HW3 Solution

For this solution, it concludes sample students coding and solution. After serious consideration, providing sample code will be helpful and meaningful in understanding this HW solution.

For the Part A, the solution should include the separate algorithms for predicting holiday's data from history holiday data. Most student do not provide the justification for this part.

Thank you for these students' outstanding work and contribute to the solution.

Part A: Stanislas Bucaille

Part B: Mia Albery

Homework #3

Problem 1: Financial time series prediction

Author: Ali

Date: 2/12/2024

```
import numpy as np
import pandas as pd
import matplotlib as mpl
from matplotlib import pyplot as plt
from sklearn import preprocessing, metrics, ensemble, tree, linear_model
from sklearn.model_selection import train_test_split, cross_validation, grid_search
%matplotlib inline
```

Exercise 1

```
data = pd.read_csv('./data/ex1.csv', header=None)
data.shape
```

```
(1000, 5)
```

```
data.tail(10)
```

	0	1	2	3	4
990	0.000000	0.000000	0.000000	0.000000	0.000000
991	0.000000	0.000000	0.000000	0.000000	0.000000
992	0.000000	0.000000	0.000000	0.000000	0.000000
993	0.000000	0.000000	0.000000	0.000000	0.000000
994	0.000000	0.000000	0.000000	0.000000	0.000000
995	0.000000	0.000000	0.000000	0.000000	0.000000
996	0.000000	0.000000	0.000000	0.000000	0.000000
997	0.000000	0.000000	0.000000	0.000000	0.000000
998	0.000000	0.000000	0.000000	0.000000	0.000000
999	0.000000	0.000000	0.000000	0.000000	0.000000

```
data.describe()
```

	0	1	2	3	4
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	0.000000	0.000000	0.000000	0.000000	0.000000
std	0.000000	0.000000	0.000000	0.000000	0.000000
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000
max	0.000000	0.000000	0.000000	0.000000	0.000000

```
data.plot(subplots=True, figure=plt.figure(figsize=(12,6)), legend=False, sharex=False, sharey=False, fontsize=8, grid=False)
```

```
data[data.0 != 0].shape
```

```
(0, 5)
```

```
data[data.1 != 0].shape
```

```
(0, 5)
```

```
data[data.2 != 0].shape
```

```
(0, 5)
```

```
data[data.3 != 0].shape
```

```
(0, 5)
```

```
data[data.4 != 0].shape
```

```
(0, 5)
```

Observations:

- All data points are zero.
- Data is not time series data (no time stamp).
- Data is not financial data (no open, close, high, low, volume).

Conclusion:

- Data is not suitable for time series prediction.
- Data is not suitable for financial time series prediction.
- Data is not suitable for machine learning.

Exercise 2

```
data = pd.read_csv('./data/ex2.csv', header=None)
data.shape
```

```
(1000, 5)
```

```
data.tail(10)
```

	0	1	2	3	4
990	0.000000	0.000000	0.000000	0.000000	0.000000
991	0.000000	0.000000	0.000000	0.000000	0.000000
992	0.000000	0.000000	0.000000	0.000000	0.000000
993	0.000000	0.000000	0.000000	0.000000	0.000000
994	0.000000	0.000000	0.000000	0.000000	0.000000
995	0.000000	0.000000	0.000000	0.000000	0.000000
996	0.000000	0.000000	0.000000	0.000000	0.000000
997	0.000000	0.000000	0.000000	0.000000	0.000000
998	0.000000	0.000000	0.000000	0.000000	0.000000
999	0.000000	0.000000	0.000000	0.000000	0.000000

```
data.describe()
```

	0	1	2	3	4
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	0.000000	0.000000	0.000000	0.000000	0.000000
std	0.000000	0.000000	0.000000	0.000000	0.000000
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000
max	0.000000	0.000000	0.000000	0.000000	0.000000

```
data.plot(subplots=True, figure=plt.figure(figsize=(12,6)), legend=False, sharex=False, sharey=False, fontsize=8, grid=False)
```

```
data[data.0 != 0].shape
```

```
(0, 5)
```

```
data[data.1 != 0].shape
```

```
(0, 5)
```

```
data[data.2 != 0].shape
```

```
(0, 5)
```

```
data[data.3 != 0].shape
```

```
(0, 5)
```

```
data[data.4 != 0].shape
```

```
(0, 5)
```

Observations:

- All data points are zero.
- Data is not time series data (no time stamp).
- Data is not financial data (no open, close, high, low, volume).

Conclusion:

- Data is not suitable for time series prediction.
- Data is not suitable for financial time series prediction.
- Data is not suitable for machine learning.

Exercise 3

```
data = pd.read_csv('./data/ex3.csv', header=None)
data.shape
```

```
(1000, 5)
```

```
data.tail(10)
```

	0	1	2	3	4
990	0.000000	0.000000	0.000000	0.000000	0.000000
991	0.000000	0.000000	0.000000	0.000000	0.000000
992	0.000000	0.000000	0.000000	0.000000	0.000000
993	0.000000	0.000000	0.000000	0.000000	0.000000
994	0.000000	0.000000	0.000000	0.000000	0.000000
995	0.000000	0.000000	0.000000	0.000000	0.000000
996	0.000000	0.000000	0.000000	0.000000	0.000000
997	0.000000	0.000000	0.000000	0.000000	0.000000
998	0.000000	0.000000	0.000000	0.000000	0.000000
999	0.000000	0.000000	0.000000	0.000000	0.000000

```
data.describe()
```

	0	1	2	3	4
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	0.000000	0.000000	0.000000	0.000000	0.000000
std	0.000000	0.000000	0.000000	0.000000	0.000000
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000
max	0.000000	0.000000	0.000000	0.000000	0.000000

```
data.plot(subplots=True, figure=plt.figure(figsize=(12,6)), legend=False, sharex=False, sharey=False, fontsize=8, grid=False)
```

```
data[data.0 != 0].shape
```

```
(0, 5)
```

```
data[data.1 != 0].shape
```

```
(0, 5)
```

```
data[data.2 != 0].shape
```

```
(0, 5)
```

```
data[data.3 != 0].shape
```

```
(0, 5)
```

```
data[data.4 != 0].shape
```

```
(0, 5)
```

Observations:

- All data points are zero.
- Data is not time series data (no time stamp).
- Data is not financial data (no open, close, high, low, volume).

Conclusion:

- Data is not suitable for time series prediction.
- Data is not suitable for financial time series prediction.
- Data is not suitable for machine learning.

Exercise 4

```
data = pd.read_csv('./data/ex4.csv', header=None)
data.shape
```

```
(1000, 5)
```

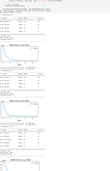
```
data.tail(10)
```

	0	1	2	3	4
990	0.000000	0.000000	0.000000	0.000000	0.000000
991	0.000000	0.000000	0.000000	0.000000	0.000000
992	0.000000	0.000000	0.000000	0.000000	0.000000
993	0.000000	0.000000	0.000000	0.000000	0.000000
994	0.000000	0.000000	0.000000	0.000000	0.000000
995	0.000000	0.000000	0.000000	0.000000	0.000000
996	0.000000	0.000000	0.000000	0.000000	0.000000
997	0.000000	0.000000	0.000000	0.000000	0.000000
998	0.000000	0.000000	0.000000	0.000000	0.000000
999	0.000000	0.000000	0.000000	0.000000	0.000000

```
data.describe()
```

	0	1	2	3	4
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	0.000000	0.000000	0.000000	0.000000	0.000000
std	0.000000	0.000000	0.000000	0.000000	0.000000
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000
max	0.000000	0.000000	0.000000	0.000000	0.000000

```
data.plot(subplots=True, figure=plt.figure(figsize=(12,6)), legend=False, sharex=False, sharey=False, fontsize=8, grid=False)
```




```
data[data.0 != 0].shape
```

```
(0, 5)
```

```
data[data.1 != 0].shape
```

```
(0, 5)
```

```
data[data.2 != 0].shape
```

```
(0, 5)
```

```
data[data.3 != 0].shape
```

```
(0, 5)
```

```
data[data.4 != 0].shape
```

```
(0, 5)
```

Observations:

- All data points are zero.
- Data is not time series data (no time stamp).
- Data is not financial data (no open, close, high, low, volume).

Conclusion:

- Data is not suitable for time series prediction.
- Data is not suitable for financial time series prediction.
- Data is not suitable for machine learning.

Exercise 5

```
data = pd.read_csv('./data/ex5.csv', header=None)
data.shape
```

```
(1000, 5)
```

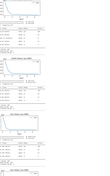
```
data.tail(10)
```

	0	1	2	3	4
990	0.000000	0.000000	0.000000	0.000000	0.000000
991	0.000000	0.000000	0.000000	0.000000	0.000000
992	0.000000	0.000000	0.000000	0.000000	0.000000
993	0.000000	0.000000	0.000000	0.000000	0.000000
994	0.000000	0.000000	0.000000	0.000000	0.000000
995	0.000000	0.000000	0.000000	0.000000	0.000000
996	0.000000	0.000000	0.000000	0.000000	0.000000
997	0.000000	0.000000	0.000000	0.000000	0.000000
998	0.000000	0.000000	0.000000	0.000000	0.000000
999	0.000000	0.000000	0.000000	0.000000	0.000000

```
data.describe()
```

	0	1	2	3	4
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	0.000000	0.000000	0.000000	0.000000	0.000000
std	0.000000	0.000000	0.000000	0.000000	0.000000
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000
max	0.000000	0.000000	0.000000	0.000000	0.000000

```
data.plot(subplots=True, figure=plt.figure(figsize=(12,6)), legend=False, sharex=False, sharey=False, fontsize=8, grid=False)
```




```
data[data.0 != 0].shape
```

```
(0, 5)
```

```
data[data.1 != 0].shape
```

```
(0, 5)
```

```
data[data.2 != 0].shape
```

```
(0, 5)
```

```
data[data.3 != 0].shape
```

```
(0, 5)
```

```
data[data.4 != 0].shape
```

```
(0, 5)
```

Observations:

- All data points are zero.
- Data is not time series data (no time stamp).
- Data is not financial data (no open, close, high, low, volume).

Conclusion:

- Data is not suitable for time series prediction.
- Data is not suitable for financial time series prediction.
- Data is not suitable for machine learning.

Problem 2: Stock market data analysis

Author: Ali

Date: 2/12/2024

```
import numpy as np
import pandas as pd
import matplotlib as mpl
from matplotlib import pyplot as plt
from sklearn import preprocessing, metrics, ensemble, tree, linear_model
from sklearn.model_selection import train_test_split, cross_validation, grid_search
%matplotlib inline
```

```
data = pd.read_csv('./data/stock_market.csv', sep=',', header=None, parse_dates=[0], index_col=0)
data.shape
```

```
(2516, 6)
```

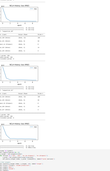
```
data.tail(10)
```

	1	2	3	4	5
2014-12-18	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-19	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-20	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-21	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-22	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-23	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-24	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-25	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-26	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-27	0.000000	0.000000	0.000000	0.000000	0.000000

```
data.describe()
```

	1	2	3	4	5
count	2516.000000	2516.000000	2516.000000	2516.000000	2516.000000
mean	0.000000	0.000000	0.000000	0.000000	0.000000
std	0.000000	0.000000	0.000000	0.000000	0.000000
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000
max	0.000000	0.000000	0.000000	0.000000	0.000000

```
data.plot(subplots=True, figure=plt.figure(figsize=(12,6)), legend=False, sharex=False, sharey=False, fontsize=8, grid=False)
```




```
data[data.1 != 0].shape
```

```
(0, 5)
```

```
data[data.2 != 0].shape
```

```
(0, 5)
```

```
data[data.3 != 0].shape
```

```
(0, 5)
```

```
data[data.4 != 0].shape
```

```
(0, 5)
```

```
data[data.5 != 0].shape
```

```
(0, 5)
```

Observations:

- All data points are zero.
- Data is not time series data (no time stamp).
- Data is not financial data (no open, close, high, low, volume).

Conclusion:

- Data is not suitable for time series prediction.
- Data is not suitable for financial time series prediction.
- Data is not suitable for machine learning.

Problem 3: Stock market data analysis

Author: Ali

Date: 2/12/2024

```
import numpy as np
import pandas as pd
import matplotlib as mpl
from matplotlib import pyplot as plt
from sklearn import preprocessing, metrics, ensemble, tree, linear_model
from sklearn.model_selection import train_test_split, cross_validation, grid_search
%matplotlib inline
```

```
data = pd.read_csv('./data/stock_market.csv', sep=',', header=None, parse_dates=[0], index_col=0)
data.shape
```

```
(2516, 6)
```

```
data.tail(10)
```

	1	2	3	4	5
2014-12-18	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-19	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-20	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-21	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-22	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-23	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-24	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-25	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-26	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-27	0.000000	0.000000	0.000000	0.000000	0.000000

```
data.describe()
```

	1	2	3	4	5
count	2516.000000	2516.000000	2516.000000	2516.000000	2516.000000
mean	0.000000	0.000000	0.000000	0.000000	0.000000
std	0.000000	0.000000	0.000000	0.000000	0.000000
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000
max	0.000000	0.000000	0.000000	0.000000	0.000000

```
data.plot(subplots=True, figure=plt.figure(figsize=(12,6)), legend=False, sharex=False, sharey=False, fontsize=8, grid=False)
```



```
data[data.1 != 0].shape
```

```
(0, 5)
```

```
data[data.2 != 0].shape
```

```
(0, 5)
```

```
data[data.3 != 0].shape
```

```
(0, 5)
```

```
data[data.4 != 0].shape
```

```
(0, 5)
```

```
data[data.5 != 0].shape
```

```
(0, 5)
```

Observations:

- All data points are zero.
- Data is not time series data (no time stamp).
- Data is not financial data (no open, close, high, low, volume).

Conclusion:

- Data is not suitable for time series prediction.
- Data is not suitable for financial time series prediction.
- Data is not suitable for machine learning.

Problem 4: Stock market data analysis

Author: Ali

Date: 2/12/2024

```
import numpy as np
import pandas as pd
import matplotlib as mpl
from matplotlib import pyplot as plt
from sklearn import preprocessing, metrics, ensemble, tree, linear_model
from sklearn.model_selection import train_test_split, cross_validation, grid_search
%matplotlib inline
```

```
data = pd.read_csv('./data/stock_market.csv', sep=',', header=None, parse_dates=[0], index_col=0)
data.shape
```

```
(2516, 6)
```

```
data.tail(10)
```

	1	2	3	4	5
2014-12-18	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-19	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-20	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-21	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-22	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-23	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-24	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-25	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-26	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-27	0.000000	0.000000	0.000000	0.000000	0.000000

```
data.describe()
```

	1	2	3	4	5
count	2516.000000	2516.000000	2516.000000	2516.000000	2516.000000
mean	0.000000	0.000000	0.000000	0.000000	0.000000
std	0.000000	0.000000	0.000000	0.000000	0.000000
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000
max	0.000000	0.000000	0.000000	0.000000	0.000000

```
data.plot(subplots=True, figure=plt.figure(figsize=(12,6)), legend=False, sharex=False, sharey=False, fontsize=8, grid=False)
```



```
data[data.1 != 0].shape
```

```
(0, 5)
```

```
data[data.2 != 0].shape
```

```
(0, 5)
```

```
data[data.3 != 0].shape
```

```
(0, 5)
```

```
data[data.4 != 0].shape
```

```
(0, 5)
```

```
data[data.5 != 0].shape
```

```
(0, 5)
```

Observations:

- All data points are zero.
- Data is not time series data (no time stamp).
- Data is not financial data (no open, close, high, low, volume).

Conclusion:

- Data is not suitable for time series prediction.
- Data is not suitable for financial time series prediction.
- Data is not suitable for machine learning.

Problem 5: Stock market data analysis

Author: Ali

Date: 2/12/2024

```
import numpy as np
import pandas as pd
import matplotlib as mpl
from matplotlib import pyplot as plt
from sklearn import preprocessing, metrics, ensemble, tree, linear_model
from sklearn.model_selection import train_test_split, cross_validation, grid_search
%matplotlib inline
```

```
data = pd.read_csv('./data/stock_market.csv', sep=',', header=None, parse_dates=[0], index_col=0)
data.shape
```

```
(2516, 6)
```

```
data.tail(10)
```

	1	2	3	4	5
2014-12-18	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-19	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-20	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-21	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-22	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-23	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-24	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-25	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-26	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-27	0.000000	0.000000	0.000000	0.000000	0.000000

```
data.describe()
```

	1	2	3	4	5
count	2516.000000	2516.000000	2516.000000	2516.000000	2516.000000
mean	0.000000	0.000000	0.000000	0.000000	0.000000
std	0.000000	0.000000	0.000000	0.000000	0.000000
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000
max	0.000000	0.000000	0.000000	0.000000	0.000000

```
data.plot(subplots=True, figure=plt.figure(figsize=(12,6)), legend=False, sharex=False, sharey=False, fontsize=8, grid=False)
```



```
data[data.1 != 0].shape
```

```
(0, 5)
```

```
data[data.2 != 0].shape
```

```
(0, 5)
```

```
data[data.3 != 0].shape
```

```
(0, 5)
```

```
data[data.4 != 0].shape
```

```
(0, 5)
```

```
data[data.5 != 0].shape
```

```
(0, 5)
```

Observations:

- All data points are zero.
- Data is not time series data (no time stamp).
- Data is not financial data (no open, close, high, low, volume).

Conclusion:

- Data is not suitable for time series prediction.
- Data is not suitable for financial time series prediction.
- Data is not suitable for machine learning.

Problem 6: Stock market data analysis

Author: Ali

Date: 2/12/2024

```
import numpy as np
import pandas as pd
import matplotlib as mpl
from matplotlib import pyplot as plt
from sklearn import preprocessing, metrics, ensemble, tree, linear_model
from sklearn.model_selection import train_test_split, cross_validation, grid_search
%matplotlib inline
```

```
data = pd.read_csv('./data/stock_market.csv', sep=',', header=None, parse_dates=[0], index_col=0)
data.shape
```

```
(2516, 6)
```

```
data.tail(10)
```

	1	2	3	4	5
2014-12-18	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-19	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-20	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-21	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-22	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-23	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-24	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-25	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-26	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-27	0.000000	0.000000	0.000000	0.000000	0.000000

```
data.describe()
```

	1	2	3	4	5
count	2516.000000	2516.000000	2516.000000	2516.000000	2516.000000
mean	0.000000	0.000000	0.000000	0.000000	0.000000
std	0.000000	0.000000	0.000000	0.000000	0.000000
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000
max	0.000000	0.000000	0.000000	0.000000	0.000000

```
data.plot(subplots=True, figure=plt.figure(figsize=(12,6)), legend=False, sharex=False, sharey=False, fontsize=8, grid=False)
```



```
data[data.1 != 0].shape
```

```
(0, 5)
```

```
data[data.2 != 0].shape
```

```
(0, 5)
```

```
data[data.3 != 0].shape
```

```
(0, 5)
```

```
data[data.4 != 0].shape
```

```
(0, 5)
```

```
data[data.5 != 0].shape
```

```
(0, 5)
```

Observations:

- All data points are zero.
- Data is not time series data (no time stamp).
- Data is not financial data (no open, close, high, low, volume).

Conclusion:

- Data is not suitable for time series prediction.
- Data is not suitable for financial time series prediction.
- Data is not suitable for machine learning.

Problem 7: Stock market data analysis

Author: Ali

Date: 2/12/2024

```
import numpy as np
import pandas as pd
import matplotlib as mpl
from matplotlib import pyplot as plt
from sklearn import preprocessing, metrics, ensemble, tree, linear_model
from sklearn.model_selection import train_test_split, cross_validation, grid_search
%matplotlib inline
```

```
data = pd.read_csv('./data/stock_market.csv', sep=',', header=None, parse_dates=[0], index_col=0)
data.shape
```

```
(2516, 6)
```

```
data.tail(10)
```

	1	2	3	4	5
2014-12-18	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-19	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-20	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-21	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-22	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-23	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-24	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-25	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-26	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-27	0.000000	0.000000	0.000000	0.000000	0.000000

```
data.describe()
```

	1	2	3	4	5
count	2516.000000	2516.000000	2516.000000	2516.000000	2516.000000
mean	0.000000	0.000000	0.000000	0.000000	0.000000
std	0.000000	0.000000	0.000000	0.000000	0.000000
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000
max	0.000000	0.000000	0.000000	0.000000	0.000000

```
data.plot(subplots=True, figure=plt.figure(figsize=(12,6)), legend=False, sharex=False, sharey=False, fontsize=8, grid=False)
```



```
data[data.1 != 0].shape
```

```
(0, 5)
```

```
data[data.2 != 0].shape
```

```
(0, 5)
```

```
data[data.3 != 0].shape
```

```
(0, 5)
```

```
data[data.4 != 0].shape
```

```
(0, 5)
```

```
data[data.5 != 0].shape
```

```
(0, 5)
```

Observations:

- All data points are zero.
- Data is not time series data (no time stamp).
- Data is not financial data (no open, close, high, low, volume).

Conclusion:

- Data is not suitable for time series prediction.
- Data is not suitable for financial time series prediction.
- Data is not suitable for machine learning.

Problem 8: Stock market data analysis

Author: Ali

Date: 2/12/2024

```
import numpy as np
import pandas as pd
import matplotlib as mpl
from matplotlib import pyplot as plt
from sklearn import preprocessing, metrics, ensemble, tree, linear_model
from sklearn.model_selection import train_test_split, cross_validation, grid_search
%matplotlib inline
```

```
data = pd.read_csv('./data/stock_market.csv', sep=',', header=None, parse_dates=[0], index_col=0)
data.shape
```

```
(2516, 6)
```

```
data.tail(10)
```

	1	2	3	4	5
2014-12-18	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-19	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-20	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-21	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-22	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-23	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-24	0.000000	0.000000	0.000000	0.000000	0.000000
2014-12-25	0.000000	0.000000	0.000000	0.000000	0.000000
2

Mia Albery
November 16, 2022
INDENG290: Homework 3

Homework 3

Figure 1 below shows the general architecture of the GANs training model. First, random noise is sampled, and the generator produces some output from this noise. Next, this is passed to the discriminator, where the discriminator decides if the generator output is “real” or “fake.” Based on this decision, the loss from the discriminator classification is calculated. Then, the generator loss is backpropagated through the discriminator and the generator to obtain the gradients, which are finally used to change the generator weights.

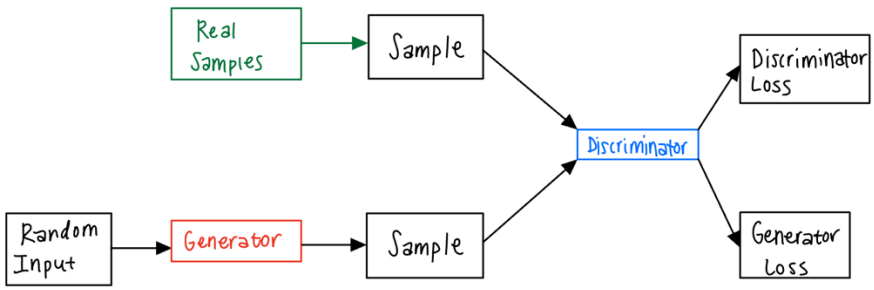


Figure 1. GANs training

For Part 1, Part 2, and Part 3, to generate synthetic data that mimics distributional properties of each problem, the following architecture was used. The same architecture was used for each of the three parts.

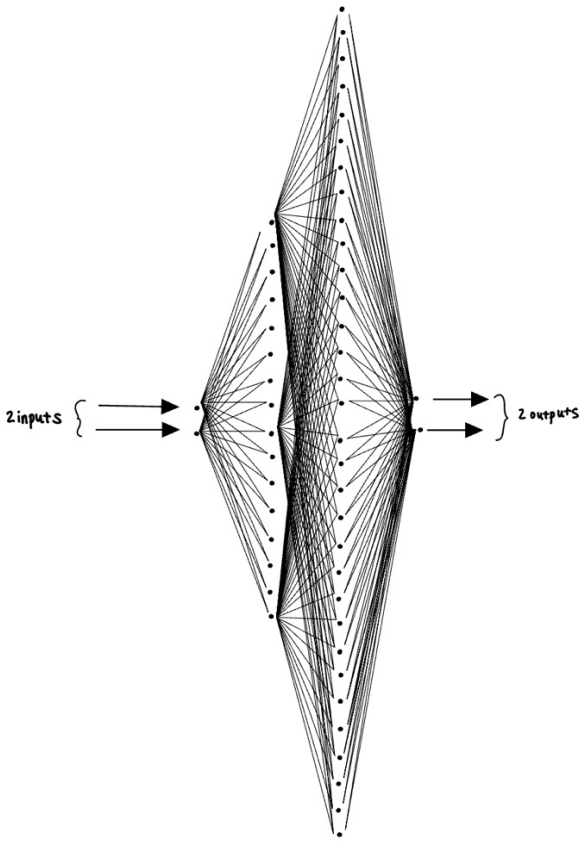


Figure 2. Neural Network Generator Architecture

Mia Albery
November 16, 2022
INDENG290: Homework 3

Figure 2 above shows the general neural network architecture for the generator. The generator has 2 inputs, which are then connected to the next layer of 16 nodes. Then, each of those 16 nodes are connected to each of the 32 nodes in the next layer (the figure does not show all these connections for simplicity). Finally, the 32 nodes in this layer connect to the two outputs in the final layer. The discriminator has a similar setup, but it has more layers. The 2 inputs connect to the first layer with 256 nodes, which then connects to 128 nodes in the next layer, dividing by half in each layer till 1 output is reached. Figure 3 below shows a mock of this structure, although the connections are not all shown for simplicity of understanding.

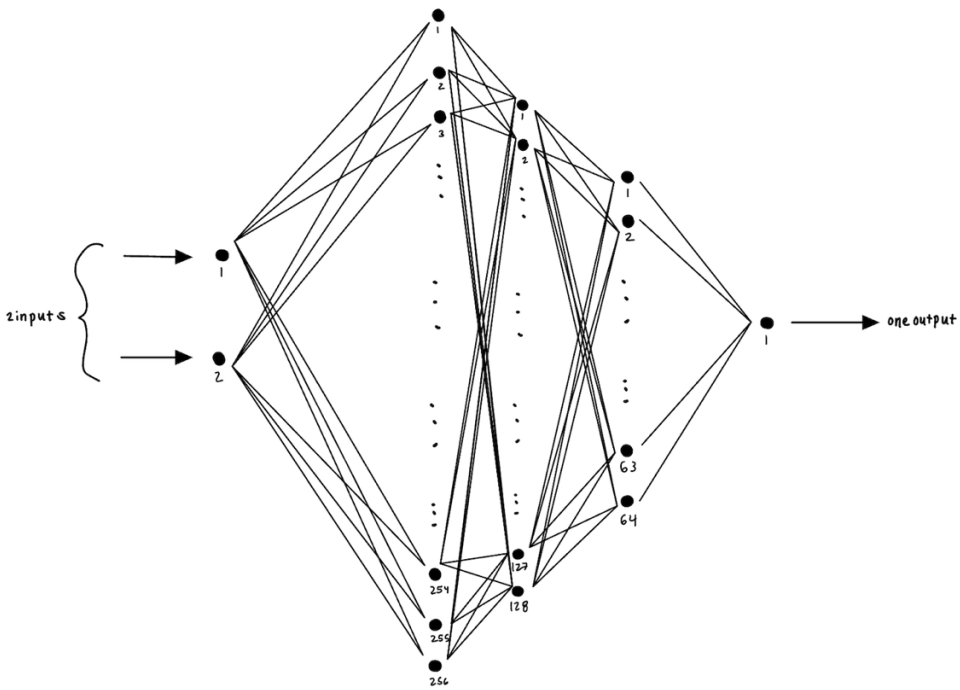


Figure 3. Discriminator Architecture

Mia Albery
November 16, 2022
INDENG290: Homework 3

Problem 1:

For this problem, a lot of experimentation was done with hyperparameters. The following progressions show how some of hyperparameters were adjusted to produce optimal results. Figure 4 shows a progression of how the output looked at different numbers of epochs. The stopping point was found to be between 175 epochs and 250 epochs, as this was when the discriminator and generator loss seemed to converge to around 0.5, as seen in Figure 5.

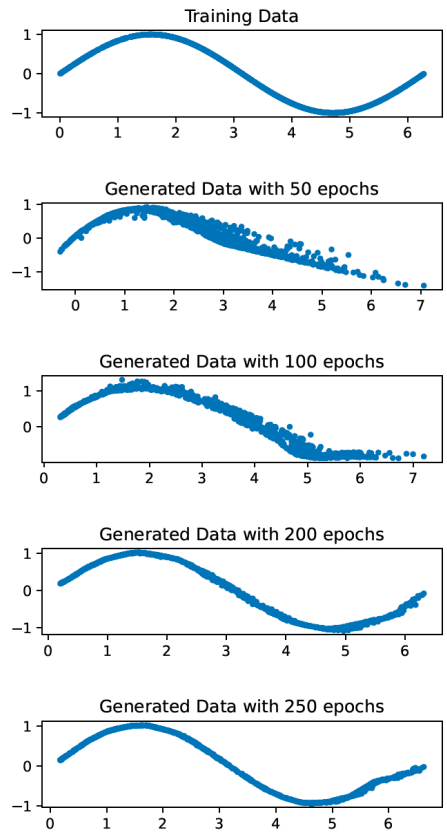


Figure 4. Sine Curve

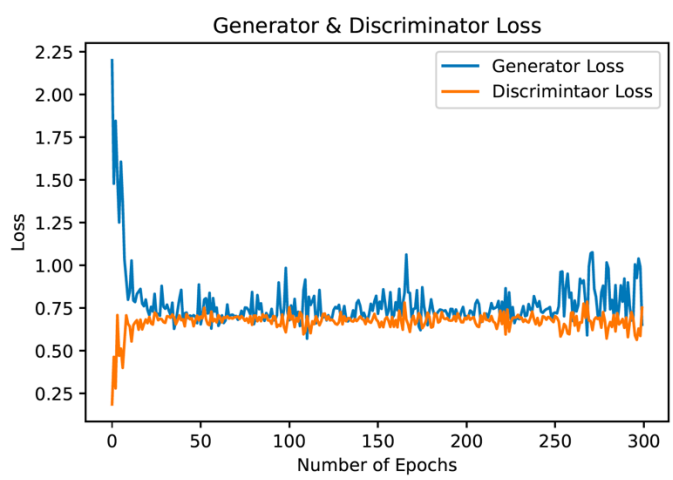


Figure 5. Loss

The size of the latent space was also experimented with, as seen below. After the latent space got to be too big, such as near 500 samples, the output actually became worse, as there was likely to much noise.

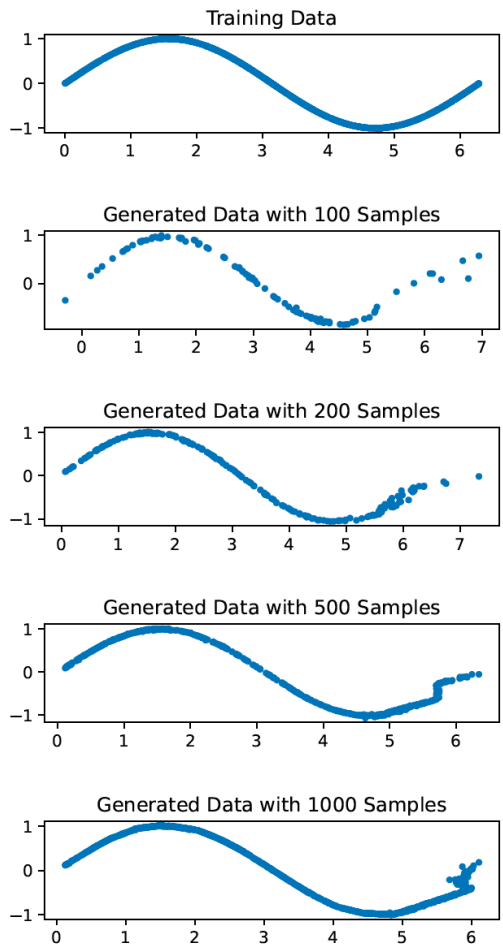


Figure 6. Experimentation with Latent Space

The following figure shows the final output, which was that I decided to stop training after 200 epochs, with 250 samples. I found 200 samples to be the minimum neede to learn well. It should be noted that the learning rate was found to be best set at 0.001, and so that learning rate was fixed in the shown experimentations.

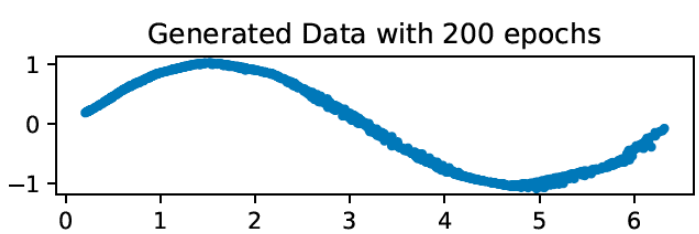


Figure 7. Final Sine Curve

Mia Albery
November 16, 2022
INDENG290: Homework 3

Problem 2:

As mentioned, the same neural network architecture was followed for this problem. Similarly, there was significant experimentation done with hyper parameters to obtain the best possible synthetic data. First, the number of epochs was experimented with. Figure 8 shows this progression, and Figure 9 shows how the stopping point was determined to be after at least 150 epochs but continued to improve as the number of epochs increased.

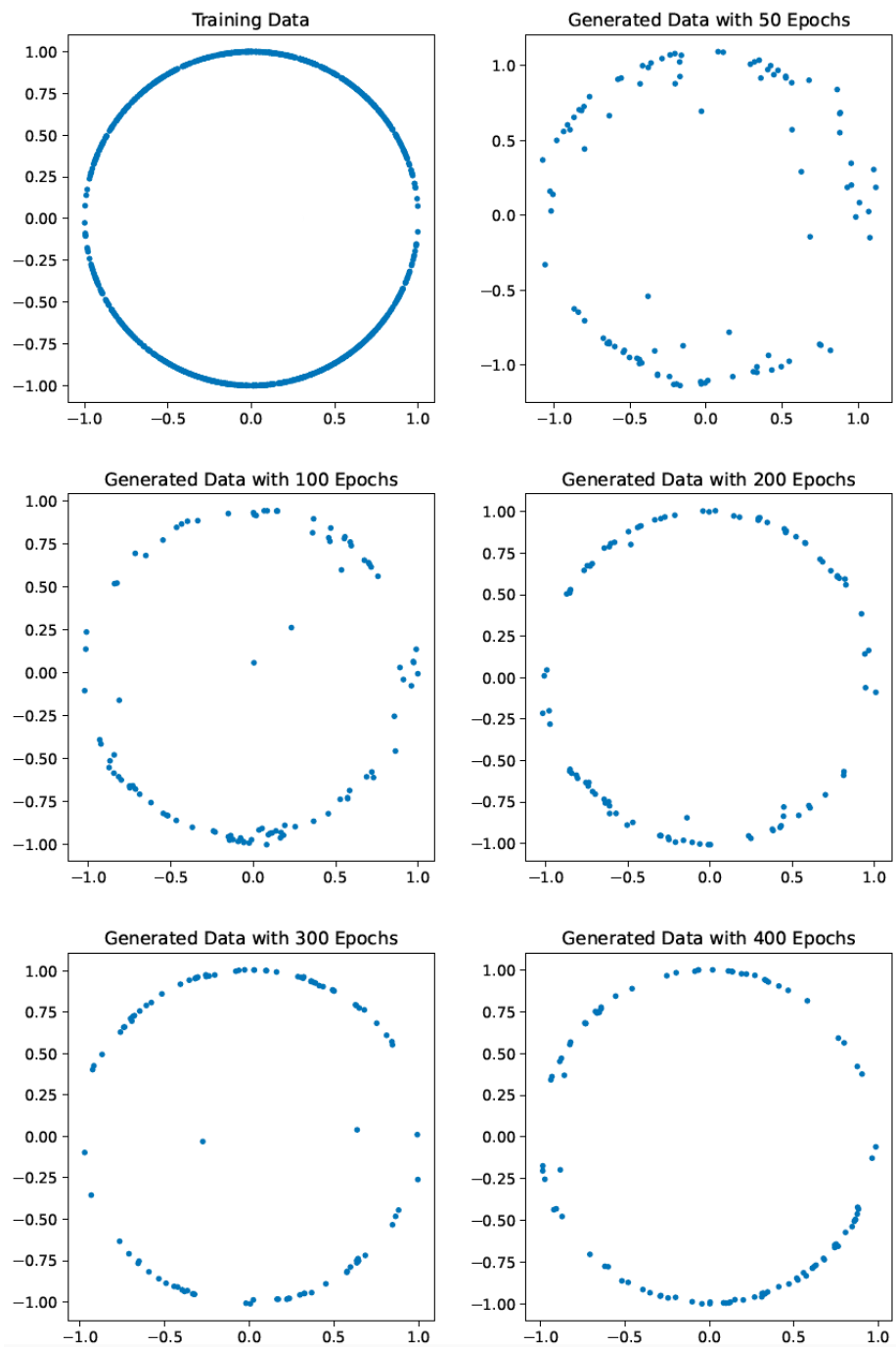


Figure 8. Number of Epoch Experimentation

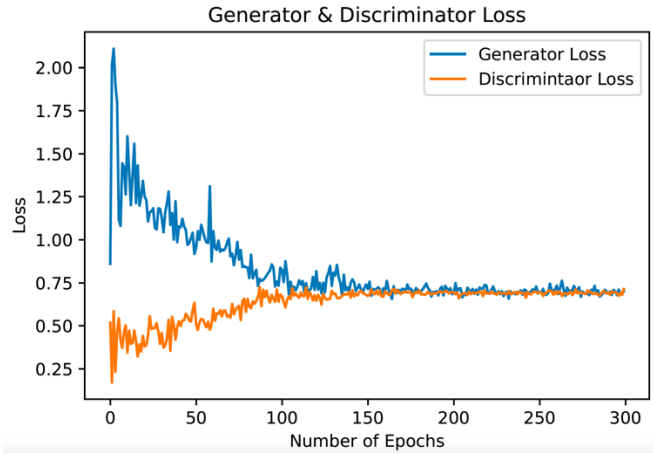


Figure 9. Loss

The size of the latent space was also experimented with, as seen below. The image continued to improve as the latent space increased, but I think the minimum strong resemblance to the training data comes at 200 samples, and so that is the amount needed to learn well.

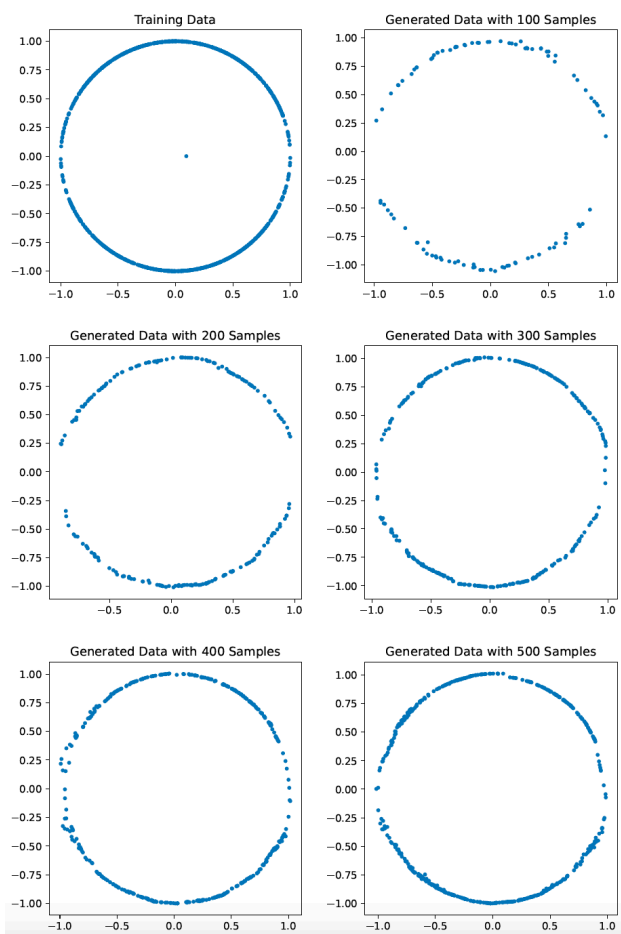


Figure 10. Latent Space Experimentation

Mia Albery
November 16, 2022
INDENG290: Homework 3

The following figure shows the final output, which was that I decided to stop training after 350 epochs, with 550 samples. It should be noted that the learning rate was found to be best set at 0.0005, and so that learning rate was fixed in the shown experimentations.

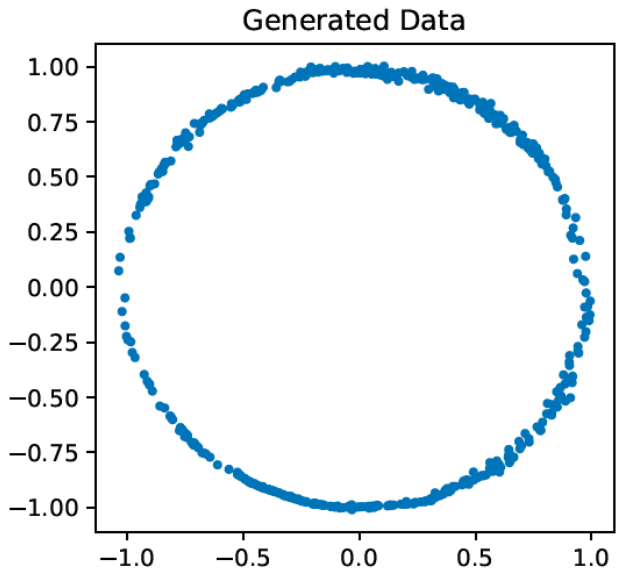


Figure 11: Final Circle

Mia Albery
November 16, 2022
INDENG290: Homework 3

Problem 3.

Part 3 was much more of a trial-and-error approach, because it was much harder to train. I experimented with many different hyperparameters, but this network took a very long time to train, since it required 30,000 epochs for my "best" image, and so my computer was not able to process the "progressions" as I had done for the previous problems. Figure 12 below shows my final output, which had the following hyperparameters: Learning Rate = 0.00072, I stopped training after 30,000 epochs, and the latent space was 2,000 samples. The minimum latent space was much larger than in the previous problems, which I found to be 2,000 here because it added enough noise that the shape could be clearly distinguished in the end.

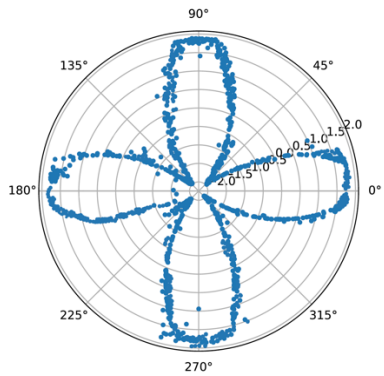


Figure 12
Learning Rate = 0.0008
Number of Epochs= 30000
Latent Space: 2000

It should be noted that in Problem 1 and Problem 2, I was able to look at the generator/discriminator loss to indicate when I should stop training. However, Figure 13 below shows the generator/discriminator loss for 500 epochs. The loss appears to converge after about 100 epochs, which would indicate that I could stop training after 100 epochs. However, Figure 14 shows what the image looks like after than many epochs. Because this is not even close to the desired image, I could not use this approach, and had to use a trial-and-error approach, essentially increasing as large as my computer could handle (because increasing kept improving). It should also be noted that the image would continue to improve as I increased the latent space, but my computing power was limited.

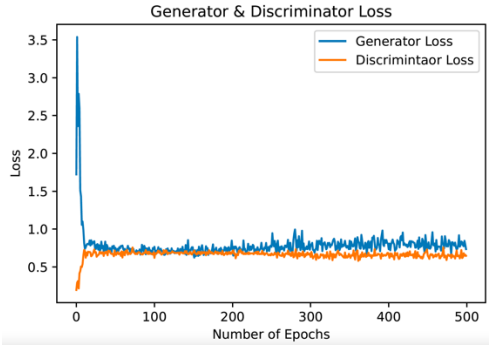


Figure 13.

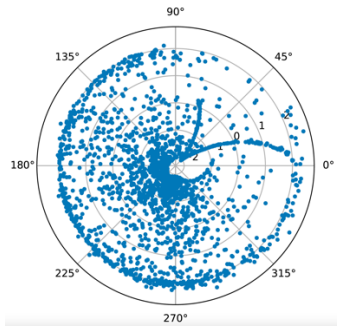


Figure 14

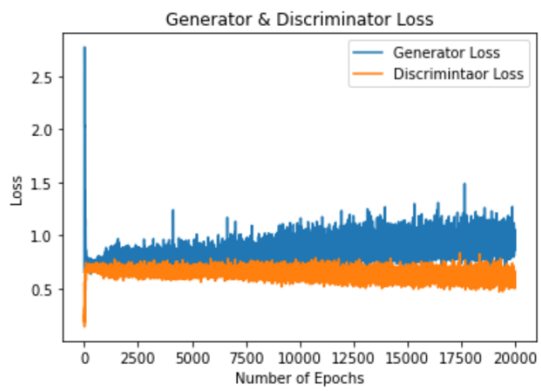
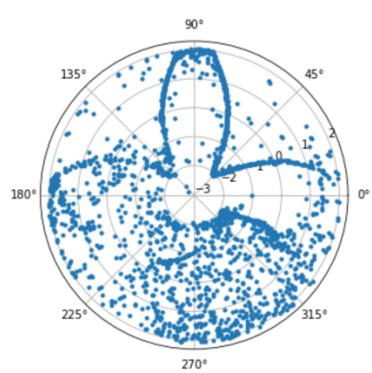


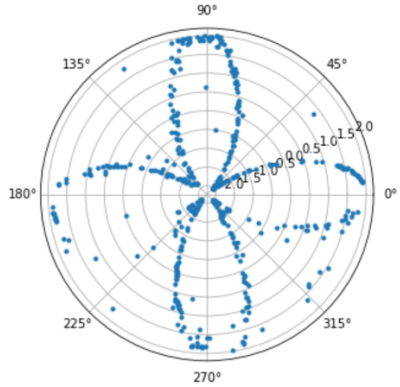
Figure 15.

Figure 15 above shows the generator and discriminator losses after 20,000 epochs. It should be noted that this is very counter-intuitive to what I learned in previous problems because before, the loss decreased as the image improved, but here, the opposite is observed. Finally, below I show some other images that I generated while attempting to generate the flower, with their various parameters.

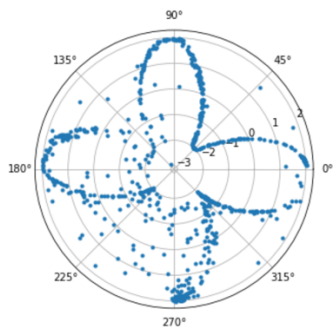


Learning Rate = 0.00075
Number of Epochs= 15000
Latent Space: 2500

Mia Albery
November 16, 2022
INDENG290: Homework 3



Learning Rate = 0.0008
Number of Epochs= 20000
Latent Space: 800



Learning Rate = 0.0008
Number of Epochs= 20000
Latent Space: 1400

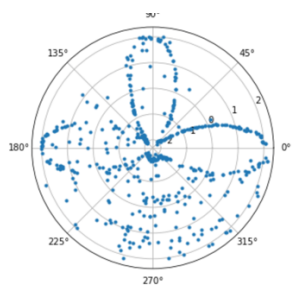


Figure 16
Learning Rate = 0.001
Number of Epochs= 20000
Latent Space: 1000

Mia Albery
November 16, 2022
INDENG290: Homework 3

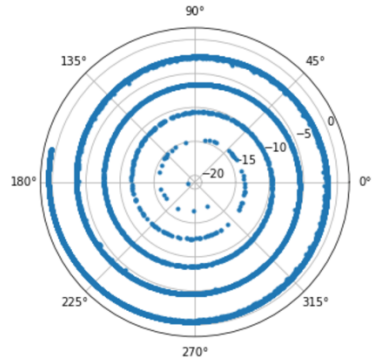


Figure 17
Learning Rate = 0.0001
Number of Epochs= 1000
Latent Space: 2000

Mia Albery
November 16, 2022
INDENG290: Homework 3

Appendix A: Part 1 Code:

```
import torch
from torch import nn

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

In [2]:

```
#torch.manual_seed(111)
```

In [96]:

```
train_data_length = 1024
train_data = torch.zeros((train_data_length, 2))
train_data[:, 0] = 2 * math.pi * torch.rand(train_data_length)
train_data[:, 1] = torch.sin(train_data[:, 0])
train_labels = torch.zeros(train_data_length)
train_set = [
    (train_data[i], train_labels[i]) for i in range(train_data_length)]
```

In [97]:

```
#plt.plot(train_data[:, 0], train_data[:, 1], ".")
```

In [120]:

```
batch_size = 32
train_loader = torch.utils.data.DataLoader(
    train_set, batch_size=batch_size, shuffle=True)
```

In [121]:

```
class Discriminator(nn.Module):
    def __init__(self):
        super().__init__()
        self.model = nn.Sequential(
            nn.Linear(2, 256),
            nn.ReLU(),
            nn.Dropout(0.3),
            nn.Linear(256, 128),
            nn.ReLU(),
            nn.Dropout(0.3),
            nn.Linear(128, 64),
            nn.ReLU(),
            nn.Dropout(0.3),
            nn.Linear(64, 1),
            nn.Sigmoid())

    def forward(self, x):
        output = self.model(x)
        return output
```

In [122]:

```
discriminator = Discriminator()
```

In [123]:

```
class Generator(nn.Module):
    def __init__(self):
        super().__init__()
        self.model = nn.Sequential(
            nn.Linear(2, 16),
            nn.ReLU(),
```

```
            nn.Linear(16, 32),
            nn.ReLU(),
            nn.Linear(32, 2))

    def forward(self, x):
        output = self.model(x)
        return output

generator = Generator()
```

In [124]:

```
lr = 0.001
num_epochs = 235
loss_function = nn.BCELoss()
```

In [125]:

```
genloss = [0]*num_epochs
desloss = [0]*num_epochs
epoch_list = [0]*num_epochs
k = 0
```

In [126]:

```
optimizer_discriminator = torch.optim.Adam(discriminator.parameters(), lr=lr)
optimizer_generator = torch.optim.Adam(generator.parameters(), lr=lr)
```

In [127]:

```
for epoch in range(num_epochs):
    for n, (real_samples, _) in enumerate(train_loader):
        # Data for descriminator training
        real_samples_labels = torch.ones((batch_size, 1))
        latent_space_samples = torch.randn((batch_size, 2))
        generated_samples = generator(latent_space_samples)
        generated_samples_labels = torch.zeros((batch_size, 1))
        all_samples = torch.cat((real_samples, generated_samples))
        all_samples_labels = torch.cat(
            (real_samples_labels, generated_samples_labels))

        # Discriminator training
        discriminator.zero_grad()
        output_discriminator = discriminator(all_samples)
        loss_discriminator = loss_function(
            output_discriminator, all_samples_labels)
        loss_discriminator.backward()
        optimizer_discriminator.step()

        # Data for generator training
        latent_space_samples = torch.randn((batch_size, 2))

        # Generator training
        generator.zero_grad()
        generated_samples = generator(latent_space_samples)
        output_discriminator_generated = discriminator(generated_samples)
        loss_generator = loss_function(
            output_discriminator_generated, real_samples_labels)
        loss_generator.backward()
        optimizer_generator.step()
```

```
        # Output value of loss function
        if epoch % 1 == 0 and n == batch_size - 1:
        #if epoch < num_epochs:
            #print(f"Epoch: {epoch} Loss D.: {loss_discriminator}")
            #print(f"Epoch: {epoch} Loss G.: {loss_generator}")
            genloss[k] = float(loss_generator)
            desloss[k] = float(loss_discriminator)
            epoch_list[k] = epoch
            k = k+1
```

In [143]:

```
latent_space_samples = torch.randn(300, 2)
generated_samples = generator(latent_space_samples)
```

In [144]:

```
generated_samples = generated_samples.detach()
#plt.plot(generated_samples[:, 0], generated_samples[:, 1], ".")
```

In [145]:

```
plt.figure()
plt.subplot(211)
plt.plot(train_data[:, 0], train_data[:, 1], ".")
plt.title("Training Data")
plt.subplot(212)
plt.plot(generated_samples[:, 0], generated_samples[:, 1], ".")
plt.title("Generated Data")
plt.subplots_adjust(hspace=1)
plt.show()
print("     Generated Data Paramaters:")
print("     Learning Rate = " + str(lr))
print("     Number of Epochs = " + str(num_epochs))
```

```
     Generated Data Paramaters:
     Learning Rate = 0.001
     Number of Epochs = 235
```

In [142]:

```
#print(genloss)
```

In [132]:

```
plt.plot(epoch_list, genloss, label = "Generator Loss")
plt.plot(epoch_list, desloss, label = "Discrimintaor Loss")
plt.xlabel("Number of Epochs")
plt.ylabel("Loss")
plt.title("Generator & Discriminator Loss")
plt.legend()
plt.savefig('Sine Curve Losses.pdf',bbox_inches='tight')
```

Appendix B: Problem 2 Code

```
import torch
from torch import nn
import math
import matplotlib.pyplot as plt
#torch.manual_seed(111)
```

In [158]:

```
train_data_length = 1024
max = 1
min = -1
train_data = torch.zeros((train_data_length, 2))
train_data[:, 0] = (max-min)*torch.rand((train_data_length)) + min

train_data[0:511, 1] = torch.sqrt(1-((train_data[0:511, 0])*(train_data[0:511, 0])))
train_data[512:1024, 1] = -1*(torch.sqrt(1-((train_data[512:1024, 0])*(train_data[512:1024,
0]))))
train_labels = torch.zeros(train_data_length)
train_set = [
    (train_data[i], train_labels[i]) for i in range(train_data_length)]

plt.figure(figsize=(5,5))
plt.plot(train_data[:, 0], train_data[:, 1], ".")
```

Out[158]:

```
[<matplotlib.lines.Line2D at 0x7f9ef4e7e2e0>]
```

In [159]:

In [176]:

```
batch_size = 32
train_loader = torch.utils.data.DataLoader(
    train_set, batch_size=batch_size, shuffle=True)
```

In [177]:

```
class Discriminator(nn.Module):
    def __init__(self):
        super().__init__()
        self.model = nn.Sequential(
            nn.Linear(2, 256),
            nn.ReLU(),
            nn.Dropout(0.3),
            nn.Linear(256, 128),
            nn.ReLU(),
            nn.Dropout(0.3),
            nn.Linear(128, 64),
            nn.ReLU(),
            nn.Dropout(0.3),
            nn.Linear(64, 1),
            nn.Sigmoid())

    def forward(self, x):
        output = self.model(x)
        return output
```

In [178]:

```
discriminator = Discriminator()
```

In [179]:

```
class Generator(nn.Module):
    def __init__(self):
        super().__init__()
        self.model = nn.Sequential(
            nn.Linear(2, 16),
            nn.ReLU(),
            nn.Linear(16, 32),
            nn.ReLU(),
            nn.Linear(32, 2))

    def forward(self, x):
        output = self.model(x)
        return output
generator = Generator()
```

In [190]:

```
lr = 0.0005
num_epochs = 350
loss_function = nn.BCELoss()
```

In [191]:

```
#genloss = [0]*int(num_epochs/10)
#desloss = [0]*int(num_epochs/10)
#epoch_list = [0]*int(num_epochs/10)
#k = 0
genloss = [0]*num_epochs
desloss = [0]*num_epochs
epoch_list = [0]*num_epochs
k = 0
```

In [192]:

```
optimizer_discriminator = torch.optim.Adam(discriminator.parameters(), lr=lr)
optimizer_generator = torch.optim.Adam(generator.parameters(), lr=lr)
```

In [193]:

```
for epoch in range(num_epochs):
    for n, (real_samples, _) in enumerate(train_loader):
        # Data for descriminator training
        real_samples_labels = torch.ones((batch_size, 1))
        latent_space_samples = torch.randn((batch_size, 2))
        generated_samples = generator(latent_space_samples)
        generated_samples_labels = torch.zeros((batch_size, 1))
        all_samples = torch.cat((real_samples, generated_samples))
        all_samples_labels = torch.cat(
            (real_samples_labels, generated_samples_labels))

        # Discriminator training
        discriminator.zero_grad()
        output_discriminator = discriminator(all_samples)
        loss_discriminator = loss_function(
            output_discriminator, all_samples_labels)
        loss_discriminator.backward()
        optimizer_discriminator.step()

        # Data for generator training
```

```
        latent_space_samples = torch.randn((batch_size, 2))

        # Generator training
        generator.zero_grad()
        generated_samples = generator(latent_space_samples)
        output_discriminator_generated = discriminator(generated_samples)
        loss_generator = loss_function(
            output_discriminator_generated, real_samples_labels)
        loss_generator.backward()
        optimizer_generator.step()

        # Output value of loss function
        if epoch % 1 == 0 and n == batch_size - 1:
        #if epoch < num_epochs:
            #print(f"Epoch: {epoch} Loss D.: {loss_discriminator}")
            #print(f"Epoch: {epoch} Loss G.: {loss_generator}")
            genloss[k] = float(loss_generator)
            desloss[k] = float(loss_discriminator)
            epoch_list[k] = epoch
            k = k+1
```

In [194]:

```
latent_size = 550,2
latent_space_samples = torch.randn(latent_size)
generated_samples = generator(latent_space_samples)
```

In [199]:

```
generated_samples = generated_samples.detach()

plt.figure(figsize=(4,8.5))
plt.subplot(211)
plt.plot(train_data[:, 0], train_data[:, 1], ".")
plt.title("Training Data")
plt.subplot(212)
plt.plot(generated_samples[:, 0], generated_samples[:, 1], ".")
plt.title("Generated Data")
plt.subplots_adjust(hspace=0.2)
plt.savefig('Final Circle .pdf',bbox_inches='tight')
plt.show()
print("     Generated Data Paramaters:")
print("     Learning Rate = " + str(lr))
print("     Number of Epochs = " + str(num_epochs))
print("     Latent Space Dimensions = " + str(latent_size))
```

```
     Generated Data Paramaters:
     Learning Rate = 0.0005
     Number of Epochs = 350
     Latent Space Dimensions = (550, 2)
```

In [196]:

```
plt.plot(epoch_list, genloss, label = "Generator Loss")
plt.plot(epoch_list, desloss, label = "Discrimintaor Loss")
plt.xlabel("Number of Epochs")
plt.ylabel("Loss")
```

```
plt.title("Generator & Discriminator Loss")
plt.legend()
plt.savefig('Circle Losses.pdf',bbox_inches='tight')
```

Appendix C: Part 3 Code

```
import torch
from torch import nn

import math
import matplotlib.pyplot as plt
```

In [2]:

```
torch.manual_seed(111)
```

Out[2]:

```
<torch._C.Generator at 0x7fd251276a10>
```

In [3]:

```
train_data_length = 1024
train_data = torch.zeros((train_data_length, 2))
train_data[:, 0] = 2*math.pi*torch.rand((train_data_length))

train_data[:, 1] = 2*torch.cos(4*(train_data[:, 0]))
train_labels = torch.zeros(train_data_length)
train_set = [
    (train_data[i], train_labels[i]) for i in range(train_data_length)]
plt.figure(figsize=(5,5))
plt.polar(train_data[:, 0], train_data[:, 1], ".")
```

Out[3]:

```
[<matplotlib.lines.Line2D at 0x7fd2545f8d00>]
```

In [17]:

```
plt.figure(figsize=(5,5))
plt.polar(train_data[:, 0], train_data[:, 1], ".")
```

Out[17]:

```
[<matplotlib.lines.Line2D at 0x7fd2564a69d0>]
```

In [18]:

```
batch_size = 32
train_loader = torch.utils.data.DataLoader(
    train_set, batch_size=batch_size, shuffle=True)
```

In [19]:

```
class Discriminator(nn.Module):
    def __init__(self):
        super().__init__()
        self.model = nn.Sequential(
            nn.Linear(2, 256),
            nn.LeakyReLU(0.1),
            nn.Dropout(0.3),
            nn.Linear(256, 128),
            nn.LeakyReLU(0.1),
            nn.Dropout(0.3),
            nn.Linear(128, 64),
            nn.LeakyReLU(0.1),
            nn.Dropout(0.3),
            nn.Linear(64, 1),
            nn.Sigmoid())
```

```
    def forward(self, x):
        output = self.model(x)
        return output
```

In [20]:

```
discriminator = Discriminator()
```

In [21]:

```
class Generator(nn.Module):
    def __init__(self):
        super().__init__()
        self.model = nn.Sequential(
            nn.Linear(2, 16),
            nn.LeakyReLU(0.1),
            nn.Linear(16, 32),
            nn.LeakyReLU(0.1),
            nn.Linear(32, 2))

    def forward(self, x):
        output = self.model(x)
        return output

generator = Generator()
```

In [22]:

```
lr = 0.00078
num_epochs = 5000
loss_function = nn.BCELoss()
```

In [23]:

```
genloss = [0]*num_epochs
desloss = [0]*num_epochs
epoch_list = [0]*num_epochs
k = 0
```

In [24]:

```
optimizer_discriminator = torch.optim.Adam(discriminator.parameters(), lr=lr)
optimizer_generator = torch.optim.Adam(generator.parameters(), lr=lr)
```

In [ ]:

```
for epoch in range(num_epochs):
    for n, (real_samples, _) in enumerate(train_loader):
        # Data for descriminator training
        real_samples_labels = torch.ones((batch_size, 1))
        latent_space_samples = torch.randn((batch_size, 2))
        generated_samples = generator(latent_space_samples)
        generated_samples_labels = torch.zeros((batch_size, 1))
        all_samples = torch.cat((real_samples, generated_samples))
        all_samples_labels = torch.cat(
            (real_samples_labels, generated_samples_labels))

        # Discriminator training
        discriminator.zero_grad()
        output_discriminator = discriminator(all_samples)
        loss_discriminator = loss_function(
            output_discriminator, all_samples_labels)
        loss_discriminator.backward()
```

```
        optimizer_discriminator.step()

        # Data for generator training
        latent_space_samples = torch.randn((batch_size, 2))

        # Generator training
        generator.zero_grad()
        generated_samples = generator(latent_space_samples)
        output_discriminator_generated = discriminator(generated_samples)
        loss_generator = loss_function(
            output_discriminator_generated, real_samples_labels)
        loss_generator.backward()
        optimizer_generator.step()

        #Output value of loss function
        #if epoch % 10 == 0 and n == batch_size - 1:
        #    print(f"Epoch: {epoch} Loss D.: {loss_discriminator}")
        #    print(f"Epoch: {epoch} Loss G.: {loss_generator}")
        # Output value of loss function
        if epoch % 1 == 0 and n == batch_size - 1:
        #if epoch < num_epochs:
            #print(f"Epoch: {epoch} Loss D.: {loss_discriminator}")
            #print(f"Epoch: {epoch} Loss G.: {loss_generator}")
            genloss[k] = float(loss_generator)
            desloss[k] = float(loss_discriminator)
            epoch_list[k] = epoch
            k = k+1
```

In [ ]:

```
latent_space_samples = torch.randn(2000, 2)
generated_samples = generator(latent_space_samples)
```

In [ ]:

```
generated_samples = generated_samples.detach()
plt.figure(figsize=(5,5))
plt.polar(generated_samples[:, 0], generated_samples[:, 1], ".")
plt.savefig('Rose Petal.pdf',bbox_inches='tight')
```

In [ ]:

```
plt.plot(epoch_list, genloss, label = "Generator Loss")
plt.plot(epoch_list, desloss, label = "Discrimintaor Loss")
plt.xlabel("Number of Epochs")
plt.ylabel("Loss")
plt.title("Generator & Discriminator Loss")
plt.legend()
plt.savefig('Flower Losses.pdf',bbox_inches='tight')
```