競馬AI班活動報告書

立命館コンピュータクラブ
2020年度プロジェクト活動

2021年2月19日

堀田 隆成[1]坪倉 奏太[2]玄元 奏[3]
木村 悠生[4]阿部 竜也[5]深田 紘希[6]
田邊 雄士[7]星名 藍乃介[8]斎藤 竜也[9]
中尾 龍矢[10]水野 悟朗[11]佐田 淳史[12]西村 雅貴[13]

[1]情報理工学部 情報理工学科 セキュリティネットワークコース 二回生
[2]情報理工学部 情報理工学科 実世界情報コース 三回生
[3]情報理工学部 情報理工学科 実世界情報コース 四回生
[4]情報理工学部 情報理工学科 セキュリティネットワークコース 三回生
[5]情報理工学部 情報理工学科 実世界情報コース 二回生
[6]情報理工学部 情報理工学科 実世界情報コース 二回生
[7]情報理工学部 情報理工学科 システムアーキテクトコース 二回生
[8]情報理工学部 情報理工学科 セキュリティネットワークコース 二回生
[9]情報理工学部 情報理工学科 実世界情報コース 二回生
[10]情報理工学部 情報理工学科 システムアーキテクトコース 二回生
[11]情報理工学部 情報理工学科 システムアーキテクトコース 二回生
[12]理工学部 電子情報工学科 一回生
[13]情報理工学部 情報理工学科 知能情報コース 一回生

目 次

1　活動概要

文責: 堀田 隆成

競馬 AI 班は競馬 AI の作成を通じ AI 作成に必要な知識及びテクニックを取得することを目的としたプロジェクト活動である．機械学習の基礎を学び，その後競馬 AI を開発する上で必要なデータの収集やモデルの構築などを勉強会形式で実践する流れで進行した．前期には簡単な競馬 AI の作成，後期にその競馬 AI の精度向上を目指し活動を行なった．

2　機械学習

文責: 佐田淳史

2.1　パーセプトロンとは

複数の信号を入力値として受け取り，一つの信号を出力する (図 1).

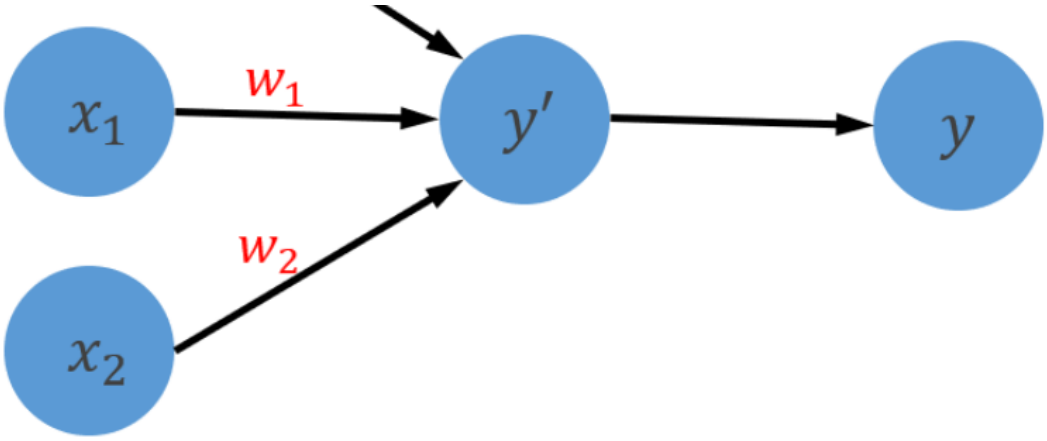

図 1

$X = x_1, x_2, .., x_n$ は入力信号，y は出力信号，$W = w_1, w_2, .., w_n$ は重みである．入力信号はパーセプトロンに送られる際にそれぞれに固有な重みが乗算される．パーセプトロンでは送られてきた入力信号に重みを掛け合わせた総和が計算されてその総和が閾値 θ を超えたときのみに 1 を返す．数式で表すと

$$y = 0(b + \Sigma_n(x_n w_n) \leq 0) \tag{1}$$
$$1(b + \Sigma_n(x_n w_n) > 0) \tag{2}$$

パーセプトロンは複数ある入力信号のそれぞれに固有の重みを持つ．その重みは各信号の重要性をコントロールする要素として働く．重みが大きければ大きいほどその重みに対する信号の重要性が高くなる．この重みを最適化するための手法が機械学習である．

2.2 簡単なパーセプトロン：論理回路を考える

Python で AND を実装した例を下記に示す.

```
def AND(x1, x2):
    w1  = 0.5
    w2  = 0.5
    b   = -0.7
    if b + x1 * w1 + x2 * w2 <= 0:
        y = 0
    else:
        y = 1
    return y

AND(0, 0)  # => 0
AND(1, 0)  # => 0
AND(0, 1)  # => 0
AND(1, 1)  # => 1
```

ソースコード 1: AND 回路

実行結果は上から　0001　となる．同様に NAND と OR を実装した．AND，NAND，OR の重み及びバイアスは下記の通りである．

- AND：$(w_1, w_2, b) = (0.5, 0.5, -0.7)$
- NAND：$(w_1, w_2, b) = (-0.5, -0.5, 0.7)$
- OR：$(w_1, w_2, b) = (0.5, 0.5, -0.2)$

2.3 XOR パーセプトロンの実装

AND や OR のような線形分離可能な単純な演算であれば単一パーセプトロンで実現可能である．一方で，XOR は線形分離不可能なため工夫が必要である．そこで，単一パーセプトロンを複数個つなげた多層パーセプトロンを考える．XOR は NAND，OR，AND の組み合わせで構築することが可能なため，これを多層パーセプトロンで再現する．

```
def XOR(x1, x2):
    return AND(NAND(x1, x2)), OR(x1, x2))

XOR(0, 0)  # => 0
XOR(0, 1)  # => 1
XOR(1, 0)  # => 1
XOR(1, 0)  # => 0
```

ソースコード 2: XOR 回路

3　スクレイピング

3.1　Web スクレイピング

文責: 田邊 雄士

第 3 週は必要となる情報を集めるためスクレイピングとは何か学び, 簡単な例を用いて実践した.

3.1.1　スクレイピングとは

Web スクレイピングとは,Web 上のデータを自動的に収集し利用しやすいように加工することである. 具体的には, 以下の 3 つの要素からなる.

1. 該当 Web ページの HTML を取得, 解析
2. 取得した HTML から取得必要なデータだけを取得
3. 取得したデータを扱いやすいように加工

3.1.2　実践練習

会員の理解を深めるため, 天気情報を取得, 加工するといった実践的練習を行った. Python のライブラリの requests,BeautifulSopu4 を用いて `https://weather.yahoo.co.jp/weather` から天気情報の取得を行った. まず,requests を用いて該当 Web ページの HTML を取得した.

```
target_url = 'https://weather.yahoo.co.jp/weather/jp/25/6010.html'

# target_url の HTML を取得
r = requests.get(target_url)
```

ソースコード 3: HTML 取得

次に,BeautifulSoup4 を用いて HTML を解析, 取得したい情報を find や select を用いて取得した.

```
soup = BeautifulSoup(r.content, 'html.parser')
print(soup)

links = soup.select("#main > div.forecastCity > table > tr > td > div > p")
```

ソースコード 4

最後に, 取得したデータを空白や不要な文字などをトリミングして見やすくした.

```
for link in links:
    print(link.text)
#取得したタグのテキスト部分だけを print
#print(links) にしてしまうと<p>~</p>とタグまでプリントされる
```

ソースコード 5

3.2　競馬データ収集プログラム

文責: 星名 藍乃助

3.2.1　概要

競馬データ収集プログラム (以下，本プログラム) は，本班で作成する競馬予想 AI プログラムの教師データとなる競馬データを収集するプログラムである．

3.2.2　データ出典

競馬データは，株式会社ネットドリーマーズが運営する競馬の総合ポータルサイト，netkeiba.com(以下，当該 Web サイト) が提供する競馬データベース (`https://db.netkeiba.com/`) から Web スクレイピングして取得する．

3.2.3　注意した点

Web スクレイピングではサーバから HTML ファイルを取得する．そのため，リクエスト間隔が短ければサーバに過剰に負荷をかけてしまい，サービスを妨害してしまう可能性がある．本プログラムでは，サーバへの負荷を考慮し，リクエストは最低でも 2 秒，サーバでデータベースの操作が必要なリクエストについては 3 秒以上の間隔を空けた．
また，robots.txt や利用規約から，当該 Web サイトがスクレイピングを禁止しているかを事前に確認した．robots.txt は慣習的にサイトの URL の直下に置かれるが，当該 Web サイトではそこには存在しなかった．利用規約には，スクレイピングの禁止やそれに類似する注意事項は記述されていなかった．

3.2.4　取得データ仕様

本班の第 5 回の活動では，機械学習に使用する競馬データの項目を班員で検討した．取得対象とした競馬データは，次の 4 種類に分けられる．

- 競走馬データ
- 騎手マスタ

- レースデータ
- 調教師マスタ

また，それぞれの競馬データにおける項目やデータ構造は次の通りである．ただし，形式は json とした．

```
[
    {
        "netkeiba_id" : "10 桁の競走馬 id",
        "name" : "競走馬名",
        "birthday" : "YYYY 年 MM 月 DD 日",
        "sex" : "性別 (0 : 牡, 0.5 : 牝, 1 : セ)",
        "ped_id" : "血統 ID(血統マスタの id)",
        "trainer_id" : "10 桁の調教師 ID",
        "wins" : "勝数 (0〜の整数)",
        "win_rate" : "直近 10 件の 1 位になった率 (0〜1 の小数)",
        "Rank_average" : "直近 10 件の順位平均",
        "money" : "獲得賞金 (0〜の整数)",
        "course_suitability" : "コース適正 (0〜1 の小数；0 : 芝 <-> 1 : ダート)",
        "distance_suitability" : "距離適正 (0〜1 の小数；0 : 短 <-> 1 : 長),
        "limb" : 脚質特性 (0〜1 の小数；0 : 逃げ <-> 1 : 追込),
        "growth" : 成長特性 (0〜1 の小数；0 : 早熟 <-> 1 : 晩成),
        "muddy_track_suitability" : 重馬場適正 (0〜1 の小数；0 : 得意 <-> 1 : 苦手),
        "summer_suitability" : 夏適正 (夏の勝率の変動；0 : 下, 1 : 上),
        "winter_suitability" : 冬適正 (冬の勝率の変動；0 : 下, 1 : 上),
        "speed" : 直近 10 件の走る速度 [m/s](タイム合計 / 距離合計),
        "breeding_center" : "産地名"
    },
    {...},
    {...},
    ...
]
```

ソースコード 6: 競走馬マスタのデータ仕様

```
[
    {
        "netkeiba_id":" 5 桁の騎手 id(文字列型)",
        "name":"騎手名",
        "win_rate":"直近 20 件の勝率 (小数)",
        "rank_average":"直近 20 件の平均順位 (小数)"
    },
    {...},
    {...},
```

```
    ...
]
```

ソースコード 7: 騎手マスタのデータ仕様

```
[
    {
    "netkeiba_id":"12 桁の調教師 id(文字列型)",
    "レース名":"レース名",
    "馬場状態": "芝/ダート",
    "周り方":"右/左",
    "距離": 走距離 (整数, 単位:m),
    "天気": "晴/曇/雨",
    "開催日":"yyyy 年 mm 月 dd 日",
    "賞金":[賞金 (単位:円),~,~],
    "競走馬": [
      {
        "競走馬 ID":"10 桁の競走馬 id(文字列型)",
        "順位": 順位 (整数),
        "馬番": 馬番 (整数),
        "性別": "牡/牝/セ",
        "年齢": 年齢 (整数),
        "斤量": 斤量 (小数)
        "単勝オッズ": オッズ (小数),
        "体重": 体重 (整数),
        "体重増減": 体重増減 (整数)
      },
      {
          ...(その他の競走馬データ)...
      }
    ]
  },
  {
      ...(その他のレースデータ)...
  }
]
```

ソースコード 8: レースのデータ仕様

```
[
    {
        "netkeiba_id":"5 桁の調教師 id(文字列型)",
        "name":"調教師名",
```

```
        "win_rate":直近 20 件の勝率 (小数),
        "rank_average":直近 20 件の平均順位 (小数)
    },
    {...},
    {...},
    ...
]
```

ソースコード 9: 調教師マスタのデータ仕様

3.2.5 使用技術

本班の第 3，4 回の活動と同様に，Python の requests モジュールと BeatifulSoup モジュールを用いて本プログラムを実装した．

3.2.6 作業手順

競走馬データを取得する手順は次の通りである．

1. 当該 Web サイトにおける競走馬 ID の一覧を取得できるページの URL を検討する
2. 競走馬 ID を取得してファイルに保存するプログラムを実装する
3. 上記プログラムを実行する
4. ページを取得してそれぞれのデータ項目を抽出し，仕様を満たす形式でファイルに保存するプログラムを実装する
5. 上記プログラムを実行する

3.2.7 URL とクエリ文字列の検討

作業手順 1 の，競走馬 ID 一覧を取得できるページの URL を検討した手順について述べる．
基本的には静的ページから取得するが，ID 一覧など，一部は動的ページから取得しなければならない場合があった．その時はクエリ文字列を付加した URL を作成して対処した．例えば，競走馬 ID 取得プログラムで使用した URL は「`https://db.netkeiba.com/?pid=horse_list&grade[]=1&grade[]=2&grade[]=3&list=100&page=`」である．これは，URL の末尾の「?」から始まるクエリ文字列によって，取得したい検索結果を指定している．
当該 Web サイトに登録されている競走馬は 53 万件以上あるため，そのすべてを取得することは困難と考えた．そこで，簡単のため，未出走を除く競走馬のうち獲得賞金の多い順に 15 万件を取得対象とした．当該 Web サイトでは，このような条件を付けた競走馬の ID は，競走馬詳細検索ページで条件を指定することで検索することができる．
また，競走馬詳細検索ページの HTML 中の未出走とラベルされた input タグを参照すると，次のようになっていた．

「<input type=”checkbox” name=”grade[]” value=”1” class=”check” id=”check_1”>」そこで，パラメータが grade[]，値が 1 のクエリ文字列を URL に付加して，`https://db.netkeiba.com/?pid=horse_list&grade[]=1` を URL にリクエストを送ると，未勝利の競走馬リストが表示された．
このように，HTML 中の属性とその値を参照しながら，対象の検索結果をリクエストできる URL を決定した．また，その他の競馬データにおいても同様の手順で URL を決定した．

3.2.8 ID 取得プログラム

作業手順 2 で紹介した競走馬 ID 取得プログラムの一部実装について述べる．
競走馬 ID 取得プログラムは，あらかじめ決められた件数分の競走馬 ID を取得した後，それらを csv 形式で保存する．ページはソースコード 10 のように，requests.get() メソッドを用いて取得した．
競走馬 ID は，ソースコード 11 のように，ページ内の馬名が記載されている a タグの href 属性から抽出することができた

```python
def download_html(url):
    ## -----*----- 指定 URL の HTML 文字列を取得 -----*----- ##
    for i in range(3):
        sleep(3)  # 優しさ
        try:
            res = requests.get(url)
            res.raise_for_status()
            return res.content

        except:
            print(str(i) + ' this page is not found')
```

ソースコード 10: ページ取得

extract_id 関数

```python
def extract_id(soup):
    ## -----*----- HTML から 1 ページ分の競走馬 ID を抽出 -----*----- ##
    table_tag = soup.find("table")

    selector = 'tr > td:nth-child(2) > a'
    a_tags = table_tag.select(selector)

    return [i.get('href')[7:-1] for i in a_tags]  # href="/horse/123456789A/"
```

ソースコード 11: extract_id 関数

3.2.9 データ取得プログラム

作業手順 4 で紹介した競走馬マスタ取得プログラムの一部実装について述べる．

競走馬マスタ取得プログラムは，csv ファイルから競走馬 ID を読み込み，これを基に生成したそれぞれの競走馬ページの URL からページを取得，それぞれの項目を抽出，そして仕様を満たす形式でファイルに保存するプログラムである．
項目は，ソースコード 12 のように，CSS セレクタでタグを指定し，select_one() メソッドで該当箇所を抽出，さらに，これを仕様を満たす形式に加工して取得した．
get_sex 関数

```
    def get_sex(self, soup):
        ## -----*----- 性別を取得 -----*----- ##
        selector = '#db_main_box > div.db_head.fc > div.db_head_name.fc > div.horse_title
        text = soup.select_one(selector).text  # 例: "抹消　牡　鹿毛"

        if '牡' in text:
            return 0
        if '牝' in text:
            return 0.5
        if 'セ' in text:
            return 1
        return None  # 欠損値
```

ソースコード 12: get_sex 関数

3.2.10　成果

競馬データ収集の活動では，予定していた血統マスタが集められなかったり，機械学習担当の班員へデータを渡すのが遅くなったりと反省点はいくつかあった．しかし，機械学習に必要な教師データを準備できたことは成果であると考える．
また，スクレイピングのソースコードは Web サイトの設計に依存し，長く複雑になりやすいが，それでも，本プログラムでは可能な限り可読性を保てるよう意識した．その過程で，徐々にプログラムの設計力が開発当初と比べて向上したと感じる．

4　前処理

4.1　前処理とは

文責: 西村雅貴

前処理とは生のデータを学習前に加工することで学習しやすいデータに整理することであり，様々な種類がある．今回はその中でも「カテゴリデータの処理」「欠損値処理」「外れ値処理」「正規化・標準化」に絞って扱った．

4.2　カテゴリデータの処理

学習するためのカテゴリデータ (項目やラベルを区別するための文字列や数値の集合) の内, 文字列などの機械が学習しにくいデータを数値にする処理である. 例として,SML からなるサイズのカテゴリ変数をそれぞれ 123 と置き換えることで説明変数として扱いやすくなる.

4.3　欠損値処理

記入漏れやデータエラーなどの理由により値がない欠損値を無視もしくは平均値や中央値などで補完する処理である. 本プロジェクトで扱うデータに欠損値はなく実装はしなかった.

4.4　外れ値処理

外れ値が異常値か意味のある数値かを判断し, どのような基準で除外するのかを決める処理である. 今回は欠損地処理と同様に外れ値はないものとし実装はしなかった.

4.5　正規化・標準化

学習時の各パラメータのスケールによる影響を均一にするために値の範囲を変える処理をスケーリングといい, 特徴量のスケーリングとして代表的に正規化と標準化がある. 正規化は一般にデータの最大値を 1, 最小値を 0 となるように変換する処理である. 画像処理などに用いられ, 変換式は以下の通りである.

$$X_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}$$

一方, 標準化はデータの平均を 0, 分散を 1 に変換する処理である. 正規化は外れ値による影響が大きいため標準化がよく使われる. 変換式は以下の通りである.

$$X_i = \frac{x_i - \mu}{\sigma} (\mu : \text{平均 } \sigma : \text{標準偏差})$$

4.6　次元削除

文責: 深田紘希

特徴量が高次元であると，その分析をする際の計算量が指数関数的に増えてしまうことで学習がうまくいかないことがある．それに対処するためにデータの次元数を削減する必要があり，その次元削除アルゴリズムとして主成分分析を用いた．主成分分析は多くの変数をできるだけ情報の損失なしに互いに独立な小数の主成分で表す分析方法である．次元削減はデータの損失がどうしても発生してしまうが，その損失をできるだけ少なくするための指標が寄与率である．寄与率は主成分の分散と元情報の分散の商で表され，これの小さい主成分から削除することで次元削減を行うことができる．主成分分析は単位の影響を受けてしまうため，前項のデータを標準化させることが必須となる．

5　軽量版レシピ

文責: 阿部竜也

5.1　概要

本プロジェクトでは軽量版の学習レシピを作成した．これは競馬の一着予想をクラスタリング問題として定式化したものであり，少量のデータと単純なニューラルネットワークで学習が可能である．仕組みはシンプルなため，競馬 AI 開発を体験する目的としてこれを用いる．
軽量版レシピのニューラルネットワークアーキテクチャは下記の通りである．

- 64 nodes の feed-forward 層
- 128 nodes の feed-forward 層
- 256 nodes の feed-forward 層
- 21 nodes の feed-forward 層

5.2　特徴量

特徴量には，レース特徴と競走馬特徴を用いる．下記 3 項目をレース特徴とする．

- 馬場状態：芝 or ダート
- 距離：[m]
- 天気：晴 or 曇 or 雨

競走馬特徴において，21 頭の競走馬がそれぞれ下記 7 次元のベクトルを持つ．

- 枠
- 性別：牡 or 牝 or セ
- 年齢
- 単勝オッズ
- 体重
- 体重増減
- 斤量

上記のレース特徴（3 次元）と競走馬特徴（21*7）次元を合わせた 150 次元のベクトルを予想するレースの特徴量とし，ニューラルネットワークへ入力する．

5.3　インストールと学習

まずは軽量版レシピをダウンロードする．

```
git clone https://github.com/averak/rcc-keiba-2020-demo
cd rcc-keiba-2020-demo
```

続いてサブセット（非公開）をダウンロードし，展開したら json ファイルを data 配下にコピーする．

```
cp -ip "サブセット/*.json" data
```

サブセットが用意できたら，学習を開始できる．train.py を実行すると学習が始まり，最終的な精度は 25%前後となった．21 クラスのクラスタリングであるため，ランダムよりは良い結果となった．

```
python train.py
```

6　今後の展望

文責: 堀田隆成

作成した競馬 AI の精度を高めるために類似した内容の論文を読み機械学習への理解を深める．現状の競馬 AI は 1 着の予想をすることしかできないため，3 着までの予想をするなどし，効率的な馬券の予想を行える AI へ改良する．また競馬 AI の予想をツイートする Bot を作成したが，作成し数日で動作しなくなったためその原因を探る．

参考文献

[1] 競馬 AI の予想をツイートする Bot [https://twitter.com/rits_haha]

[2] Python によるスクレイピング&機械学習開発テクニック, クジラ飛行机