Python Handout for EE2211

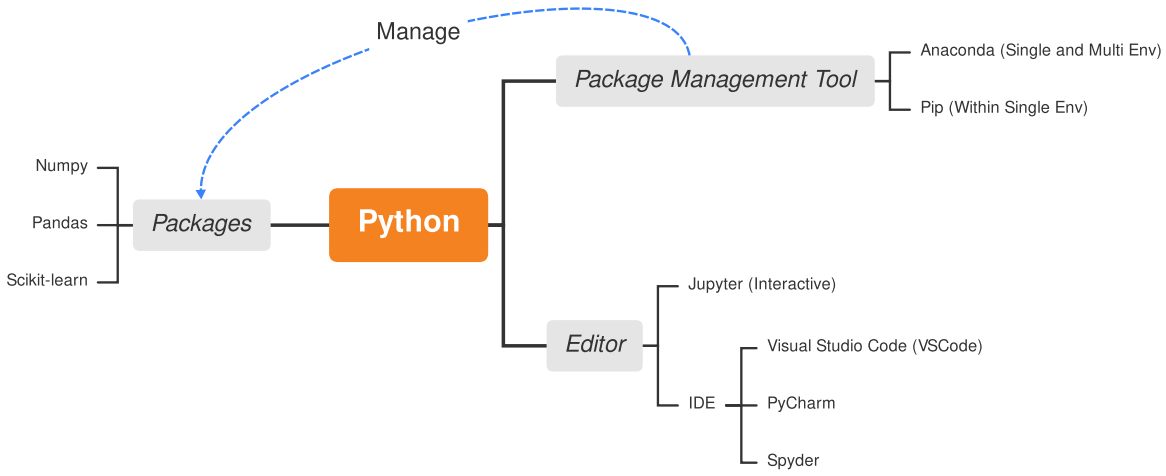
Prepared by Xingyi Yang

August 13, 2023

1 Overview

First things first, we need to learn some basic concepts in **Python**.

- `Python`: A programming language. We write code with python.
- `Editor`: A tool where you write your code.
- `Packages`: Code written by other people. We can import them into our code to avoid "build wheel". For example, `numpy` is a mathematical package that supports diverse matrix calculations.
- `Package Management Tool`: Package A might depend on another Package B. Package Management Tool automatically detects the dependency tree and addresses it.



2 Anaconda

Now, let's talk about Anaconda[1][2]. TL;DR, it is both a package or dependency manager as well as an environment manager.

[1] `https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)`
[2] `https://en.wikipedia.org/wiki/Conda_(package_manager)`

2.1 Installation

After installing Anaconda [3], you will probably find `python=3.8.8` in your base(root) environment as shown in the Anaconda Navigator under the environment tab. In this module we will use

```
python=3.8 numpy=1.24
```

So let's create a new environment for EE2211 and we will call it `ee2211`. To do this we enter the following command in the terminal (for Windows user search `Anaconda Powershell Prompt`, for MacOS user just search for `terminal`)

```
conda create --name ee2211 python=3.8 numpy=1.24 matplotlib scikit-learn pandas
```

Now in the conda navigator you should find a new environment we just created `ee2211`, with the version we want :-)

Go ahead and activate the `ee2211` environment, by the following command

```
conda activate ee2211
```

In order to view the Jupyter notebook, let's install it in ee2211

```
conda install -c conda-forge jupyterlab
```

2.2 FAQ

Q. How to start Jupyter notebook?
A. In the terminal enter the command `jupyter-lab` or `jupyter lab`. It will start in your web browser.

Q. I encounter this error in VSCode "Importing the numpy c-extensions faile", what should I do?
A. Referring to the answer here[4].

1. Open VS Code's Command Palette menu by pressing Ctrl+Shift+P or F1.
2. Choose "Terminal: Select Default Profile" entry.
3. Then pick "Command Prompt" option.
4. Restart VS Code.

Q. How to select the `ee2211` environment in VSCode/PyCharm/Spyder?
A. For VSCode, refer to this[5].
For PyCharm, refer to this[6].
For Spyder, refer to this[7].

Q. [Errno 2] No such file or directory.
A. Enter the correct path.

3 Importing Modules

As your program gets longer, you may want to split it into several files for easier maintenance. Such a file is called a *module*; definitions from a module can be imported into other modules or into the main module. For example, we can define a `fino.py` module

```
# Fibonacci numbers module
def fib(n):    # write Fibonacci series up to n
    a, b = 0, 1
```

[3] https://docs.anaconda.com/anaconda/install/index.html
[4] https://stackoverflow.com/questions/58868528/importing-the-numpy-c-extensions-failed
[5] https://www.jetbrains.com/help/pycharm/conda-support-creating-conda-virtual-environment.html
[6] https://www.jetbrains.com/help/pycharm/conda-support-creating-conda-virtual-environment.html
[7] https://docs.spyder-ide.org/current/installation.html#using-anaconda

```
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()

def fib2(n):   # return Fibonacci series up to n
    result = []
    a, b = 0, 1
    while a < n:
        result.append(a)
        a, b = b, a+b
    return result
```

Now enter the Python interpreter and import this module with the following command

```
>>> import fibo
>>> # OR from fibo import fib, fib2
>>> # OR from fibo import *
>>> fibo.fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
>>> fibo.fib2(100)
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
>>> fibo.__name__
'fibo'
```

Then we can install pre-defined modules and packages, and re-use the code by importing.

`https://docs.python.org/3/tutorial/modules.html`

4 Python & Numpy

Please refer to the examples in the `Jupyter Notebook`. Here are some awesome resources,

`https://scipy-lectures.org/index.html`
`https://numpy.org/doc/1.20/user/basics.creation.html`
`https://numpy.org/doc/1.20/user/basics.indexing.html`
`https://numpy.org/doc/1.20/user/basics.broadcasting.html`
`https://numpy.org/doc/1.20/reference/generated/numpy.linalg.inv.html`
`https://numpy.org/doc/1.20/reference/generated/numpy.linalg.pinv.html`
`https://docs.scipy.org/doc/scipy/reference/tutorial/stats.html`

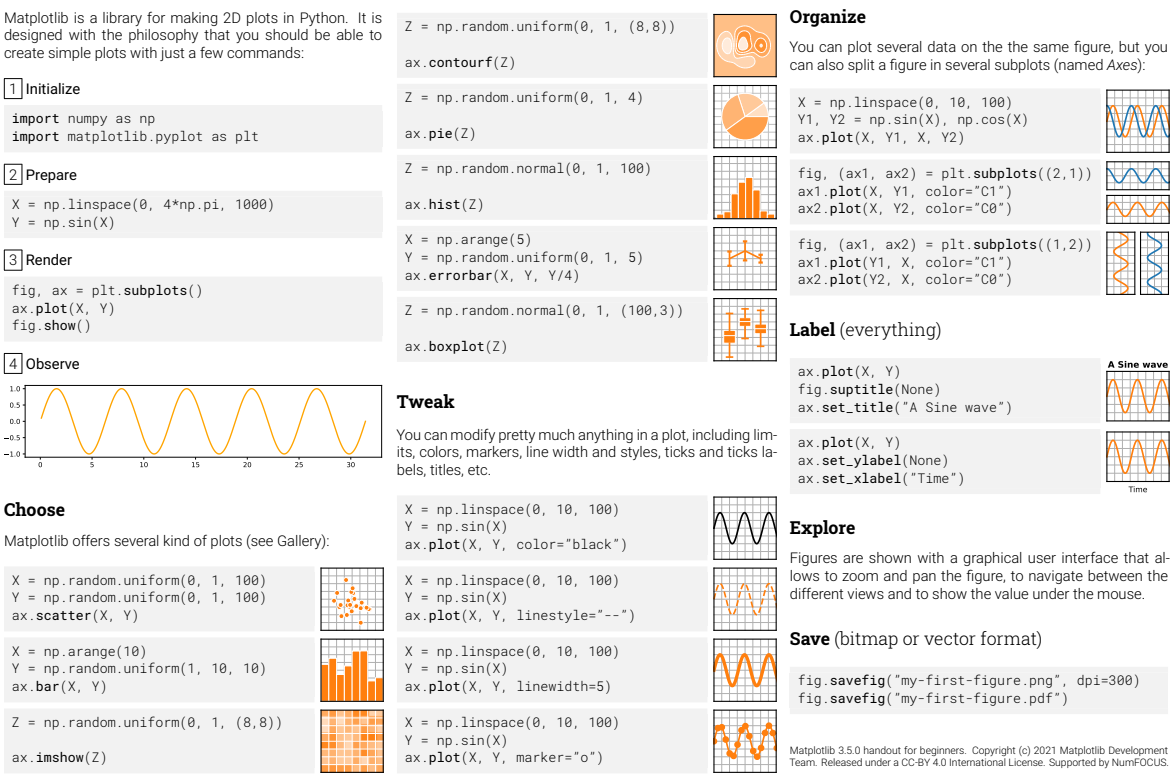
Knowing them is not a must and you should definitely not consume them all at one go. However, if you find yourself stuck on your assignment or tutorial questions, you can most probably find the answer there.

5 Plotting with Matplotlib

`Matplotlib` is a plotting library. This section gives a brief introduction to the `matplotlib.pyplot` module. Plotting is an indispensable part of data visualization. A picture tells a thousand words, and a good picture tells more. What is presented in the official matplotlib cheatsheet[8] is sufficient for this module. Refer to Fig. 2.
To further your plotting skill, you can start from the examples[9] and tweak them to your advantage, of course,

[8] `https://github.com/matplotlib/cheatsheets`
[9] `https://matplotlib.org/stable/gallery/index.html`

it will pay off to go through the basic tutorials[10].
Plus: plotting is not tested in exams



Matplotlib for beginners

Matplotlib is a library for making 2D plots in Python. It is designed with the philosophy that you should be able to create simple plots with just a few commands:

1 Initialize

```
import numpy as np
import matplotlib.pyplot as plt
```

2 Prepare

```
X = np.linspace(0, 4*np.pi, 1000)
Y = np.sin(X)
```

3 Render

```
fig, ax = plt.subplots()
ax.plot(X, Y)
fig.show()
```

4 Observe

Choose

Matplotlib offers several kind of plots (see Gallery):

```
X = np.random.uniform(0, 1, 100)
Y = np.random.uniform(0, 1, 100)
ax.scatter(X, Y)
```

```
X = np.arange(10)
Y = np.random.uniform(1, 10, 10)
ax.bar(X, Y)
```

```
Z = np.random.uniform(0, 1, (8,8))

ax.imshow(Z)
```

```
Z = np.random.uniform(0, 1, (8,8))

ax.contourf(Z)
```

```
Z = np.random.uniform(0, 1, 4)

ax.pie(Z)
```

```
Z = np.random.normal(0, 1, 100)

ax.hist(Z)
```

```
X = np.arange(5)
Y = np.random.uniform(0, 1, 5)
ax.errorbar(X, Y, Y/4)
```

```
Z = np.random.normal(0, 1, (100,3))

ax.boxplot(Z)
```

Tweak

You can modify pretty much anything in a plot, including limits, colors, markers, line width and styles, ticks and ticks labels, titles, etc.

```
X = np.linspace(0, 10, 100)
Y = np.sin(X)
ax.plot(X, Y, color="black")
```

```
X = np.linspace(0, 10, 100)
Y = np.sin(X)
ax.plot(X, Y, linestyle="--")
```

```
X = np.linspace(0, 10, 100)
Y = np.sin(X)
ax.plot(X, Y, linewidth=5)
```

```
X = np.linspace(0, 10, 100)
Y = np.sin(X)
ax.plot(X, Y, marker="o")
```

Organize

You can plot several data on the the same figure, but you can also split a figure in several subplots (named *Axes*):

```
X = np.linspace(0, 10, 100)
Y1, Y2 = np.sin(X), np.cos(X)
ax.plot(X, Y1, X, Y2)
```

```
fig, (ax1, ax2) = plt.subplots((2,1))
ax1.plot(X, Y1, color="C1")
ax2.plot(X, Y2, color="C0")
```

```
fig, (ax1, ax2) = plt.subplots((1,2))
ax1.plot(Y1, X, color="C1")
ax2.plot(Y2, X, color="C0")
```

Label (everything)

```
ax.plot(X, Y)
fig.suptitle(None)
ax.set_title("A Sine wave")
```

```
ax.plot(X, Y)
ax.set_ylabel(None)
ax.set_xlabel("Time")
```

Explore

Figures are shown with a graphical user interface that allows to zoom and pan the figure, to navigate between the different views and to show the value under the mouse.

Save (bitmap or vector format)

```
fig.savefig("my-first-figure.png", dpi=300)
fig.savefig("my-first-figure.pdf")
```

Matplotlib 3.5.0 handout for beginners. Copyright (c) 2021 Matplotlib Development Team. Released under a CC-BY 4.0 International License. Supported by NumFOCUS.

6 Pandas

`Pandas` is a tool for manipulating and analyzing tabular data. It is one of the most widely-used packages in data science programming. Here are some awesome resources,

- Installation: `https://pandas.pydata.org/getting_started.html`
- Official Documentation: `https://pandas.pydata.org/docs/`
- Pandas Tutorial: `https://www.w3schools.com/python/pandas/default.asp`
- Pandas Course on Kaggle: `https://www.kaggle.com/learn/pandas`

7 Scikit-Learn

Scikit-Learn (Sklearn) is a python tool for machine learning and data analysis. Sklearn could be used for classification, regression, clustering, dimension reduction, and data pre-processing. Here are some awesome resources,

- Installation: `https://scikit-learn.org/stable/install.html`
- Official Documentation: `https://scikit-learn.org/stable/index.html#`

[10] `https://matplotlib.org/stable/tutorials/index.html`

- Examples: `https://scikit-learn.org/stable/auto_examples/index.html`
- Official Tutorials: `https://scikit-learn.org/stable/tutorial/index.html`

8 Assignment Guidance

Assuming that the question asks us to compute the sum of 2 matrices, $\mathbf{X}$ and $\mathbf{Y}$ and return the answer $\mathbf{W}$ as a numpy array. $\mathbf{X} \in \mathbb{R}^{5\times2}$, $\mathbf{Y} \in \mathbb{R}^{5\times2}$. Suppose your assignment answer file that is to be submitted is as follows,

```
import numpy as np
def A1_A1234567X (X, Y):
    """
    Input type
    : X type : numpy . ndarray
    : Y type : numpy . ndarray
    Return type
    : W type : numpy . ndarray
    """

    # your code goes here
    W = X + Y
    # return in this order
    return W
```

This seems okay, but before submitting this file, please run the code to make sure. The suggested way to run the code is by importing this function.

In the "test.py" file write the following,

```
import numpy as np
from A1234567X import A1_A1234567X as grading

# create some dummy inputs for testing
x = np.random.randn(5, 2)
y = np.random.randn(5, 2)
w = grading(x, y)
print(type(w))
print(w)
```

Then run "test.py" either from the python console or terminal, the output will be like this,

```
Traceback (most recent call last):
File "test.py", line 7, in <module>
    w = grading(X , y)
File "/ A1_demo / A1_A1234567X .py", line 12, in A1_A1234567X
    w = x + y
NameError : name "x" is not defined
```

This error message is very informative, it tells us that in "A1_A1234567X.py" file, line 12, we made a mistake. The mistake is called `NameError` which means that we have used an undefined variable. And indeed we have mistakenly wrote x instead of X.
After rectifying the error, the output of "test.py" should look like this

```
<class 'numpy.ndarray'>
[[ -0.46911376   0.75976783]
 [ -0.9546599    -1.57550369]
 [ 1.50917485    0.14661074]
 [ -0.3125534    0.52009137]
```

```
 [ -0.66608823  1.14272 ]]
```

The first line of the output corresponds to "print(type(w))". This is to make sure that our return type fulfills the requirement as specified in the template code. In the "test.py", we can also print out X and y to check if w is indeed their sum.

The above method of debugging is recommended because it best simulates the grading condition. And we urge you all to test your code in this way before submitting. Most of you may be more comfortable debugging in the following way. Or use jupyter notebook or use visual studio code or use pycharm ... all of them are fine.

```
1  import numpy as np
2  # Please replace "MatricNumber" with your actual matric number here and in the filename
3  def A1_A1234567X(X, y):
4      """
5      Input type
6      :X type : numpy.ndarray
7      :y type : numpy.ndarray
8      Return type
9      :w type : numpy.ndarray
10     """
11
12     # your code goes here
13     w = X + y
14     # return in this order
15     return w
16
17 X = np.random.randn(5, 2)
18 y = np.random.randn(5, 2)
19 w = A1_A1234567X (X, y)
20 print(type(w))
21 print(w)
```

You can debug this way, but please remember to **DELETE** the lines for debugging (i.e. line 17-21). The final script for submission should only include the function and the necessary imports as follows.

```
1  import numpy as np
2  def A1_A1234567X(X, y):
3      """
4      Input type
5      :X type : numpy.ndarray
6      :y type : numpy.ndarray
7      Return type
8      :w type : numpy.ndarray
9      """
10
11     # your code goes here
12     w = X + y
13     # return in this order
14     return w
```

9 Some Pointers

- In the template code we specify some input type and return type. The input type is ensured by the grader, so you can safely assume that X and y will be numpy arrays and it's not necessary to do another typecasting like `X = np.array(X)`. However, it is your responsibility to make sure that the w returned is in the correct return type.

- A function's goal is to manipulate the input to get the desired output. So the input should not be redefined in the function.
- Regarding `import`: in the future assignment, you will need to import packages other than `numpy`. You can import all these packages at the very top, before and not inside the function declaration.
- `from xx import yy as zz`: the `xx` is the filename without the `.py` extension, `yy` is the function name written inside `xx`, `zz` is any name that is convenient to you. `from xx import yy as zz` works when `xx.py` and `test.py` are under the same folder.
- Please make sure your submitted code does not include any lines used for debugging.
- Show the file extension, avoid having files saved as `A0134567X.py.py`
- And definitely don't submit in `.ipynb` format.

Happy coding!

10 Extended Reading: Example for Binary Classification

To demonstrate how all the packages can be integrated to conduct a machine learning task, we provide a Python example to do two-dimensional classification in the following code.

We first create a 2-dimensional toy classification dataset with `sklearn` and save it to the local file "data.csv" for further usage. Then we visualize it with `matplotlib`. In the end, we use a neural network classifier to fit the binary classification task and visualize its probability in the 2D plane.

Note that, at this point in time, we do not explain the details for the model and advanced API, but leave it to later classes. If you are interested in the example, please contact the GAs or resort to the above-mentioned resources for help.

[21]:
```
from sklearn.datasets import make_blobs
import numpy as np
import pandas as pd
# Generate a toy example using sklearn
X, y = make_blobs(n_samples=1000, centers=20, random_state=123)
labels = ["b", "r"]
y = np.take(labels, (y < 10))
# Print the data and label sizes
print(X.shape)
print(y)
```

```
(1000, 2)
(1000,)
```

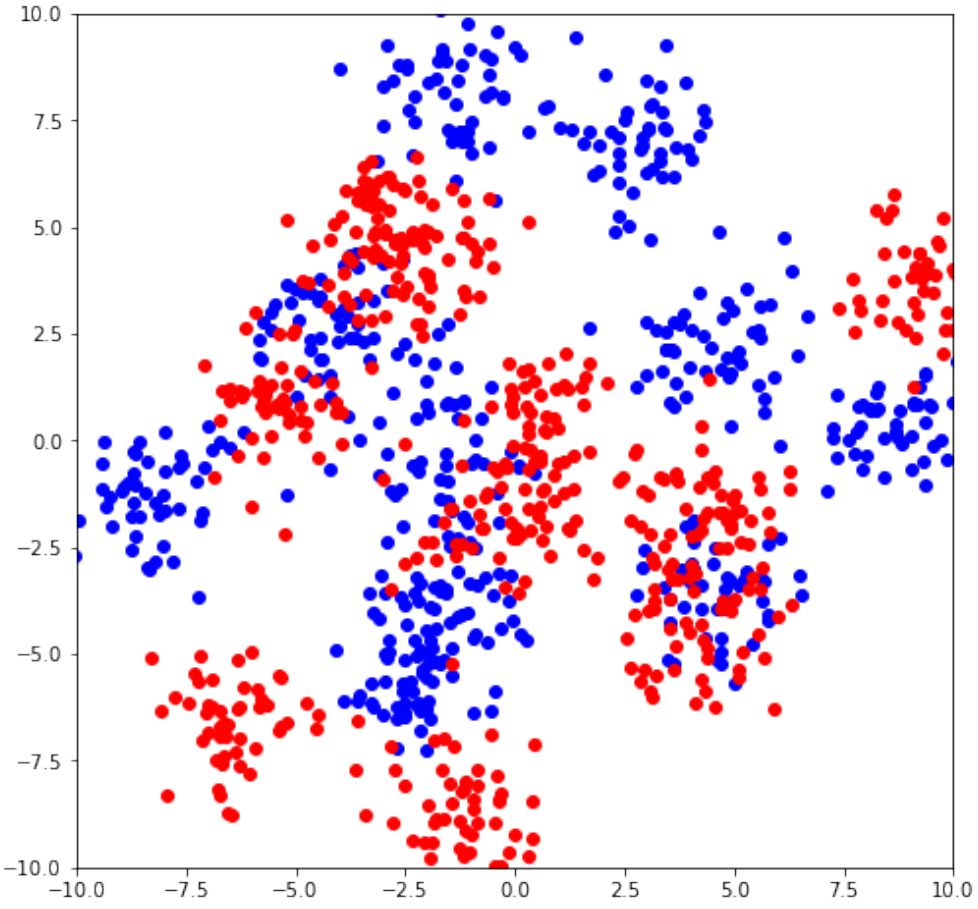
[23]:
```
Y = np.expand_dims(y, axis=-1)
# Concatenate the input data and label.

data = np.concatenate([X, Y], axis=-1)
# Convert it to pandas.Dataframe and save to local file.
data = pd.DataFrame(data, columns=['x','y', 'label'])
data.to_csv("data.csv")

# Print the first 10 samples.
data.head(10)
```

[23]:

	x	y	label
0	-6.45255646603369	-8.763582588004452	r
1	0.2898214122160939	0.14677196152391225	r

```
2   -5.184122930268115  -1.2534702518970096     b
3   -4.713888470809617    3.674404625516476     r
4    4.515582960628491   -2.881380327190773     b
5  -0.5765894463501406    4.615896711742838     r
6   0.4202627550782958   -7.090973475638641     r
7    4.980083613605647  -1.4225720752999897     r
8    4.680253783136185  -1.2582081273528194     r
9    -2.16028169667958    2.751322646270088     r
```

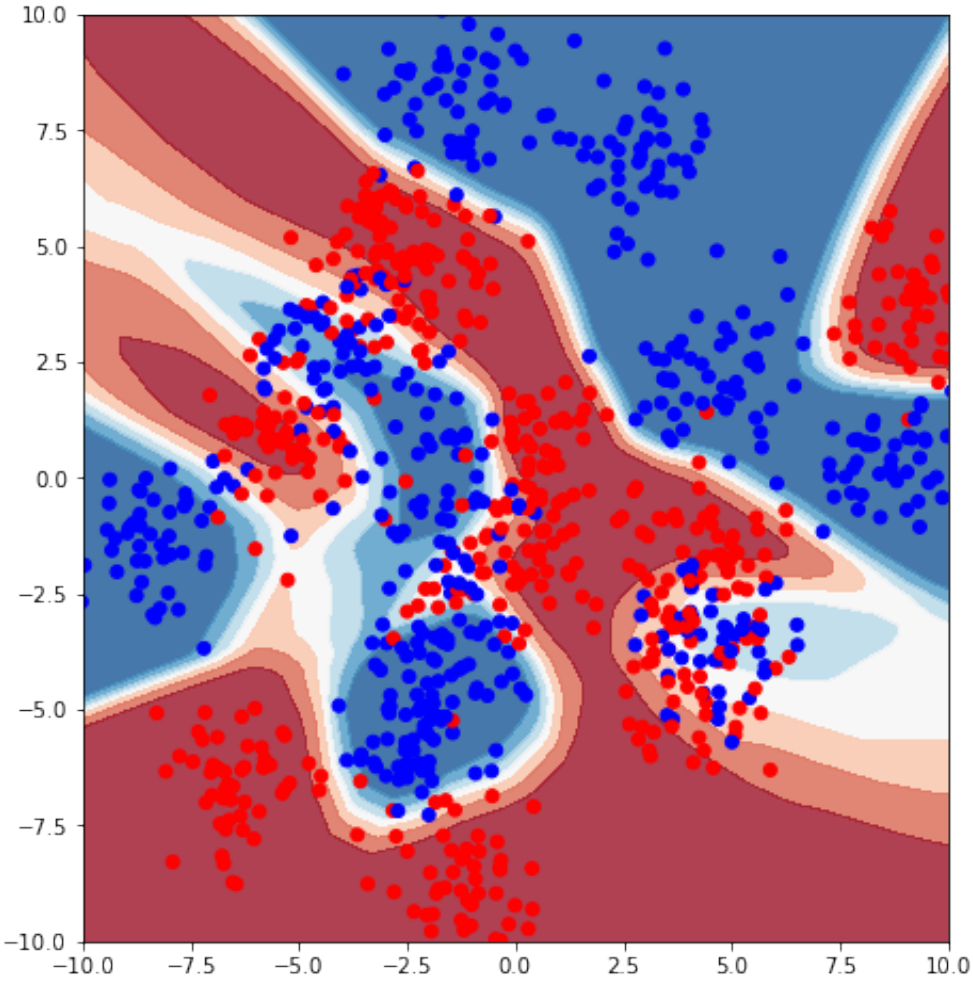
[14]:
```
import matplotlib.pyplot as plt
# Visualize the samples by its category.
plt.figure(figsize=(8, 8))
for label in labels:
    mask = (y == label)
    plt.scatter(X[mask, 0], X[mask, 1], c=label)
plt.xlim(-10, 10)
plt.ylim(-10, 10)
plt.show()
```



[12]:
```
from matplotlib import pyplot as plt
import numpy as np

# Function to plot the classification probability.
```

```
def plot_surface(clf, X, y,
                 xlim=(-10, 10), ylim=(-10, 10), n_steps=250,
                 subplot=None, show=True):
    if subplot is None:
        fig = plt.figure()
    else:
        plt.subplot(*subplot)

    xx, yy = np.meshgrid(np.linspace(xlim[0], xlim[1], n_steps),
                         np.linspace(ylim[0], ylim[1], n_steps))

    if hasattr(clf, "decision_function"):
        z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
    else:
        z = clf.predict_proba(np.c_[xx.ravel(), yy.ravel()])[:, 1]

    z = z.reshape(xx.shape)
    plt.contourf(xx, yy, z, alpha=0.8, cmap=plt.cm.RdBu_r)
    plt.scatter(X[:, 0], X[:, 1], c=y)
    plt.xlim(*xlim)
    plt.ylim(*ylim)

    if show:
        plt.show()


def plot_clf(clf, X, y):
    plt.figure(figsize=(16, 8))
    plot_surface(clf, X, y, subplot=(1, 2, 1), show=False)
```

[13]:
```
from sklearn.neural_network import MLPClassifier

# Define a multi-layer perception to fit the data.
clf = MLPClassifier(hidden_layer_sizes=(100, 100, 100), activation="relu",␣
↪learning_rate="invscaling")
clf.fit(X, y)
plot_clf(clf, X, y)
```


10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0