〔技術報告〕

機械学習による習熟度別クラス分けの検討

中嶋　康博[*1]・境　優一[*2]・中村　理央[*3]
花元　誠一[*4]・西岡　昌幸[*5]・金井　政宏[*1]

Class placement using machine learning

Yasuhiro NAKASHIMA[*1], Yuichi SAKAI[*2], Riou NAKAMURA[*3],
Seiichi HANAMOTO[*4], Masayuki NISHIOKA[*5], Masahiro KANAI[*1]

Abstract

We provided a remedial mathematics class in the first semester and two basic mathematics classes in the second semester for the first-year students of engineering. In these classes, we introduced placement classes to improve the learning environment. In the second semester, students were divided into three classes based on their results in the first semester. The classification of the students can be achieved through machine learning, such as clustering and deep learning. We applied deep learning to our data by using R interface to Keras and discussed the results in this paper.

Key Words：mathematical education, first year education, remedial education, R-language, machine learning

1．緒　　言

本学では１年次の前期に数学のリメディアル教育を，そして後期に微分積分学および線形代数学を開講している．少子化や入学試験の細分化を受けて，学力の多様性が増しており，その影響を緩和するためにいずれの科目でも，習熟度別のクラス分けを行っている．前期科目では入学時点でのクラス分け試験に基づいてクラスを分けているが，後期科目では前期の成績をそれに加味して分けている．前期のクラス分けに関しては全員必修のため試験も全員受験しており，クラス分けが機能していると実感しているものの，後期では２科目とも選択科目であり，振り分けに苦心する．大きな点でいえば，前期時点で履修登録をしていたが後期に登録を抹消したり，その逆があったりするために，クラスの人数配分が変動してしまう．また数学がそれほど得意ではないと自己分析する学生が，前期の成績結果で発展クラスに分類されて成績を落としてしまうケースも見受けられる．

今回の報告ではこのクラス分けの作業に，機械学習を取り入れてみるという趣向である．近年の AI や機械学習に関する需要の高まりもあって，著者らがその技術に触れることにも意義があると考えた．しかし適切な習熟度別のクラス配置を機械学習によって実施できれば望ましいが，そのための案にはいたらず，前段階としての履修登録者を予測するまでに留まった．それでもクラス分けの作業に対しては有益であり，予測した履修者の人数に基づいたクラス配置が可能となり，作業の効率化が期待できる．

2．機械学習について

TensorFlow とは Google が開発した機械学習（Machine Learning）や深層学習（deep learning）のためのソフトウェアで，Web 上では画像認識やクラスタリングなどへの利用例を見つけることができる．データ分析での利用がさかんなプログラミング言語である Python で TensorFlow を利用するのが主流であるようだが，統計処理ソフトの R でも RStudio を用いて TensorFlow パッケージを読み込むことで利用が可能である．Keras は TensorFlow などをバックエンドにして利用する手軽なフレームワークで，たとえばプレプリント・サーバ arXiv.org にアップロードされた科学技術論文で言及されているフレームワークの中では，二番目に多く使用されているとされる(1)．幸いなことに，こちらも

[*1] 教育創造工学科　[*2] 九州大学多重ゼータ研究センター　[*3] 久留米工業大学　[*4] 九州大学大学院数理学研究院　[*5] 九州大学大学院数理学府
令和元年11月30日受理

TensorFlow と同様に RStudio から利用できるパッケージが開発されており，本稿では RStudio から Keras を利用する．

R と Python はいずれも数理解析に向いた言語であり，どちらを使ってもほぼ同等の分析が可能であるが，多少の向き不向きや使い勝手の違いはある[2]．端的には R は統計解析に向き，Python は機械学習に向くといえるが，どちらも他方を呼び出すことができるパッケージが提供されており，慣れた環境から用途に応じてもう一方を利用すればよい．データの処理に携わった著者らは Python に触れたことはないが R での処理の経験があったため，R interface to Keras によって Python を利用した処理を行った．なお実行時の環境は R：3.5.2（2018-12-20），RStudio：Version 1.2.5001, Python：3.6.0，TensorFlow：2.0.0，Keras：2.2.5である．

3．利用するデータの概要

Keras において，学習に利用されるデータ（学習用データ），および予測に利用されるデータ（予測用データ）について，以下で言及する．学習用データは，2018年度前期に開講された数学基礎の成績と，同年後期に開講された微分積分学および線形代数学の履修状況のデータである．340件のデータがあるが，欠損値のあるデータなどを除いて334件を利用した．列については８列あり，c１：前期科目の習熟度別のクラス，c２：前期科目の定期試験における基礎問題の点数，c３：同応用問題の点数，c４：前期科目の平常点，c５，c６：後期科目の前期時点での履修状況（線形代数学および微分積分学），c７，c８：後期科目の実際の履修状況（線形代数学および微分積分学）である．学習には c７，c８を除いた６列を用いて，出力として c７と c８をそれぞれ用いる．すなわち前期科目の状況から後期科目の履修状況を判断するという設定だが，今回は試験的な取り組みということもあり，c７の線形代数学についてのみ学習を行った．予測用データは2019年度前期の，学習用データと同様の形式である．このデータを Keras によって学習されたモデルに与えることで，2019年度の後期履修者を予測できる．本稿での取り組みを開始した時点では履修状況は確定していなかったため６列であったが，のちに確定した情報を追加して８列ある．行数は345件あったが不備のある１件を除外して344件を利用した．

学習用データと予測用データに関しては，非常によく似た傾向にある．前期試験の結果の部分は c２，c３，c４の３列が該当するが，３列それぞれについて，点数に関して等分したうえで集計表を作成し，年度でペアにして fisher 検定を行うと，いずれの列に関しても p 値は１であった．このデータをとるための試験問題がまったく同一であることを考えると，年度による集団の差はないと考えられる．履修状況に関しても，前期時点での履修登録者が後期で登録を抹消した人数と，逆に前期未登録者が後期で登録した人数の，どちらに関しても２つの年度でほぼ同じであった．このようなデータの様子から鑑みて，上述のような行動を機械学習で予測することを期待し，後述のようにある程度の精度を持つ結果を得た．

4．Keras による処理および結果

R で TensorFlow や Keras を利用するには RStudio を用いる必要がある．Python も必要になるため，あらかじめインストールしておく．なお草稿の時点での，macOS Catalina に関する以下の留意事項がある．macOS Catalina がインストールされた時点で，すでに Python2.7がインストールされた状態になっている．その状態からたとえば Python3.6 を導入しても，Rstudio から TensorFlow パッケージの利用を試みると，Python2.7が動作して（つまり Python3.6の方は動作せず），TensorFlow の関数が動かない現象がある．ただし Keras パッケージであれば実行できる．Keras の導入と動作については RStudio の公式 Web ページの他にも，GitHub や Qiita などにわかりやすい情報が多い．

学習用データの読み込みに関しては，c１のクラスの情報は文字列であったため整数に変換した．点数を表す列である c２，c３，c４の３列を除く列，つまり c１，c５，c６，c７の４列については１列ですむのだが，one-hot エンコーディングを施すために，to_categorical 関数によってそれぞれ展開した．そして c２，c３，c４の３列については scale 関数で正規化を行った[3]．このように加工した都合12列について，Keras による学習を行う．留意点として，データが tibble 型の場合，モデルを学習する fit 関数でも予測する predict 関数でも，動作しないようであるため，matrix 型にする必要があると思われる．

学習は全結合層で行った．最初に keras_model_sequential 関数を呼び出して全結合層であることを示して，layer_dense 関数でユニット数と活性化関数を指定する．本来はデータに応じた結合やユニット数，活性化関数を選ぶべき[3]だが，知識に乏しいため諸々のパラメータを取り替えてよりよい学習ができないかと試みた．階層数とユニット数に関しては，これらが少ないと予測に偏りが生じてしまうが，ある程度の大きさを指定すれば安定した予測を返すことがわ

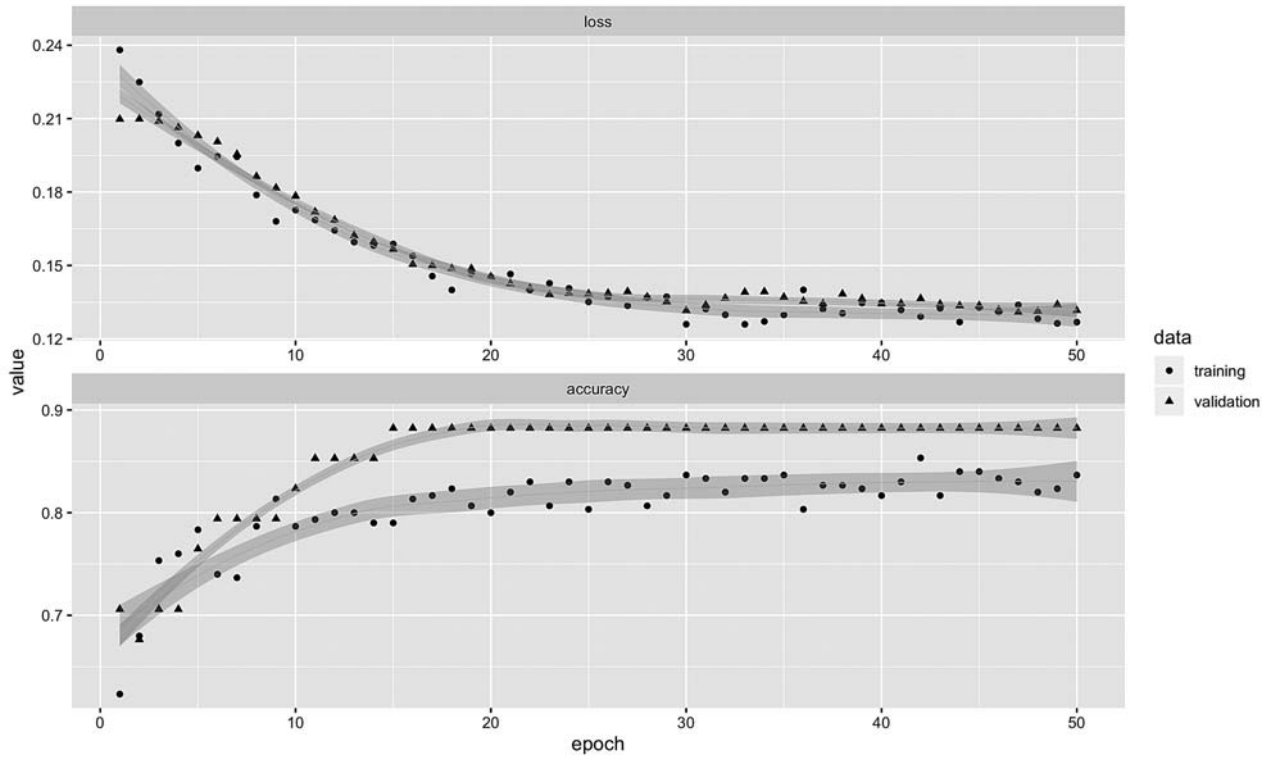

Fig. 1: Progress of Machine Learning

かった. 活性化関数については, それを変更しても予測に著しい変化は見られなかった. そのため階層の数は 3 でユニット数を順に64，64，2 とし，活性化関数は relu を先の 2 層で，また softmax を最後の層で用いたモデルを採用した. そしてモデルを compile 関数に渡すが，こちらも上同様にパラメータを調整しながら，損失関数は mean_squared_error を，最適化器は optimizer_rmsprop を，評価基準は accuracy を，それぞれ指定することとした. 作成したモデルと学習用データを fit 関数に与え，モデルの訓練回数である epochs を50，毎回の学習に利用するデータ数である batch_size を80，学習で検証用データとして用いられるデータの割合を示す validation_split を0. 1として，学習を実行した. なお検証用データについては，紛らわしいが学習用データのことではない. 学習用データは訓練用データと検証用データに分けられて学習を進めるが，その割合を validation_split で指定する. 学習中のそれぞれの訓練においてはオプションである callbacks を指定すれば，たとえば callback_early_stopping 関数を利用し val_accuracy を基準にして特定の状況下で学習を中断することができる.

fit 関数で学習している間は RStudio の Plot ウィンドウに，各 epoch ごとの学習の進行状況が表示される. fit 関数の実行結果を history などの名前の変数に代入しておけば，fit 関数による学習後にその変数を plot 関数に渡すことで，図 1 のような近似曲線も含めた出力が得られ，さらに ggplot 2 パッケージの ggsave 関数を利用すれば描画ファイルの保存が可能である. デフォルトでは赤と青のカラフルな描画がなされるが，history と名前をつけた keras_training_history オブジェクトを data. frame に変換した後に手動で ggplot を実行することによって，本稿のような印刷向きの出力が可能である[3]. 図中の loss は入出力の損失の程度を，accuracy は入出力の正確度を表し，training は訓練用のデータにおける精度，validation は検証用のデータにおける精度を表す[3]. 精度の高い学習では回数を増すごとに，loss の値が減少し，accuracy の値が増加する. たとえば（3）では MNIST データセットを用いて分類を行っているが，検証用のデータにおける正確度は98. 8%，損失は0. 03という値が，わずか 5 回の学習で達成されている. そこまでの精度は達成できなかったとはいえ，我々の設定下の学習でも正答率が85%前後，損失は12%前後の精度を恒常的に達成できており，図 1 はそのうちの一例である. したがって予測も一定の水準を保つことが期待され，実際にある程度の結果を得られた.

fit 関数での学習を終えたモデルによる予測には，predict 関数にモデルと予測用データを与えれることで実行される. 出力結果は行和が 1 であるような 2 列からなるが，前者が未履修，後者が履修を表す列であり，それらを確率的にとると解釈して，予測結果は 2 列のうちの大きい値の方を 1 に，小さい値の方を 0 に丸めた. この結果と実際の履修状況を照らし合わせて，予測の正答率を求めることができる. すなわち行のパターンには，4 月時点で未登録（c 5 列が 0 ）

で10月の確定時点でも未登録（c７列が０）というパターンの他，未登録から登録（c５列が０でc７列が１），登録から未登録（c５列が１でc７列が０），登録から登録（c５列，c７列とも１）という４つのパターンがあり，そのうち“00”と“11”が正答で“01”と“10”が誤答なので，“00”と“11”を合わせた人数を全体数で割ったものを正答率と扱った．上でパラメータを変えて学習を行ったり，パラメータの決定後も複数回の学習を行ったが，どの学習においても概ね85％前後の正答率を達成できた．たとえば上記の設定で100回の学習を行った結果を記述すると，正答率に関しては84％から87％にすべて収まっており，中央値は86％であった．誤答を表すパターンのうち“01”に該当するのは28名～39名で中央値は34名，“10”に該当するのは13行～15名で中央値は14名であり，100回をとおしてほぼ均一であるといえる．誤答だと判断される行は100回の学習で共通しており，“01”のパターンで下限の28名のケースに含まれる行はそのまま上限の39名のケースに含まれる状況であるし，“10”のケースも同様である．学習は初期値を変えて行われるにも関わらず予測結果は概ね一貫しており，それなりに満足のいく結果だといえる．

最後に今回のKerasによる学習における所見を挙げる．階層数とユニット数の変更は予測への影響が大きかったが，関数の変更は試した中ではそれほど変化を生じなかったと感じた．階層数は３としたが，４層で学習しても予測の改善は見られなかった．１層目のユニット数が16の場合では予測のばらつきが見られたが，64に設定してからは安定した．処理の当初は名義尺度の列をそのまま整数として，つまりone-hotエンコーディングをせずに１列で入力データとして扱っていたが，その状況下での結果は散々なものであった．たとえば予測値においてすべてが０（未履修），またはすべてが１（履修）と予測されるケースが全学習中の９割近く発生してしまい，残りの１割でようやく55％程度の正答率を達成できる程度であった．one-hotエンコーディングをしなければ連続値として扱われてしまうため，データの解釈がまずいというのはわかるが，予測にこれほど影響があるというのは驚きであった．なおone-hotエンコーディングの考え方では名義尺度と順序尺度を同じように扱うため，習熟度別に分けたクラスという順序尺度を加味したデータの読み込みが実現できれば，さらなる精度向上もあるかもしれないが，そのような手段があるのかは不明であった．精度の向上に関しては変数の追加による効果が期待され，たとえば予測用データには出席回数も入っていたのだが，学習用データには一部の行で入っておらず，学習の列から除外している．出席回数は履修状況に影響するように思われるため，これを加えると多少は精度が改善された可能性がある．そして本稿ではCNNやRNNといった手法は適用できておらず，それらによる予測の改善も見込まれる．処理過程では，学習用データの並び順による学習成果に興味をもった．Kerasで利用される学習用データにおいて，fit関数のオプションであるshuffleを変更すれば訓練用データは入れ替えることができ，またvalidation_splitは検証用データの割合を表すが，それらのデータ自体は固定される(1)．そうすべき理由があってKerasで設定されているのかもしれないが，たとえば学習用データの最終行付近に学習が難しい標本が固まっている場合はval_accuracyが向上しにくい，という可能性を考えた．そこで毎回の学習の最初に，学習用データの行をランダムに取り替える操作を追加する操作を200回行い正答率をかぞえてみたが，特に有意な差は得られなかった．

文　献

(1)　https://keras.io/ja/why-use-keras/, “Keras Documentation”
(2)　有賀友紀，大橋俊介，“R と Python で学ぶ実践的データサイエンス＆機械学習”，技術評論社.
(3)　長尾高弘，“R と Keras によるディープラーニング”，オライリー・ジャパン
(4)　梅田弘之，“エンジニアなら知っておきたい AI のキホン”，インプレス
(5)　https://tensorflow.rstudio.com,“TensorFlow for R from RStudio”