สรุปวิชา Machine Learning

1. อธิบายส่วนของโปรเจกต์ (15 คะแนน)

โปรเจกต์กลุ่มเราชื่องานว่า *โปรแกรมเพื่อแปลงลายมือ ให้เป็นตัวอักษร* นำเสนองานเป็น 2 ชิ้นงาน คือ ชิ้นงานที่สามารถเปิดหน้าต่างแล้วใช้เมาส์ในการวาดรูป แล้วแปลงรูปที่วาดส่งให้โมเดลเพื่อให้โมเดลจำแนกว่าตรงกับเลขอะไร อีกชิ้นงานนึงคือการเปิดเป็นกล้องออกมาแล้วเปิดภาพที่วาดด้วยลายมือก่อนแปลงส่งให้โมเดลจำแนกอีกที

โปรเจกต์ใช้ Algorithm รูปแบบ CNN (Convolutional Neural Network) โดย*ใช้ฐานข้อมูล MNIST จาก Keras ซึ่งเป็นฐานข้อมูลรูปภาพตัวเลข*ในการเทรนด์ให้กับโมเดล

ส่วนสำคัญในการเทรนด์โมเดล ส่วนแรกคือการทำ Normalization ให้กับภาพจากฐานข้อมูลให้อยู่ในช่วงที่สามารถนำเข้าไปเทรนด์ได้ เช่น การแปลงภาพให้เป็นขาวดำโดยการแปลงข้อมูลเป็น Float32 แล้วหารด้วย 255 และการ Reshape ให้มีขนาดเท่ากัน จากนั้นจึงสร้าง Layer ขึ้นมาโดยอาศัยโมเดลรูปแบบ Sequential และสร้าง Layers ขึ้นมาทั้งหมด 7 Layers ใช้ relu มาเพื่อสกัดจุดเด่นของรูปภาพเพื่อให้สามารถเทรนด์ได้ดีขึ้น ใช้ Maxpool เพื่อแปลงจุดเด่นเหล่านั้นให้มีขนาด 2*2 แล้วใช้ Flatten เพื่อเปลี่ยนข้อมูลเป็น vactor จากนั้นนำ Dropout มาเพื่อปิดกัน Node เพื่อลดปัญหาการเกิด Overfitting แล้วจึงใช้ Danse เพื่อส่งออกข้อมูล Output

ในขั้นตอนของการเทรนด์โมเดล เลือกใช้ Optimizer เป็นแบบ Adam เพื่อลดอัตราสูญเสีย และเพิ่มความแม่นยำได้ดี ตั้งค่าให้ข้อมูลที่สนใจคือค่า Accuracy และค่า Loss คือค่า Crossentropy ที่เป็นข้อมูลที่ถูกจำแนกผิดฝั่งระหว่างการเทรนด์

เพิ่มเติมคำสั่ง EarlyStopping เข้ามาแก้ไขการเทรนด์ไม่ดี โดยตั้งค่าให้เมื่อการเทรนด์มีค่าเพิ่มขึ้นไม่สูงกว่า 0.01 ติดต่อกัน 4 ครั้งให้หยุดทำงานทันที และใช้ ModelCheckpoint เพื่อบันทึกข้อมูลรอบการเทรนด์ที่มีผลดีที่สุด และกำหนดค่า Epoch อยู่ที่ 100 ครั้งเนื่องจากหากมากขึ้น โมเดลจะเริ่มทำงานได้แย่ลงด้วย

ในส่วนการแสดงผล เราใช้ Pygame ขึ้นมาเพื่อเปิดหน้าต่างที่สามารถใช้เมาส์ในการวาดรูปได้ โดยสร้าง Array ขึ้นมาเพื่อเก็บข้อมูลเส้นที่วาดแล้ว (มีการปล่อยเมาส์ซ้าย) และบันทึกเส้นที่วาดอยู่ (อยู่ระหว่างกดคลิกซ้าย) เพื่อให้สามารถแสดงผลขึ้นมาบนหน้าจอ หากวาดเสร็จแล้วให้แปลงรูปภาพเป็นขาวดำ และ Reshape ให้ตรงกับที่โมเดลต้องการเพื่อส่งภาพให้กับโมเดลจำแนก และส่งผลขึ้นมาเพื่อแสดงผลบนหน้าจอต่อไป

ตัวอย่างโค้ด

ส่วนเทรนด์

https://drive.google.com/file/d/1Lw0YMNUdRP8VwKyTIC6i34dPCD7VxdR2/view?usp=drive_link

ส่วน UI

https://drive.google.com/file/d/13qDV2vbMLyIIYPTQDRoJZFjfZYjloK6l/view?usp=sharing

2. Q-Learning (10 คะแนน)

Q-Learning คือ Algorithm ในรูปแบบ Reinforcement ซึ่งหมายถึงรูปแบบที่ Model จะทำการเรียนรู้เพื่อแลกให้กับแต้มรางวัล โดยอาศัยประสบการณ์ ในการเทรนด์รอบก่อนเพื่อใช้ในการพัฒนารูปแบบการตัดสินใจในรอบต่อไป เพื่อให้ได้ผลลัพธ์ที่ดีที่สุด แล้วนำไปพัฒนาเพื่อให้ตอบโจทย์ความต้องการในระยะยาว Reinforcement จึงอาศัยความถี่ในการฝึกเพื่อการเรียนรู้ แล้วนำไปพัฒนารูปแบบการทำงานให้ดีมากขึ้น จะเห็นงานได้จากการพัฒนาศัตรูในเกมต่าง ๆ เช่น dota2 หรือ Atari

Q-Learning เป็นการสนใจเกี่ยวว่าเมื่อโมเดลอยู่บน State (S) นี้ใน Environment แล้วทำ Action (A) นี้จะได้รับรางวัล Reward (R) อยู่ที่เท่านี้ แล้วเล่นจนจบเพื่อดูว่าจะได้ Reward รวมที่เท่าไหร่ แล้วนำมาเรียนรู้เพื่อให้ได้รับผลรวม Reward ที่ดีที่สุด

ส่วนสำคัญที่สุดของ Q-Learning Bellman ที่ใช้ในการหา และอัพเดทค่า Q-Value ที่มีหน้าที่บ่งบอกค่าที่ใช้ในการตัดสินใจหา Action ที่ดีที่สุด โดยการหาค่าโดย ใช้ค่า Q-Value ปัจจุบัน รวมกับค่า Learning rate ที่กำหนดไว้ให้โมเดลไม่ยึดติดกับผลการเทรนด์เดิม ๆ คูณร่วมกับค่า Discount rate เพื่อกำหนด noise เพื่อบอกให้โมเดลเข้าใจความคลาดเคลื่อนของผลลัพธ์ ร่วมกับค่าความคาดหวังจาก Next Q-Value

$$NewQ(s,a) = Q(s,a) + \alpha[R(s,a) + \gamma maxQ'(s',a') - Q(s,a)]$$

ขั้นตอนในการเทรนด์คือการกำหนด State และ Action ที่เป็นไปได้ แล้วเข้าขั้นตอนการเลือกว่าจะทำ Action อะไรผ่านประสบการณ์ และการเลือกค่า Epsilon แล้วจึงเริ่มการฝึกและคำนวณหาค่า Reward แล้วนำค่าที่ได้มาใช้ในการ Update Q-Value แล้วทำงานวนซ้ำไปเรื่อย ๆ

ในการเทรนด์จะสามารถปรับแต่งค่า Hyperparameter ได้ดังนี้

1. Learning rate (อัตราการเรียนรู้): ค่าที่กำหนดความสำคัญของการอัปเดตค่า Q-value ที่ได้รับจากประสบการณ์ใหม่ ค่าในช่วง 0 ถึง 1
2. Discount factor (อัตราส่วนการลดรางวัล): ใช้ในการปรับปรุงค่า Q-value โดยกำหนดความสำคัญของค่ารางวัลในอนาคต เพื่อคำนวณค่าความคาดหวังในการกระทำในสถานะต่อไป ค่านี้ต้องอยู่ในช่วงระหว่าง 0 ถึง 1
3. ค่า Epsilon: เพื่อการกำหนดแนวทางในการเลือก Action โดยค่า epsilon สูง (ใกล้เคียง 1) หมายถึงโมเดลมีประสบการณ์น้อย ให้ลอง Action ลองผิดลองถูก แล้วเมื่อค่าประสบการณ์สูงขึ้น ค่า Epsilon จะต่ำลง เพื่อให้โมเดลสนใจและทำตามสิ่งที่เคยทำมา
4. จำนวนรอบในการเทรนด์
5. ค่าการมอบรางวัลต่อ Action

ตัวอย่างโค้ด:
https://drive.google.com/file/d/1aubWLjgBEcaE8BDVcDrMvBklbwWJVVLc/view?usp=sharing

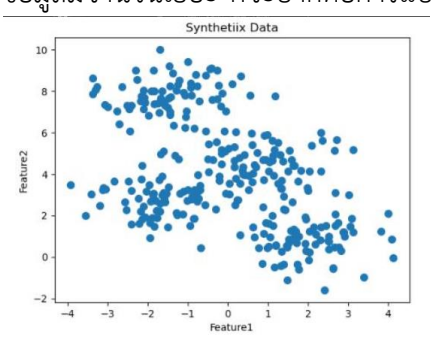
3. K-Mean (Unsupervised) (5 คะแนน)

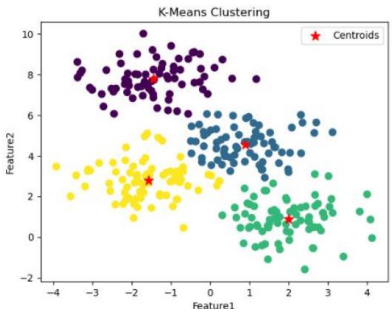
K-mean เป็น Algorithm แบบ Unsupervised คือการศึกษาเพื่อหา Pattern ของข้อมูลโดยไม่มี Labels บอกความสัมพันธ์ของข้อมูล โดย K-Mean จะสามารถใช้งานได้กับเฉพาะข้อมูลที่สามารถหาค่าเฉลี่ยของข้อมูล (ข้อมูลแบบตัวเลขที่สามารถดำเนินการทางคณิตศาสตร์ได้) เพราะจำเป็นต้องหาค่า K และค่าเฉลี่ยของข้อมูล

ขั้นตอนในการทำ K-mean คือการเลือกค่า K มา (ค่า K คือจำนวนของกลุ่มที่ต้องการแบ่ง) เพื่อหาค่า Centroid แล้วทำการเปรียบเทียบข้อมูลที่เข้าใกล้ Centroid นั้น ๆ มากที่สุด แล้วทำการเปลี่ยนค่า Centroid ใหม่ จากค่าเฉลี่ยของข้อมูลในกลุ่ม แล้วใช้ค่านั้นเป็น Centroid ใหม่ ทำซ้ำ ๆ จนกว่าค่าเฉลี่ยและข้อมูลที่อยู่ใกล้กับ Centroid ไม่มีการเปลี่ยนกลุ่มอีกแล้ว

ข้อดี: ง่าย สะดวก สามารถแบ่งแยกข้อมูลได้ถูกต้องแม่นยำ

ข้อเสีย: หากข้อมูลมีจำนวนเยอะ ก็จะยากต่อการแบ่ง และข้อมูลบางประเภทก็ไม่สามารถใช้ Algorithm นี้






```python
f1 = plt.figure(1)
plt.scatter(X[:,0], X[:,1], s = 50) # พล็อตข้อมูลแต่ละจุดบนกราฟ โดยใช้ข้อมูลจากคอลัมน์ที่ 0 และ 1 ของเมทริกซ์ X
plt.xlabel("Feature1") # กำหนดชื่อแกน x ว่า "Feature1"
plt.ylabel("Feature2") # กำหนดชื่อแกน y ว่า "Feature2"
plt.title("Synthetic Data") # กำหนดหัวข้อกราฟว่า "Synthetic Data"

kmeans = KMeans(n_clusters  = 4) # กำหนดโมเดล KMeans โดยกำหนดจำนวน cluster เป็น 4
kmeans.fit(X) # สร้างโมเดล KMeans จากข้อมูล X
centroids = kmeans.cluster_centers_ # คำนวณหาตำแหน่งของ centroids
labels = kmeans.labels_ # ระบุ label ของข้อมูลแต่ละจุด

# คำนวณหาระยะทางระหว่างแต่ละจุดข้อมูลกับ centroids
distances = np.zeros((X.shape[0], centroids.shape[0]))
for i, centroid in enumerate(centroids):
    distances[:, i] = np.linalg.norm(X - centroid, axis = 1)

# สร้าง DataFrame เพื่อแสดงค่าระยะทางระหว่างแต่ละจุดกับ centroids และ label ของ centroids ที่ถูกกำหนดให้
df_distances = pd.DataFrame(distances, columns=[f"Centroid{i + 1}" for i in range(centroids.shape[0])])
df_distances['Assigned_Centiod'] = labels + 1
df_distances.index.name = "data point"

print("Table showing distance of each data point to each centroid assigned centroid: ")
print(df_distances)

f2 = plt.figure(2)
plt.scatter(X[:, 0], X[:, 1], c = labels, s = 50, cmap = 'viridis') # พล็อตข้อมูลแต่ละจุดบนกราฟโดยให้มีสีของ
                                                                    # cluster ตาม label ที่ได้จากการแบ่งข้อมูล
plt.scatter(centroids[:, 0], centroids[:, 1], marker = '*', s = 100, c ='red', label = 'Centroids')
# พล็อตตำแหน่งของ centroids โดยให้เป็นดาว
plt.xlabel("Feature1") # กำหนดชื่อแกน x ว่า "Feature1"
plt.ylabel("Feature2") # กำหนดชื่อแกน y ว่า "Feature2"
plt.title("K-Means Clustering") # กำหนดหัวข้อกราฟว่า "K-Means Clustering"
plt.legend() # เพิ่มคำอธิบาย label ของ centroids
plt.show() # แสดงกราฟ
```

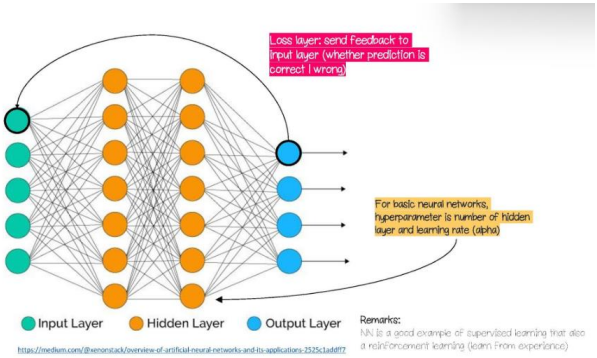
ในการเทรนด์จะมี Hyperparameter ที่สามารถปรับค่าแล้วส่งผลถึงของ Model ได้คือ

1. จำนวนกลุ่ม: กำหนดจำนวนกลุ่มที่ต้องการให้โมเดลคัดแยก
2. ค่า Centroid: สามารถเลือกค่าที่ดีที่สุดที่หาไว้เพื่อประสิทธิภาพ หรือสามารถตั้งสุ่มค่าได้
3. ระยะห่างจาก Centroid: กำหนดค่าระยะห่าง ข้อมูลจะมีระยะห่างจาก Centroid เท่าใดเพื่อการแบ่งกลุ่ม

---

4. Neural Network (Supervised) (5 คะแนน)

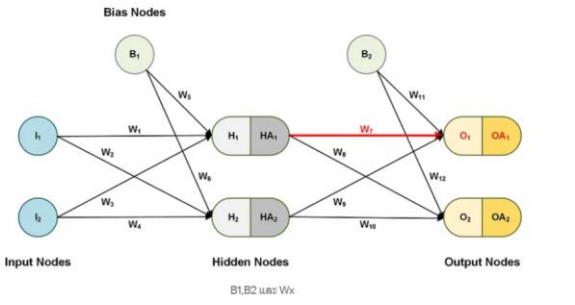
Neural Network เป็นรูปแบบของ Algorithm ที่เลียนแบบชื่อมาจากโครงข่ายระบบประสาทในสมอง เพราะว่าการทำ Neural Network จะมีการสร้าง Node ต่าง ๆ เพื่อการทำงานที่ต่างกันตาม Layer ประกอบไปด้วย 3 Layers คือ Input > Hidden > Output Layer



Input Layer: ส่วนของ Layer ที่นำเข้า และเตรียมข้อมูล แปลงข้อมูลให้เหมาะสมที่จะส่งต่อให้โมเดล

Hidden Layer: ส่วนของ Layer ที่เชื่อมต่อระหว่าง Input และ Output ซึ่งสามารถเพิ่มจำนวนได้ ทำหน้าที่ในการประมวลผลข้อมูล

Output Layer: หลังจากข้อมูลได้รับการประมวลผลแล้ว ก็จะมีค่าต่างกัน Output layer จะทำหน้าที่แยก Class ข้อมูลที่มีความแตกต่างกันตามที่ได้ตั้งไว้



ใน Hidden Layer จะมี Bias Nodes เพื่อให้การส่งต่อข้อมูลมีความเท่าเทียม ป้องกันไม่ให้ข้อมูลวิ่งเข้าผ่าน Node เดิม ๆ ตลอด

ขั้นตอนในการเทรนด์คือการกำหนดชุดข้อมูลออกเป็น 2 ชุดคือ Train และ Test คือการแบ่งชุดข้อมูลออกมาเป็นข้อมูลที่ใช้ในการฝึก และใช้ในการทดสอบ แล้วทำการ Normalization ให้ข้อมูลมีความเข้ากันและสามารถเข้าส่งข้อมูลเข้าไปเทรนด์ได้ เช่นการ Reshape ข้อมูล

หลังจากนั้นจึงทำการสร้าง Layer ทั้ง 3 แบบ ตั้งค่าเลือกชุดข้อมูลที่สนใจในการฝึก แล้วจึงเริ่มฝึกโมเดล

ในการเทรนด์จะมี Hyperparameter ที่สามารถปรับค่าแล้วส่งผลถึงของ Model ได้คือ

1. จำนวนชั้น: คือจำนวนของ Layer ที่จะตั้งค่าเพื่อปรับข้อมูล การเพิ่ม Layer ใน Hidden layer ที่ส่งผลถึงการฝึก เช่น Dropout ที่ช่วยปิด Node บางจุดเพื่อป้องกันปัญหา Overfitting
2. จำนวน Node ในแต่ละ Layer: คือการตั้งค่า Node ที่จะใช้ในการกรองข้อมูลในแต่ละ Layer
3. Activation functions: ฟังก์ชั่นการทำงานต่าง ๆ ที่ทำงานใน Layer เช่น Relu
4. จำนวนรอบการฝึกฝน: ค่า Epoch ที่สามารถส่งผลต่อความถูกต้องของโมเดลได้
5. Optimizer algorithm: อัลกอริธึมที่ปรับแต่งค่าน้ำหนักในการฝึกเช่น ADAM
6. จำนวนข้อมูลที่ใช้ในการ Train Test ว่ามีอัตราส่วนเท่าไหร่

ข้อดี: ความสามารถสูง ทำงานกับข้อมูลที่ซับซ้อนได้ดี
มีความยืดหยุ่นสูง สามารถ Optimizer ตัว Parameter ได้หลากหลาย

ข้อเสีย: มีความซับซ้อนสูง เนื่องจากต้องทำการตั้ง Optimizer เอง
ข้อมูลที่ใช้ต้องมีจำนวนมาก
ใช้ทรัพยากร ประสิทธิภาพอุปกรณ์สูง และใช้เวลามาก

โค้ดพร้อม Comment ให้ดูงานกลุ่ม
https://drive.google.com/file/d/1Lw0YMNUdRP8VwKyTIC6i34dPCD7VxdR2/view?usp=drive_link

---

5. Supervise Algorithm สุ่ม ๆ 1 แบบ (5 คะแนน)