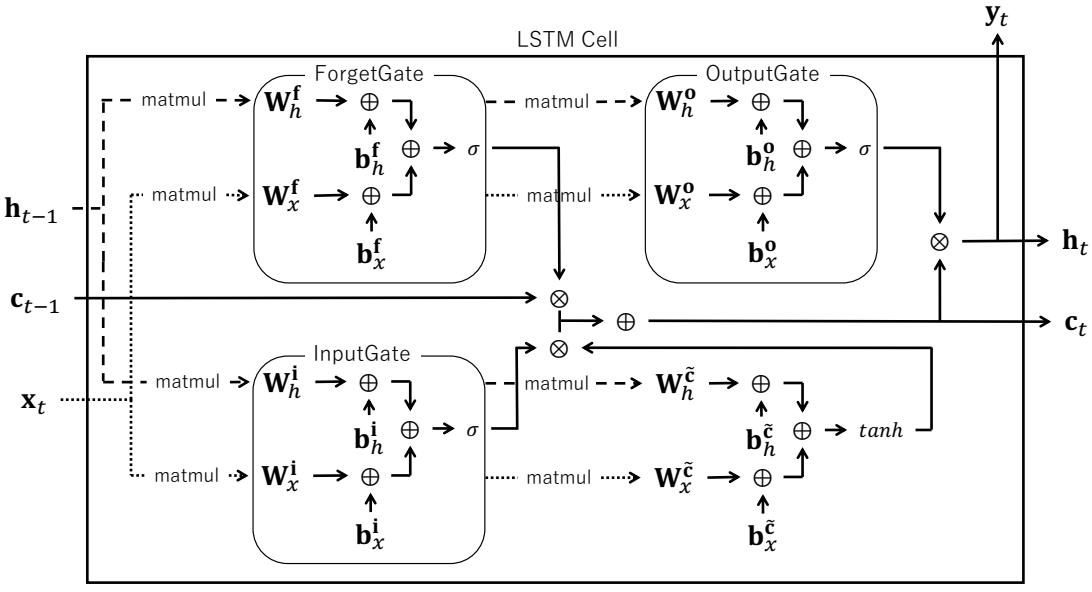

図 15　LSTM の入出力関係

ここで，$\mathbf{f}$は忘却ゲートの出力，$\mathbf{W}_h^{\mathbf{f}}$，$\mathbf{b}_h^{\mathbf{f}}$は$h$に対する忘却ゲートの重みと閾値，$\mathbf{W}_x^{\mathbf{f}}$，$\mathbf{b}_x^{\mathbf{f}}$は$x$に対する忘却ゲートの重みと閾値，$\mathbf{i}$は入力ゲートの出力，$\mathbf{W}_h^{\mathbf{i}}$，$\mathbf{b}_h^{\mathbf{i}}$は$h$に対する入力ゲートの重みと閾値，$\mathbf{W}_x^{\mathbf{i}}$，$\mathbf{b}_x^{\mathbf{i}}$は$x$に対する入力ゲートの重みと閾値，$\mathbf{o}$は出力ゲートの出力，$\mathbf{W}_h^{\mathbf{o}}$，$\mathbf{b}_h^{\mathbf{o}}$は$h$に対する入力ゲートの重みと閾値，$\mathbf{W}_x^{\mathbf{o}}$，$\mathbf{b}_x^{\mathbf{o}}$は$x$に対する入力ゲートの重みと閾値，$\mathbf{c}_t$は時刻$t$におけるセルゲートの出力，$\mathbf{W}_h^{\tilde{\mathbf{c}}}$，$\mathbf{b}_h^{\tilde{\mathbf{c}}}$は$h$に対するセルゲートの重みと閾値，$\mathbf{W}_x^{\tilde{\mathbf{c}}}$，$\mathbf{b}_x^{\tilde{\mathbf{c}}}$は$x$に対するセルゲートの重みと閾値，$\mathbf{y}_t$は時刻$t$における LSTM の出力，$\mathbf{h}_t$は時刻$t$における LSTM の隠れ層の状態，$\otimes$はベクトルの要素ごとの積である．tanh 関数を(11)，図 16 に示す．

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{11}$$

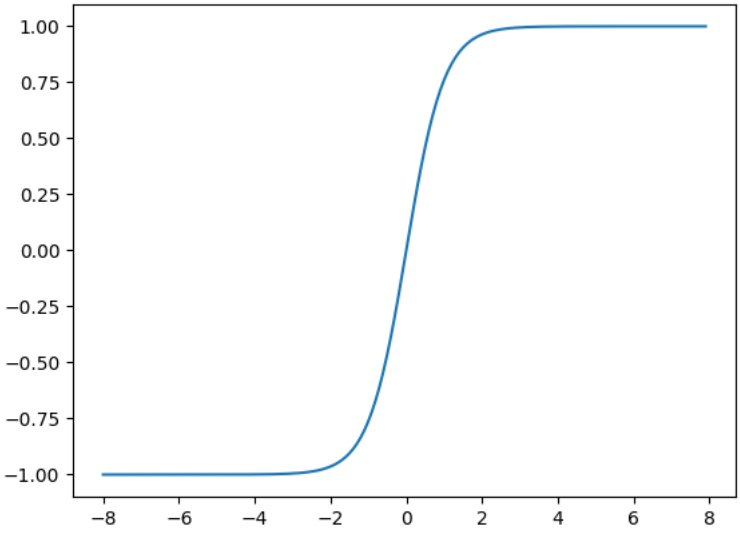

図 16　tanh 関数

第 3 章　Steam Game の接続数に対する時系列予測

本研究では，プラットフォーム Steam[5]で販売されているオンラインゲームの接続数を用いて，来月の接続数と今月の接続数の差を求めることで，ゲームの成長傾向やエンドユーザーに対しての需要について知ることができ，ゲームの品質の向上，サーバーに負荷のかかる時期の特定，サーバーダウン対策の起点となり，ゲーム開発者がエンドユーザーにあったコンテンツの提供を行うことができると考える．そこで，本研究では，Kaggle のデータセットから時系列データへと変換したものを使用し，この学習データを用いて時系列予測を行う．また RNN や LSTM など様々なニューラルネットワークを検証し，オンラインゲームの来月の接続数に対する時系列予測をする学習モデルを提案する．提案手法のイメージを図 17 に示す．

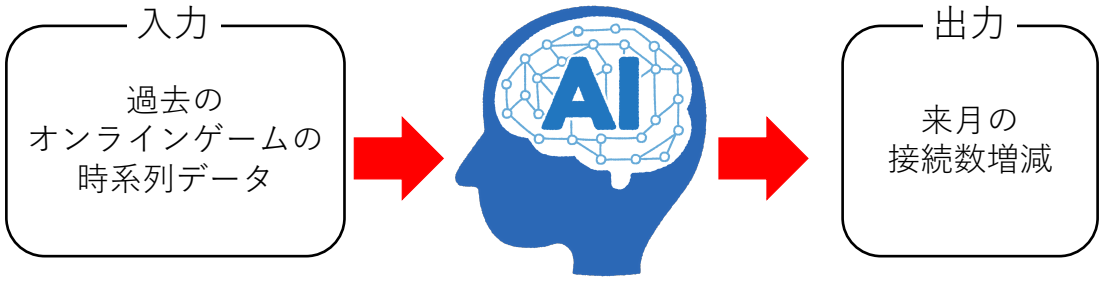

図 17　提案手法のイメージ

第 4 章　実験

本章では，本研究で扱ったデータセット，作成したモデル構造，作成したモデルでの実験結果を示す．

第 4 章　第 1 節　データセットの作成

本研究では，Kaggle にある Popularity of games on Steam[6]と Steam DB[7]から二つのデータ price.csv，Steam.csv を用いてデータセットを作成する．Popularity of games on Steam のデータ項目を表 1 に示す．

表 1　Popularity of games on Steam のデータ項目

項目名	内容
gamename	ゲーム名
year	年
month	月
avg	各ゲームの平均接続数
gain	前月と比較した 平均接続数の差
peak	各ゲームの最大接続数
avg_peak_perc	最大値(平均/ピーク)における 平均の割合(%)

Popularity of games on Steam は，Steam DB にある表を 2021 年 2 月時点でスクレイピングしたデータセットであるため Steam DB と Popularity of games on Steam は，同じ形のデータセットである．Steam DB は，現在でも更新されているため 2021 年 2 月以降のデータが存在する．price.csv は，価格が変更した日，変更された日の価格が要素として存在する．price.csv のデータ項目を表 2 に示す．

表 2　price.csv のデータ項目

項目名	内容
Date Time	価格の変更された日
Final price	変更された価格

表 2 では，価格の変更された日しかデータがないため，月ごとのデータセットである Popularity of games on Steam と整合性が取れないため，表 2 から月ごとの平均の価格を求めた month_avg_price.csv データセットを作成した．month_avg_price.csv のデータ項目を表 3 に示す．

表 3　 month_avg_price.csv のデータ項目

項目名	内容
year	年
month	月
avg_price	月の平均の価格

Steam.csv は，日付，Steam 全体の最大接続数，平均接続数，Steam の中のゲームを行っている数が要素として存在する．Steam.csv のデータ項目を表 4 に示す．

表 4　Steam.csv のデータ項目

項目名	内容
Date Time	日付
Users	Steam 全体の最大接続数
Average Users	Steam の平均接続数
In-Game	Steam の中のゲームを行っている数

Steam.csv は，price.csv と同様に日ごとのデータセットになるので，Popularity of games on Steam と整合性が取れない．よって，Users の月ごとの値を求めた Steam_online.csv データセットを作成した．Steam_online.csv のデータ項目を表 5 に示す．

表 5　Steam_online のデータ項目

項目名	内容
year	年
month	月
Steam_online	Steam 全体の最大接続数

本研究では，表 1 の各オンラインゲームにデータセットを作成している．例として表 1 のデータセットの先頭にあるオンラインゲーム Counter-Strike:Global Offensive の時に作成する．CSGO_datas.csv データセットを表 6 に示す.

表 6　CSGO_datas.csv のデータ項目

項目名	内容
year	年
month	月
peak	ゲームの最大接続数
avg_price	月の平均の価格
Steam_online	Steam 全体の最大接続数

本研究では，表 6 の CSGO_datas.csv の peak，avg_price，Steam_online を入力信号とする．また，表 6 の CSGO.datas_csv の他に二つのゲーム Dota2_datas_csv，Rust_datas_csv を使用する．ゲームによってデータ数が異なるため，約 10 年分のデータが存在するゲームを選択している．peak，avg_price，Steam_online は，一月ごとのデータになっている．本研究では，peak，avg_price，Steam_online の一月ごとのデータを時系列データとする．教師信号となる接続数の増減を求める数式を(12)に示す.

$$next_{gainpercent} = \left(\frac{next_{peak}}{now_{peak}}\right) \times 100 - 100 \qquad (12)$$

ここで，$next_{gainpercent}$は来月の接続数の増減，$next_{peak}$は来月の最大接続数，now_{peak}は今月の最大接続数である．$next_{gainpercent}$は，扱う時系列データの長さと同じだけ教師信号が必要となる．データの入出力のイメージを表 7 に示す.

表 7　データの入出力のイメージ

入力(一部)	出力(一部)
[[1 月のデータ], [2 月のデータ], …[5 月のデータ]]	[2 月の$next_{gainpercent}$, 3 月の$next_{gainpercent}$, … 6 月の$next_{gainpercent}$]
[[2 月のデータ], [3 月のデータ], …[6 月のデータ]]	[3 月の$next_{gainpercent}$, 4 月の$next_{gainpercent}$, … 7 月の$next_{gainpercent}$]
[[3 月のデータ], [4 月のデータ] …[7 月のデータ]]	[4 月の$next_{gainpercent}$, 5 月の$next_{gainpercent}$, … 8 月の$next_{gainpercent}$]

ここで，表 7 の時系列tの長さは 4 とする．1 月のデータなどは一月ごとのデータであり，その中には，入力信号となる peak，avg_price，Steam_online の三つの要素が入っている．入力信号と教師信号の値は，値の大きさが大きく，そのままの値を入れてしまうと，推論結果にも大きなズレが生じてしまう．そのため標準化を行うことにより，勾配爆発を起こしにくくし, 推論結果の誤差を少なくしている. 標準化の数式を(13)に示す.

$$z = \frac{x - \bar{x}}{\sigma} \tag{13}$$

ここで，zは標準化したデータの要素，xはデータの要素，$\bar{x}$は平均化したデータ，σは標準偏差を求めたデータである．標準化を行うことで，平均が 0 分散が 1 となり，データの偏りを少なくしている．

第 4 章　第 2 節　モデルの作成

本節では，本研究で実装する Steam Game の接続数の予測を行うモデル構築を行う．モデル構築は， RNN を用いた一層のモデル構造，LSTM を用いた一層のモデル構造，RNN を用いた多層のモデル構造，LSTM を用いた多層のモデル構造の四つのモデルを作成した．各モデル構造を図 18 に示す．

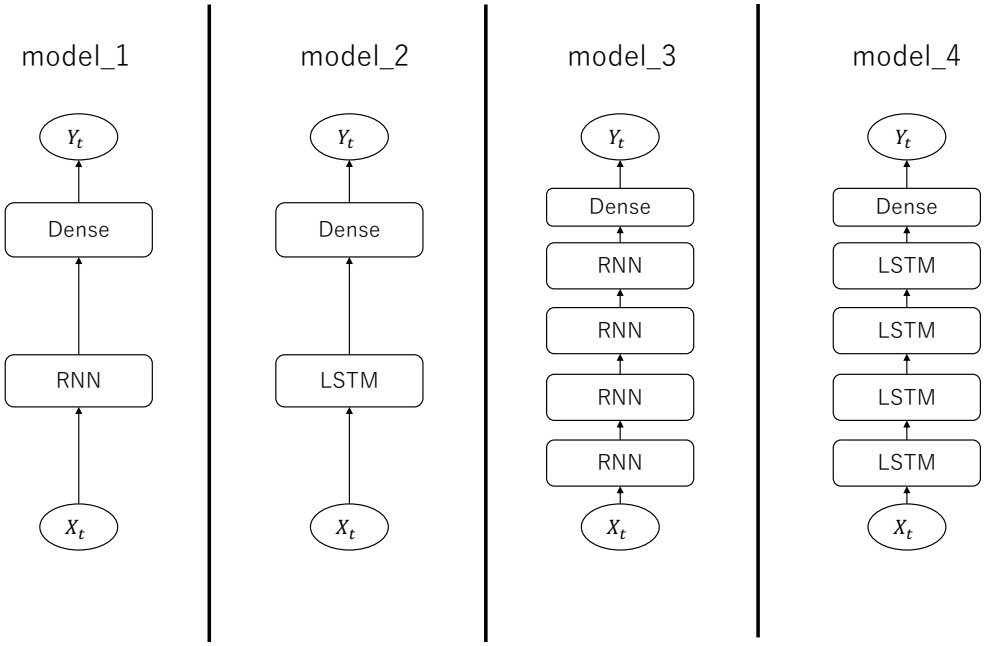

図 18　実験に使用するモデル構造

ここで，X_tは時系列t番目の入力信号，Y_tは時系列t番目の教師信号である．作成した四つのモデルを図 18 のように model_1, 2, 3, 4 と定義する．また各モデルの構造を表 8 に示す．

表 8　各モデルの構造

Layer name	Output Shape
model_1	
Input Shape	85x3x3
RNN	85x3x3
Dense	85x3x1
model_2	
Input Shape	85x3x3
LSTM	85x3x3
Dense	85x3x1
model_3	
Input Shape	85x3x3
RNN	85x3x3
RNN	85x3x3
RNN	85x3x3
RNN	85x3x3
Dense	85x3x1
model_4	
Input Shape	85x3x3
LSTM	85x3x3
LSTM	85x3x3
LSTM	85x3x3
LSTM	85x3x3
Dense	85x3x1

表 8 の OutShape の形は，$datas \times times \times feature$の形となっている．$datas$は扱う学習データの数，$times$は時系列の長さここでは過去何ヶ月を使用するかとする，$feature$は要素の数である．表 8 のモデル構造は$times$=3 である．

第 4 章　第 3 節　モデルの学習結果

前節より作成したモデルの実験結果と各モデルの推論結果のグラフを表 9, 10, 11, 12 を示す.

表 9　model_1　実験結果

times	val_loss	val_MSE
3	0.4388	0.6574
12	0.2493	0.6534
24	0.1373	0.6536

表 10　model_2　実験結果

times	val_loss	val_MSE
3	0.4067	0.6560
12	0.2919	0.6512
24	0.2892	0.6511

表 11　model_3　実験結果

times	val_loss	val_MSE
3	0.4178	0.6518
12	0.1616	0.6526
24	0.1168	0.6584

表 12　model_4　実験結果

times	val_loss	val_MSE
3	0.3581	0.6523
12	0.2210	0.6509
24	0.1404	0.6505

表 9, 10, 11, 12 の実験結果で，各モデルの*times*=24 の時，最も良い予測精度を確認することができた．各モデルの推論結果を図 19, 20, 21, 22 に示す.

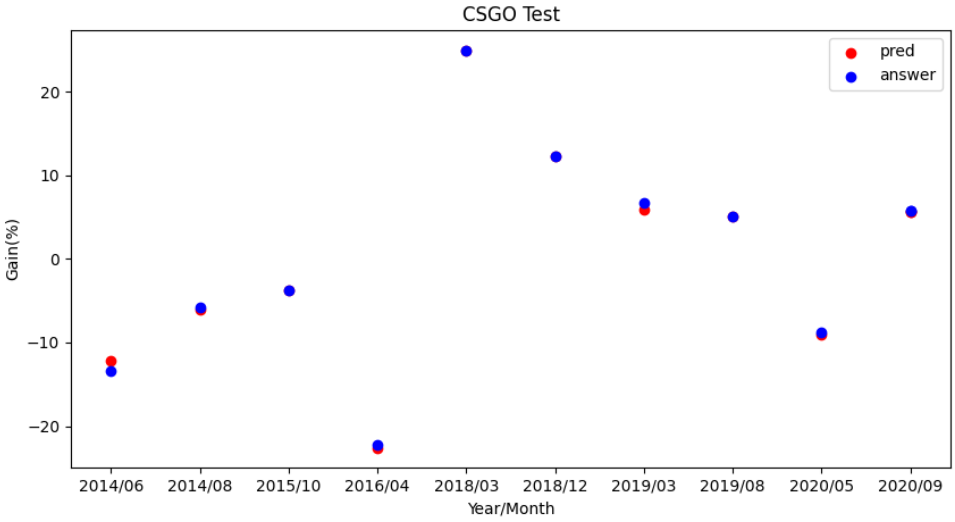

図 19　model_1　推論結果

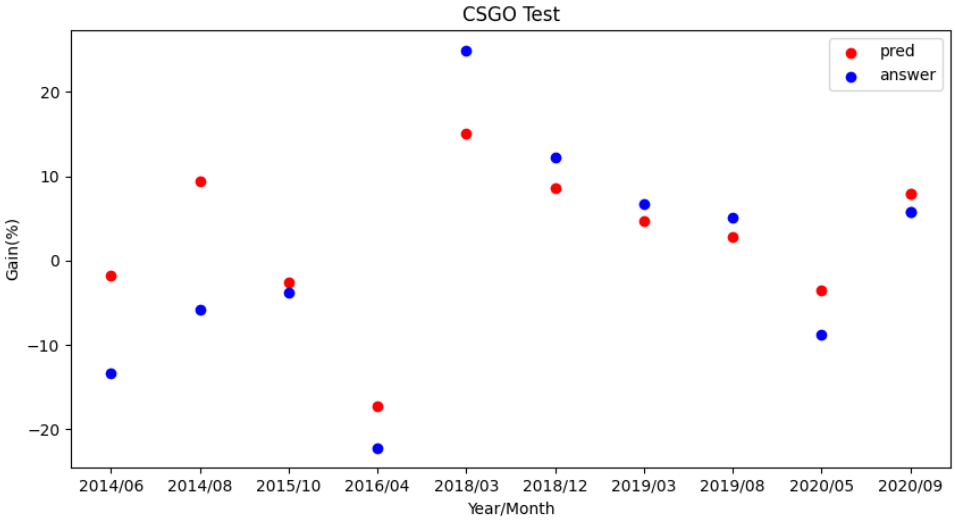

図 20　model_2　推論結果

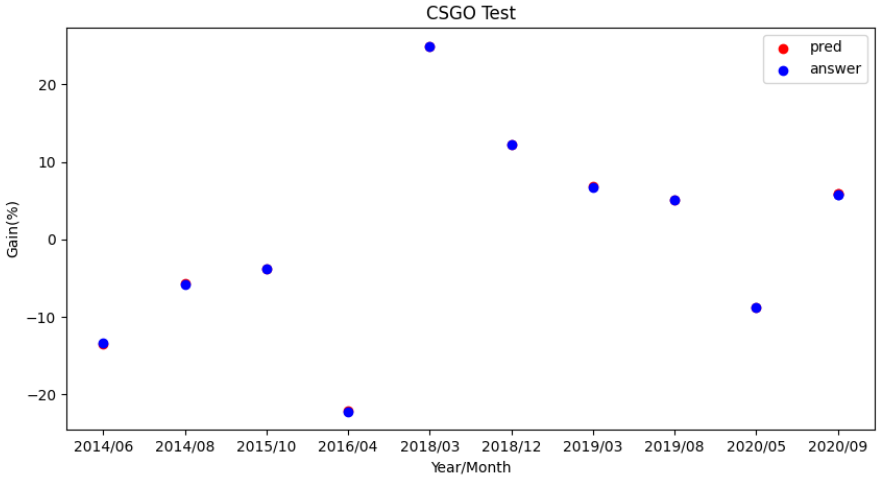

図 21　model_3　推論結果

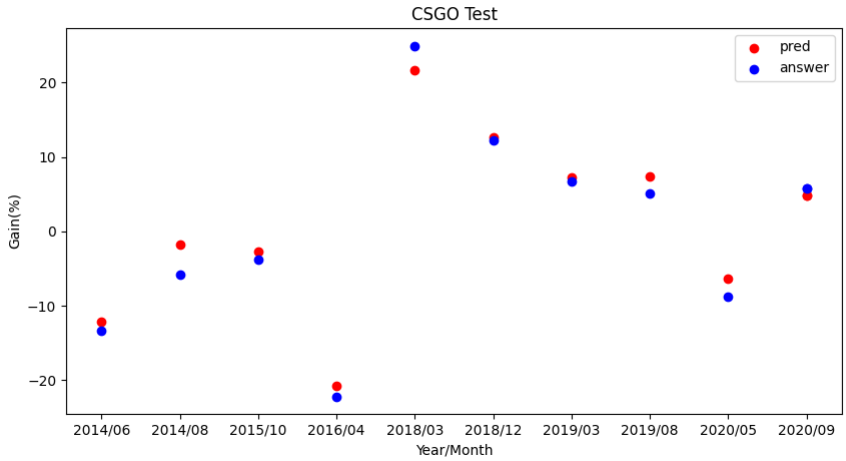

図 22　model_4　推論結果

ここで，図 1 のy軸は gain，x軸は年・月である．gain は標準化した値から割合にスケールを戻している．実験結果から LSTM よりも RNN の方が予測精度が良いことがわかった. 予測を行うモデルは, 一層よりも多層の方が, 予測精度が向上することがわかった. また多層のモデルは, 時系列の長さを短くしても精度を保つことが確認できた. model_3 の$times$=12 の時の予測結果を図 23 に示す.

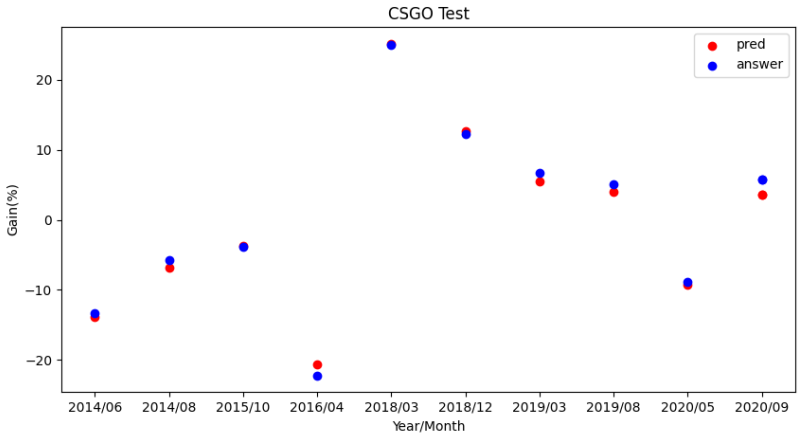

図 23　model_3　$\boldsymbol{times = 12}$

図 19 と見比べると時系列長を 12 ヶ月分減らしても大幅に推論結果が異なることなく精度が保たれていることが確認できる．学習データに依存したモデルの可能性があるため，Dota2_datas_csv，Rust_datas_csv の二つのゲームを使って推論を行った．

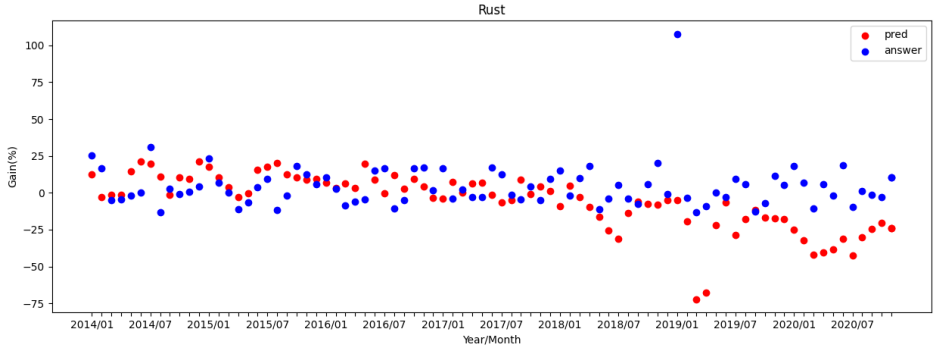

図 24　Rust_datas_csv　推論結果

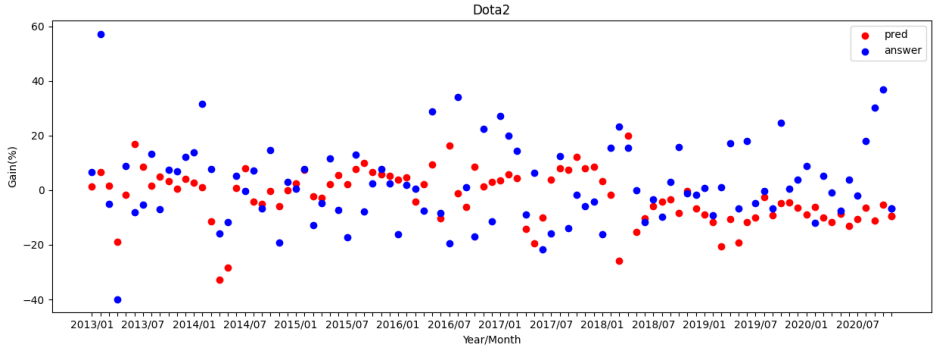

図 25　Dota2_datas_csv　推論結果

図 23, 24 は，精度が最も良い$times = 24$の model_3 を用いている．図 23, 24 から学習データに依存したモデル構造であることを確認できた．

考察として，データセットが 1 ヶ月ではデータの総数が少ないため精度を上げるには限界がある．よって 1 日ごとのデータセットにすることで，より精度の高い予測をできるモデル構築が可能になると考える．また，データセットが各ゲームごとに作成されているため，モデルが学習データに依存していると考える．全てのゲームのデータが集約されたデータセットを作成し，学習させることで学習データに依存しないモデル構築をすることができると考える．LSTM より RNN の予測精度が良かった理由として LSTM の忘却ゲートが原因となっていると考えられる．時系列の長さを長くすることで，最後の一ヶ月の予測を行う際に，過去の情報が忘れすぎて新しい情報に更新してしまい，予測精度が落ちてしまうと考える．また，RNN の精度が良いのは，24×3次元の大きさであれば問題なく過去の情報を引き継いで学習することができていると考える．

第 5 章　まとめ

本研究では，RNN や LSTM を用いたモデル構築を行い，Steam Game の接続数増減の予測を行った．実験結果から，層を重ねることで，時系列データの長さを短くし，予測精度を保持することが確認できた．今後の研究では，LSTM 以外の層を追加して実験を行い，時系列長を短くし，予測精度を保持できるモデル構築を目指す．

謝辞

本研究を進めるにあたり，卒業研究を指導していただいた二宮 洋先生，渡辺 重佳先生，マハブービ シェヘラザード先生にはご指導ご鞭撻を賜りました．感謝申し上げます．

また日頃の議論を通じて知識やアドバイスをいただいた，山富 龍先輩，堀 雄介先輩，学科横断型学修プログラム AI コースの皆様には深謝の意を表します．

最後に，二宮研究室の皆様には，本研究の遂行にあたり多大なご助言，ご協力いただきました．感謝いたします．

参考文献

[1] アイティメディア株式会社，”5 分でわかる人工知能(AI)”，https://atmarkit.itmedia.co.jp/ait/articles/2103/29/news019.html,（最終確認日：2024/02/03）

[2] 総務省統計局，” 第 3 章機械学習（教師あり学習）”，https://www.stat.go.jp/teacher/dl/pdf/c4learn/materials/fourth/dai3.pdf，（最終確認日：2024/02/03）

[3] 斎藤 康毅（2016），ゼロから作る DEEP LEARNING ― PYTHON で学ぶディープラーニングの理論と実装，オライリージャパン

[4] 斎藤 康毅（2016），ゼロから作る DEEP LEARNING ❷ ― 自然言語処理編，オライリージャパン

[5] Steam，“Steam へようこそ”，https://store.steampowered.com/?l=japanese,（最終確認日：2024/02/03）

[6] Kaggle，“Popularity of games on Steam”，https://www.kaggle.com/datasets/michau96/popularity-of-games-on-steam,（最終確認日：2024/02/03）

[7] Steam DB，“Datebase of everything on Steam”，https://steamdb.info/,（最終確認日：2024/02/03）