Lecture 9

Matplotlib – Part 2

Data Visualization Using Python

Prepared by: Assistant Lecturer Hadi Salah
Department of Artificial Intelligence

1. Recap of Part 1

In Part 1, we covered:

- Basic concepts of data visualization.
- Simple line, scatter, bar, and histogram plots.
- Titles, axis labels, grid, legend.
- Saving plots as image files.

Example (simple line plot recap):

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

x = np.linspace(0, 5, 50)
y = x ** 2

plt.plot(x, y, label="y = x^2")
plt.xlabel("x")
plt.ylabel("y")
plt.title("Recap: Simple Line Plot")
plt.grid(True)
plt.legend()
plt.show()
```

2. Figure and Axes Concepts

Matplotlib is organized around two main objects:

- **Figure**: The entire window or page where everything is drawn.
- **Axes**: The actual area where the data is plotted (a subplot).

Example: using the object-oriented (OO) interface.

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

x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)

fig, ax = plt.subplots()  # fig: Figure, ax: Axes
ax.plot(x, y, label="sin(x)")
ax.set_title("Sine Function (OO Interface)")
ax.set_xlabel("x (radians)")
ax.set_ylabel("sin(x)")
ax.grid(True)
ax.legend()

plt.show()
```

3. Subplots: Multiple Axes in One Figure

Subplots allow us to show multiple related plots in a single figure.

3.1 Using `plt.subplot`

- Syntax: `plt.subplot(nrows, ncols, index)`.
- Index starts from 1 and goes to $nrows \times ncols$.

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

x = np.linspace(0, 2*np.pi, 200)
y1 = np.sin(x)
y2 = np.cos(x)

plt.figure(figsize=(8, 4))

plt.subplot(1, 2, 1)
plt.plot(x, y1)
plt.title("sin(x)")
plt.grid(True)

plt.subplot(1, 2, 2)
plt.plot(x, y2, color="orange")
plt.title("cos(x)")
plt.grid(True)
```

```
plt.suptitle("Subplot Example Using plt.subplot")
plt.tight_layout()
plt.show()
```

3.2 Using `plt.subplots`

- Returns a Figure and an array of Axes.
- More flexible and recommended in new code.

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

x = np.linspace(0, 2*np.pi, 200)
y1 = np.sin(x)
y2 = np.cos(x)

fig, axes = plt.subplots(2, 1, figsize=(6, 6))

axes[0].plot(x, y1)
axes[0].set_title("sin(x)")
axes[0].grid(True)

axes[1].plot(x, y2, color="orange")
axes[1].set_title("cos(x)")
axes[1].grid(True)

fig.suptitle("Subplot Example Using plt.subplots")
fig.tight_layout()
plt.show()
```

4. Customizing Plots: Style, Color, and Markers

We can control many visual aspects:

- Line color, style, and width.
- Marker type and size.
- Transparency (alpha).

Example:

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

x = np.linspace(0, 10, 50)
```

```
y1 = np.sin(x)
y2 = np.cos(x)

plt.plot(x, y1, color="blue", linestyle="-", marker="o",
         linewidth=2, markersize=5, label="sin(x)")
plt.plot(x, y2, color="red", linestyle="--", marker="s",
         linewidth=2, markersize=5, alpha=0.7, label="cos(x)")

plt.xlabel("x")
plt.ylabel("Value")
plt.title("Customized Line Styles and Markers")
plt.grid(True)
plt.legend()
plt.show()
```

5. Multiple Plots on the Same Axes

We can overlay multiple data series on the same Axes.

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

x = np.linspace(0, 2*np.pi, 200)
y1 = np.sin(x)
y2 = np.sin(2*x)
y3 = np.sin(3*x)

plt.plot(x, y1, label="sin(x)")
plt.plot(x, y2, label="sin(2x)")
plt.plot(x, y3, label="sin(3x)")

plt.xlabel("x (radians)")
plt.ylabel("Amplitude")
plt.title("Multiple Sine Waves")
plt.grid(True)
plt.legend()
plt.show()
```

6. Legends and Annotations

6.1 Legend Position

We can control where the legend appears.

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

```
x = np.linspace(0, 2*np.pi, 200)
y = np.sin(x)

plt.plot(x, y, label="sin(x)")
plt.xlabel("x")
plt.ylabel("sin(x)")
plt.title("Legend Position Example")

plt.legend(loc="upper right")  # or "lower left", "center", etc.
plt.grid(True)
plt.show()
```

6.2 Adding Text and Annotations

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

x = np.linspace(0, 2*np.pi, 200)
y = np.sin(x)

plt.plot(x, y)
plt.xlabel("x")
plt.ylabel("sin(x)")
plt.title("Annotation Example")
plt.grid(True)

# Add text at a specific point
plt.text(np.pi, 0.0, "Zero crossing", ha="center", va="bottom",
         fontsize=10, color="red")

# Add an arrow annotation
plt.annotate("Peak",
             xy=(np.pi/2, 1.0),
             xytext=(2, 1.2),
             arrowprops=dict(facecolor="black", shrink=0.05))

plt.show()
```

7. 3D Plotting

Matplotlib supports simple 3D plots using `mpl_toolkits.mplot3d`.
Example: 3D surface plot.

```
import numpy as np
```

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D  # needed for 3D

x = np.linspace(-3, 3, 50)
y = np.linspace(-3, 3, 50)
X, Y = np.meshgrid(x, y)
Z = np.sin(np.sqrt(X**2 + Y**2))

fig = plt.figure(figsize=(6, 5))
ax = fig.add_subplot(111, projection="3d")

ax.plot_surface(X, Y, Z, cmap="viridis")
ax.set_title("3D Surface Plot")
ax.set_xlabel("X")
ax.set_ylabel("Y")
ax.set_zlabel("Z")

plt.show()
```

8. Time Series Plots

We often plot data indexed by time (e.g., daily temperature, heart rate).
Example using Python `datetime`:

```
import matplotlib.pyplot as plt
import datetime as dt

dates = [
    dt.date(2024, 1, 1),
    dt.date(2024, 1, 2),
    dt.date(2024, 1, 3),
    dt.date(2024, 1, 4),
    dt.date(2024, 1, 5),
]
values = [100, 105, 102, 110, 108]

plt.plot(dates, values, marker="o")
plt.xlabel("Date")
plt.ylabel("Value")
plt.title("Simple Time Series Plot")
plt.grid(True)
plt.gcf().autofmt_xdate()  # Rotate date labels
plt.show()
```

9. Matplotlib with Pandas

Pandas integrates directly with Matplotlib.
Example:

```
import pandas as pd
import matplotlib.pyplot as plt

data = {
    "Day": [1, 2, 3, 4, 5],
    "HeartRate": [72, 75, 78, 74, 76],
    "BloodPressure": [120, 118, 122, 119, 121],
}
df = pd.DataFrame(data)

# Set index to day for prettier plot
df.set_index("Day", inplace=True)

df.plot(kind="line", marker="o")
plt.title("Heart Rate and Blood Pressure Over Days")
plt.xlabel("Day")
plt.ylabel("Value")
plt.grid(True)
plt.show()
```

10. Heatmaps and Confusion Matrix

Heatmaps are useful to display matrices, e.g., confusion matrices in AI.
Example: confusion matrix for a binary classifier.

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

cm = np.array([[50, 10],
               [ 5, 35]])  # rows: true class, columns: predicted

plt.imshow(cm, cmap="Blues")
plt.colorbar(label="Count")

plt.title("Confusion Matrix")
plt.xlabel("Predicted Class")
plt.ylabel("True Class")

# Add text inside each cell
for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        plt.text(j, i, cm[i, j],
```

```
                    ha="center", va="center", color="black")

plt.show()
```

11. Twin Axes (Two Y-Scales)

Sometimes we want to show two different scales on the same x-axis.

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

x = np.arange(1, 11)
temperature = np.array([36.5, 36.7, 36.9, 37.1, 37.0, 36.8, 36.7,
   36.6, 36.5, 36.4])
heart_rate  = np.array([70, 72, 75, 78, 76, 74, 72, 71, 70, 69])

fig, ax1 = plt.subplots()

color1 = "tab:red"
ax1.set_xlabel("Time (unit)")
ax1.set_ylabel("Temperature ( C )", color=color1)
ax1.plot(x, temperature, color=color1, marker="o")
ax1.tick_params(axis="y", labelcolor=color1)

ax2 = ax1.twinx()  # second axes sharing the same x-axis
color2 = "tab:blue"
ax2.set_ylabel("Heart Rate (bpm)", color=color2)
ax2.plot(x, heart_rate, color=color2, marker="s")
ax2.tick_params(axis="y", labelcolor=color2)

plt.title("Twin Axes Example: Temperature and Heart Rate")
fig.tight_layout()
plt.show()
```

12. Saving High-Quality Figures

For reports and publications, we often need high-resolution images.

- Use `dpi` parameter to control resolution.
- Use vector formats (PDF, SVG) when possible.

Example:

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

```
x = np.linspace(0, 2*np.pi, 200)
y = np.sin(x)

plt.plot(x, y)
plt.title("High-Resolution Example")
plt.xlabel("x")
plt.ylabel("sin(x)")

plt.savefig("high_res_plot.png", dpi=300)  # high DPI
plt.savefig("high_res_plot.pdf")           # vector format
plt.show()
```

13. Simple Animation (Concept)

Matplotlib can create simple animations using `FuncAnimation`.
Example (conceptual code):

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

fig, ax = plt.subplots()
x = np.linspace(0, 2*np.pi, 200)
line, = ax.plot(x, np.sin(x))

def update(frame):
    # frame goes from 0, 1, 2, ...
    y = np.sin(x + 0.1 * frame)
    line.set_ydata(y)
    return line,

ani = FuncAnimation(fig, update, frames=100, interval=50, blit=True)

plt.title("Animated Sine Wave")
plt.show()

# To save as GIF (requires additional dependencies):
# ani.save("sine_wave.gif", writer="imagemagick")
```

14. Case Study: AI Model Visualization

In machine learning, we frequently visualize:

- Training and validation loss.

- Training and validation accuracy.
- Confusion matrix of predictions.

Example: training curves using subplots.

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

epochs = np.arange(1, 11)
train_loss = np.exp(-0.3 * epochs) + 0.05*np.random.rand(len(epochs)
   )
val_loss   = np.exp(-0.28 * epochs) + 0.1*np.random.rand(len(epochs)
   )

train_acc = 0.5 + 0.05*epochs + 0.05*np.random.rand(len(epochs))
val_acc   = 0.5 + 0.045*epochs + 0.05*np.random.rand(len(epochs))

fig, axes = plt.subplots(1, 2, figsize=(10, 4))

# Loss subplot
axes[0].plot(epochs, train_loss, marker="o", label="Train Loss")
axes[0].plot(epochs, val_loss, marker="s", label="Val Loss")
axes[0].set_xlabel("Epoch")
axes[0].set_ylabel("Loss")
axes[0].set_title("Training and Validation Loss")
axes[0].grid(True)
axes[0].legend()

# Accuracy subplot
axes[1].plot(epochs, train_acc, marker="o", label="Train Acc")
axes[1].plot(epochs, val_acc, marker="s", label="Val Acc")
axes[1].set_xlabel("Epoch")
axes[1].set_ylabel("Accuracy")
axes[1].set_title("Training and Validation Accuracy")
axes[1].grid(True)
axes[1].legend()

fig.suptitle("AI Model Training Curves")
fig.tight_layout()
plt.show()
```

The confusion matrix visualization from Section 10 is also part of this case study.