経営学部における実践的なAIの実習について

湊　　信吾*

Practical training of AI on Faculty of Business Administration

Shingo MINATO*

[1]*Department of Information Management, Faculty of Business Administration, Ishinomaki Senshu University,Miyagi 986-8580, Japan*

Abstract

This article discusses practical training of AI on faculty of business adminstration through the development process of web application with trained model.

1. 緒言

この小論では経営学部においてAIの実践的な実習を行う方法について考察する。AIについてはディープラーニングを対象とし、学習済みモデルを使用してWebアプリの開発を行うためのフローについて考えてみる。

2. ディープラーニングの考え方

次の図１のようにディープラーニングはAIの一分野である。機械学習の一分野でもある。

ディープラーニングでは大量のデータを用意し､それぞれのデータに判定の元になるデータ（ラベル）も用意する。このラベルによってデータをどのように分類するのかを示すことができる。これをニューラルネットワークにより学習という作業を行い、分類を行えるような値になるまでパラメータを調整していく。この時、エポック数と呼ばれる学習回数を設定する。ディープラーニングでは図２[(1)]のように入力層と出力層の間に隠れ層というものを用意し、データは入力層のノードと呼ばれる部分（図では○）から入力された後で、決められた条件により次の層のノードに値を渡し

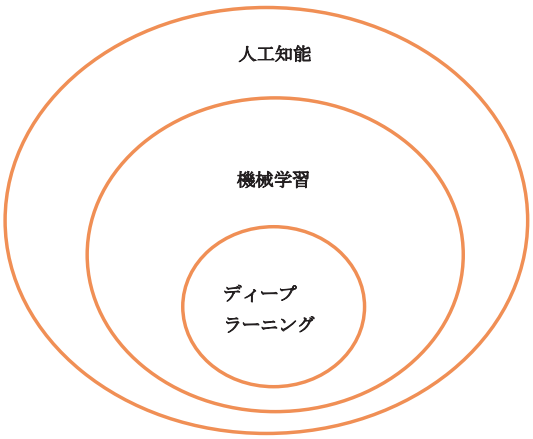

図１．人工知能の分類

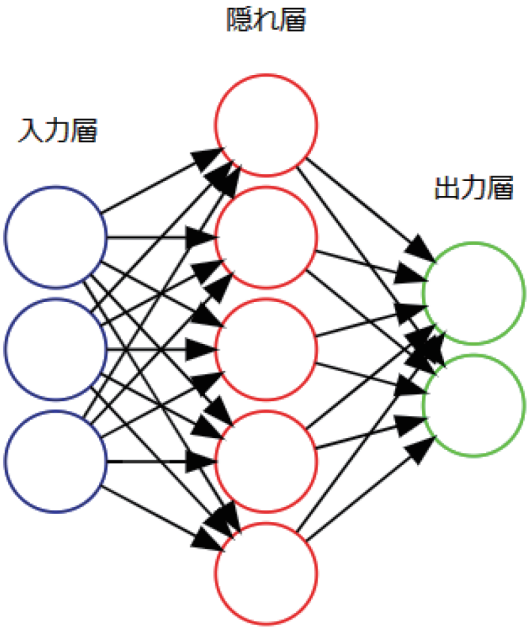

図２．ニューラルネットワーク

*石巻専修大学経営学部情報マネジメント学科

ていく。

最終的に出力値が得られた時に、数値化したラベルと比較し、ラベルの値との差が小さくなるようにパラメータを調整していく。この時ニューラルネットワークを逆にたどりながらパラメータを調整していく。このようにして最終的に各層間のパラメータが決定する。ディープラーニングではこのような各層間のパラメータを特徴量というもので捉える。特徴量をまとめたものはモデルとして扱われる。このモデルを使用して、テスト用のデータを使ってそのデータを正確に分類できるかどうかを調べていく手法である。

実習ではこのようなディープラーニングの概念についてはっきり理解してもらうことが大切である。また統計的機械学習という分野もあるがディープラーニングの手法が比較的単純なことから各種のデータに応用が効くというところも示す必要があるだろう。

3. ディープラーニングの実習で考慮すべきこと

モデルを作成するには一般に大量のデータを必要とする。そして、特徴量を求めモデルをファイルとして出力するためには、大量の演算を処理する能力が実習時に使用するパソコン（PC）に求められる。そのため通常の PC のマルチコアの CPU では限界があるため演算用に GPU を備えたものが求められる。

情報マネジメント学科では学生に PC を所有させ授業で活用してもらっているが、Excel、Word、PowerPoint などビジネス向けのソフトウェアが使えることを前提としている。ゼミで試したところ、学生が所有している平均的な PC では Web アプリの開発に必要な Django などのフレームワーク、仮想環境を作りだすための miniconda およびプログラミング言語 python などをインストール、実行しても特に支障が出るということは少ない。しかしディープラーニングにおいて大量のデータをもとに学習させる作業は GPU なしの PC を持っている場合、学習の実行だけで数時間を要し、授業内で処理が終わらない。

そこで GPU のパワーを利用することができる環境として Google Colaboratory がある。Google Colaboratory は Google のクラウド上のサービスであり、gmail のアカウントを持っていれば Google Colaboratory の開発環境を試すことができる。Google Colaboratory では jupyter notebook の形式（ノートブック）で python のプログラムをセルと呼ばれる入力欄に入力後実行し、出力結果も同じノートブック上に表示することができる。またディープラーニング向けの python のライブラリとして PyTorch、TensorFlow が標準として用意されていることから、開発環境を最初に構築する必要もなく即座にプログラムを試すことができるようになっている。重要なのはランタイムの環境を設定できることであり[(2)]、用意したノートブックごとに T4 GPU を課金なしで利用することができる。後で実際の例を示すが学習時の計算は何回か行われる。毎回の処理時間を数十秒程度で終わらせることができる。このようにディープラーニングにおける学習という作業を実際の実習で行うためには Google Colaboratory を利用することは必須であると思われる。

しかし、モデルが作られても Google Colaboratory のルールでは作業用のメモリ上に残されているものがセッション終了時に削除されてしまう。いくら GPU を使って学習の作業が短時間に行えるとはいえ、毎回学習の作業を行うのは SDG の観点からも非効率である。そのためモデルを保存、読み込みという作業が必要になる。

Google Colaboratory で作られたモデルをファイルとして保存しても、一次保存用の領域に保存されているため、このファイルをユーザ用のフォルダにコピーしておく必要がある。この作業を行うために実習ではフォルダやディレクトリの考え方について慣れてもらう必要がある。

ユーザ用のフォルダは Google のサービスのうちドライブ用のフォルダに移すことになる。通常は Google Colaboratory ではドライブと連携していないため python のライブラリを利用してドライブのフォルダをマウントするという作業を行う。この時に必要なのはモデルのファイルがどこにあるのか、コピー先のフォルダディレクトリはどこにあるのかを自分で調べることができるようにすることである。この考え方はオペレーティングシステムのファイルシステムの考え方を学習する際の良い例であると考えている。

保存したモデルを使用し、用意したデータを分類するための準備が整った。この段階ではノートブックで実験している状態である。今後、これを自分以外の人に使ってもらうことでAIを搭載したアプリが完成することになる。これを簡単に実現するためにはWebフレームワークと呼ばれるライブラリを使うべきであろう。pythonではDjango、Flask、FastAPIなどのフレームワークが用意されている。これらのWebフレームワークと連携させることでAI搭載のWebアプリの開発も簡単に行えるものと思われる。今回は新しいライブラリとしてstreamlit[3]というものを使用してみる。

streamlitを利用することでWebアプリを簡単に開発することができるとともに、実際にWebアプリをクラウドのサービスに公開するためのデプロイという作業も簡単に行える。AI搭載のアプリをデプロイ後、顧客をつかむことが大切になってくる。獲得した顧客からの要求を吸い上げ、短いサイクルで要求を即時に実現するようなCI/CD（Continuous Integration ／ Continuous Delivery、継続的インテグレーション／継続的デリバリー）を行っていく。またUI/UX（User Interface ／ User Experience、ユーザインターフェイス／ユーザエクスペリエンス）[4]を用意し顧客が次のバージョンを期待させるような対応を行っていく必要が出てくるだろう。この段階までくると経営の分野に大きく関わってくると思われる。AIを搭載したWebアプリを開発するだけで終わってはいけない。次の顧客を獲得し、その要求に応えられる段階になって製品が完成する。経営におけるAIに関わる製品開発の授業ではこのような姿勢を学生に教えることが必要になってくるだろう。

また学習済みのモデルはHugging face[5]などのサイトで公開されている。似たような分類が行えるモデルがあればこれを利用して自分の目的に合致したAIのアプリが作れる。これを実現するためにディープラーニングの次の段階として転移学習やファインチューニングというものが行われている。転移学習では学習に使ったモデルを利用し新たな分類を行えるようにするものである。例えば犬の分類で使われたモデルの出力層を除去し残りの部分を利用し、猫のデータを用意し猫の分類を行うモデルを作ろうとするものである。この時猫のデータは少量で済む。

ファインチューニングでは新たにラベルを使ってさらに細かく分類できるように再学習させるものと捉えるとよいだろう。例えば、犬と猫の分類を、新たなラベルにより品種の分類までできるようにするものである。このようにAIでアプリを作ろうと考えているものは、一から大量のデータを集め、学習を行わせるよりは、Hugging faceなどのサイトを閲覧し利用可能で公開されているモデルを自分のアプリで使うことができるか市場調査や実験を行い、すでにあるものを使って問題を解決する能力が求められるだろう。

4. Google Colaboratoryを使用したWebアプリの実習例

ここではGoogle Colaboratoryを使用したWebアプリの実習例を示す。

Google Colaboratoryを利用するためにはインターネットに接続できること、gmailのアカウントを持っていることが必要である。

Google Colaboratoryのサイトを開いたら新規にノートブックを作成する。メニューのランタイムのところで「ランタイムのタイプを変更」をクリックし、ハードウェアアクセラレータでT4 GPUを選択する。これでAIアプリを開発する準備が整った。

今回は「PyTorchとfastaiではじめるディープラーニング」[6]の1章の例にある犬と猫を分類するモデルを作成するプログラムを使用した。以下で使用するプログラムはGitHub上で公開されているものに一部コードを追加している。ファイル操作のコマンドやstreamlitのプログラムについては実際に作成したものを紹介する。Google Colaboratoryではこの書籍で使用しているfastaiというライブラリも標準で搭載されている。

streamlitについては次のコマンドによりインストールする。ノートブック上ではコードと呼ばれるセルにコマンドやプログラムを入力し、実行のボタンをクリックして実行する。コードのセルは自動的に追加される。あるいは「＋コード」をクリックする。

図 3. 学習時の実行結果

```
!pip install streamlit
```

fastai のサイトで提供されるファイル 01_intro.ipynb 内の Running Your First Notebook のタイトル下のコードをコピーし実行する。コードはサイトのものを参考にしてほしい。

やっていることは犬と猫の写真とラベルを読み込みディープラーニングにより学習させモデルを作成することである。実行後次の図 3 のような結果が表示される。

実行結果より学習の処理は 2 分程度で終了している。fastai のサイトのノートブックではプログレスバーを表示するようにしている。この図では敢えて表示しないようにしている。これは streamlit で Web アプリを起動する時に ProgressCallback に関わるエラーが発生するためである[7]。従ってコードの最後の行の前に 1 行（次コードの 1 行目）を追加しプログレスバーを使わないようにしている。

```
learn.remove_cb(ProgressCallback)
learn.fine_tune(1)
```

fastai のライブラリを使用すると犬と猫の写真、ラベルを読み込み、ディープラーニングにより学習を行いモデルを作成するコードは 10 行程度で済む。このように fastai では理論よりもまず実践することを勧めている。

次のプログラムにより作成したモデルを保存する。

```
learn.export()
```

このプログラムにより export.pkl というファイルが作成される。このファイルを Web アプリに読み込ませて簡単な AI を搭載した Web アプリを作ってみる。

Web アプリは streamlit を用いて開発するが、streamlit のコードは Google ドライブの Colab Notebooks というフォルダに保存する。従って Web アプリで利用できるようにするために export.pkl を Colab Notebooks というフォルダにコピーする。

次のコマンドで学習用の変数 learn が対象にしているフォルダの場所（パス）を知ることができる。

```
learn.path
```

コマンドの実行結果より作業用のパスが/root/.fastai/data/oxford-iiit-pet/images であることがわかった。

export.pkl を次のコマンドで確認する。

```
%ls /root/.fastai/data/oxford-iiit-pet/images/*.pkl
```

実行結果より export.pkl を確認することができた。

```
/root/.fastai/data/oxford-iiit-pet/images/
export.pkl
```

上記のパスをもとに export.pkl をドライブの Colab Notebooks というフォルダにコピーする。

そのためにノートブックでドライブのフォルダをマウントする。マウントとは外部のフォルダなどを利用できるようにする操作である。次のプログラムによりドライブのフォルダをマウントする。

```
from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)
root_dir = "/content/gdrive/My Drive/"
```

実行後、アクセスするかどうか聞いてくる。「Google ドライブに接続」をクリックする。アカウントの選択のウィンドウが開くので使用中のアカウントをクリックする。許可する項目を確認後、「許可」ボタンをクリックする。

次のコマンドでドライブがマウントされたか確認する。

```
%ls
```

実行結果よりドライブをマウント後、gdrive というフォルダ名を確認することができる。

次のコマンドで Colab Notebooks フォルダ内のファイルを確認する。

```
%ls gdrive/MyDrive/'Colab Notebooks'
```

Colab Notebooks フォルダ内のファイルはこの時点で次のようになっていた。

```
chapter1_cat_example.jpg Untitled0.ipynb
Untitled1.ipynb Untitled2.ipynb
```

chapter1_cat_example.jpg はモデルで判定するための猫の画像ファイル[8]である。拡張子 ipynb は Google Colaboratory で作成したノートブックのファイルである。

このフォルダに export.pkl をコピーする。次のコマンドによりコピーを行う。

```
%cp /root/.fastai/data/oxford-iiit-pet/images/
export.pkl /content/gdrive/MyDrive/'Colab
Notebooks'
```

次のコマンドで Colab Notebooks フォルダ内のファイルに export.pkl が確認できればよい。

```
%ls gdrive/MyDrive/'Colab Notebooks'
```

以上のようにパスを使ったファイル操作は必要なので、パスという考え方を理解できるようにする必要がある。

ここまでの操作で犬と猫の写真を分類するためのモデルを準備がすることができた。

次の段階として作成したモデルを利用して判定用に用意した写真の分類結果を Web アプリを使って表示する部分を作る。

次のコマンドでファイルを出力するためのフォルダを Colab Notebooks フォルダにする。

```
%cd gdrive/MyDrive/'Colab Notebooks'
```

次のプログラムをコードのセルに入力し実行する。

```
%%writefile app.py

import streamlit as st

from fastai.vision.all import *

global is_cat
is_cat = None

def main():

  learn_inf = load_learner('./export.pkl')

  img = PILImage.create('./chapter1_cat_example.jpg')
  is_cat,_,probs = learn_inf.predict(img)

  st.write(f"Is this a cat?: {is_cat}.")
  st.write(f"Probability it's a cat: {probs[1].item():.6f}")

if __name__ == '__main__':
    main()
```

1 行目のプログラムにより上記プログラムがファイル名 app.py で Colab Notebooks フォルダに出力される。

app.py のプログラムについて説明する。

次のプログラムは分類するために必要なモデルのファイルを読み込んでいる。

```
learn_inf = load_learner('./export.pkl')
```

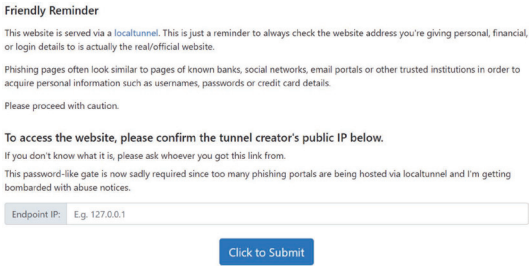

図 4.　IP アドレスを設定する画面

判定が必要な画像ファイルを読み込む。

```
img = PILImage.create('./chapter1_cat_example.jpg')
```

読み込んだ写真のデータを判別する。

```
is_cat,_,probs = learn_inf.predict(img)
```

is_cat という名前を使用した場合、組み込み関数名との競合でエラーが発生することがわかったので次の 2 行を加えている[9]。

```
global is_cat
is_cat = None
```

判定結果は is_cat、判定時の確率は probs で受け取る。表示は streamlit の関数により行う。

書き出した app.py を次のコマンドで実行する。

```
!streamlit run app.py & sleep 3 && npx
localtunnel --port 8501
```

実行するとサーバ起動後、Web アプリが実行され、以下の情報がノートブック上に表示される。

```
Collecting usage statistics. To deactivate, set
browser.gatherUsageStats to False.
  You can now view your Streamlit app in your
browser.

  Network URL: http://IP アドレス 1:8501
  External URL: http://IP アドレス 2:8501
```

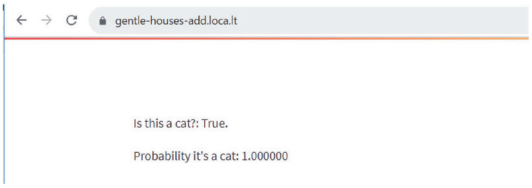

図 5.　Web アプリで表示された判定結果

```
npx: installed 22 in 1.752s
your url is: https://gentle-houses-add.loca.lt
```

your url is:に示されたアドレスをブラウザで開くと、次図 4 のように Endpoint IP:を入力するように要求してくるので上記の表示のうち External URL の IP アドレス 2 を入力し、「Click to Submit」ボタンをクリックする。

図 5 のように Web アプリが実行され判定結果が表示される。

chapter1_cat_example.jpg（猫の写真の画像ファイル）は、確率 1 で猫と判定されたことがわかる。

以上のように分類に必要なデータを集めラベル付けをした後で学習させ、その後 Web アプリと連携させるためにモデルのファイルを出力後、そのモデルのファイルを利用して分類を行うことができた。

簡単なコードにより AI 搭載の Web アプリを実現することができた。

今後は UI/UX を考慮した Web アプリにバージョンアップしデプロイしていくことができるだろう。

また、すでに再利用可能なモデルを探し、転移学習やファインチューニングにより自分の目的に合った AI 搭載のアプリを開発していくことも可能だと思われる。

このような Web アプリの開発スタイルは経営学部でも十分に対応できるものと思われる。

4.　まとめ

経営学部での AI の実習は実践的なものが望ま

しい。AI のモデルを開発するにあたり、Google Colaboratory のようなソフトウェア開発環境、ハードウェア資源を利用すべきであろう。また fastai のようなライブラリを使用することで、詳細よりも基本的な AI の考え方を身につけるべきであろう。また、streamlit のようなライブラリを使用して、AI のモデルによる分類、判別機能を Web アプリとして実現できる仕組みに慣れておくことも必要だろう。これらを行うために python プログラミング、ファイルシステムの考え方については習熟しておくべきだろう。今後は Hugging face などのサイトで実用化、公開されているモデルを使って転移学習、ファインチューニングを行い、UI/UX を積極的に採用し、CI/CD を意識しながら Web アプリをデプロイできることを目標としてもらいたい。

5. 注釈、文献リスト

(1) streamlit と graphvis によりニューラルネットワークの図を作成した。
図を作成するにあたり以下のサイトの情報を参考にした。
https://gist.github.com/thigm85/5760134
(2) ランタイムの環境はノートブックごとに設定できる。GPU を選択した複数のノートブックを使用した場合、セッションが多すぎるとメッセージが出ることがある。このようにリソースの管理が厳しく行われている。
(3) stremlit のサイト
https://streamlit.io/
インストールは python の pip を使用して行う。

API reference のページに様々な例がある。python のライブラリと親和性が高い。
https://docs.streamlit.io/library/api-reference
(4) ユーザインターフェイス（UI）はソフトウェアとユーザが直接接触を持つ部分と考えるとよい。ボタンや入力欄などをイメージして欲しい。ユーザエクスペリエンス（UX）はユーザが新しい体験をできるようにする仕組みで、使いやすさ、顧客を獲得するための手法を用いてサービスを提供する。
(5) Hugging face のサイト
https://huggingface.co/

学習済みモデル、AI を搭載したアプリケーション、データセットなどを公開している。
今後の AI のあり方について考えるためには、このサイトの動向を追いかけていく必要があるだろう。
(6) PyTorch と fastai ではじめるディープラーニング、Jeremy Howard、Sylvain Gugger 著、中田 秀基訳、2021、オライリージャパン
最初に AI に触れて使ってみる、次に少しずつより詳しく勉強していく姿勢を勧めている。

書籍のソフトウェアなどのサポートは以下で行っている。
https://github.com/fastai/fastbook

書籍のソフトウェアに関するフォーラムでは以下のサイトで活発に質問や意見交換が行われている。
https://forums.fast.ai/
(7) progresscallback に関するエラーの訂正は下記のページを参考にした。

AttributeError: 'NBProgressBar' object has no attribute 'start_t' #72
https://github.com/fastai/fastprogress/issues/72
(8) この画像ファイルは(6)の fastai のサポートサイトで公開されている。
(9) 名前空間に関するエラーの訂正は下記のページを参考にした。

AttributeError: Can't get attribute 'my_func' on < module '__main__' from 'main.py'>
https://stackoverflow.com/questions/72899222/attributeerror-cant-get-attribute-my-func-on-module-main-from-main-p