CS2109S

AI

BFS

1. Define state representation
2. Identify initial state
3. Identify goal state(s) / goal test
4. Identify transitions

```
from collections import deque

def search(...):
  def is_goal(node):
    state, steps = node
    # DEFINE GOAL STATE TESTS HERE
    return state == GOAL_STATE

  def expand(node):
    next_nodes = []
    state, steps = node
    # DEFINE TRANSITIONS HERE
    return next_nodes

  q = deque()
  visited = set()
  q.append(INITIAL_STATE)

  while len(q) > 0:
    cur = q.popleft()

    if is_goal(cur):
      return cur[1]

    for move in expand(cur):
      state, steps = move
      if state not in visited:
        visited.add(state)
        q.append(move)

  return False
```

Tuple operations

```
t = 3,5,7

# Indexing
print(t[1]) # 5
print(t[-1]) # 7

# Appending a tuple
print (t + ((1,2),)) # (3, 5, 7, (1, 2))
```

PROBLEM SETS

PS1 (Informed Search)

- Tree: Q1 (BFS), Q2 (DFS)
- Graph: Q3 (BFS), Q5 (DFS)
- A*: Q6

PS3 (Regression)

Linear Regression

- MSE: Q1
- MAE: Q2
- Bias column: Q3
- Get bias and weights: Q4
- Prediction line: Q5
- Gradient descent: Q6 (one-var), Q7 (multi-var)

Polynomial Regression

- Create matrix: Q9
- Prediction line: Q10
- Feature scaling: Q11
- Find iterations for convergence: Q12

PS4 (Logistic Regression and SVM)

Logistic Regression (binary)

- Undersampling: Q2, oversampling: Q3
- Train-test split: Q4
- Cost function: Q5
- Weight update for BGD: Q6
- Classification: Q7
- BGD: Q8
- SGD: Q9

Logistic Regression (multi)

- BGD: Q12
- Classification: Q13

Support Vector Machines

- Linear kernel: Q14
- Gaussian kernel: Q15

PS5 (PyTorch and Neural Networks)

PyTorch

- Fit polynomial: Q2

Neural Networks (Fit $y = |x - 1|$)

- Manual: Q4 (forward pass), Q5 (backprop)
- PyTorch: Q7 (forward pass), Task 3.2 (backprop), Q8 (obtain model weights)

DigitNet

- Model architecture and forward pass: Q9
- Training loop: Q10
- Accuracy of model: Q11

PS6 (ConvNets)

Manual implementations

- Convolution: Q1
- Max pool: Q2

ConvNet (digits)

- Vanilla: Q3 (architecture), Q5 (training)
- Dropout: Q4 (architecture), Q6 (training)

CIFAR-10 (image)

- Augmentations: Q9
- Architecture: Q11
- Training: Q12
- CAM (class activation mapping) heatmap: Q13

PS7 (Clustering)

K-Means

- Assing points to clusters: Q1
- Update centroids: Q2
- Check convergence: Q3
- K-Means: Q4 (one iteration), Q6 (full)
- Loss: Q5
- Compress image: Q7

Image classification

- Predict labels using K-Means: Q9
- PCA with SVD: Q10
- No. of components for $\geq$ 99% explained variance for PCA: Task 2.2.3
- K-Means with PCA: Q13
- Predict labels using K-Means with PCA: Q15

Mock Paper

- AI
- Regression
- Classification

1. Missionaries and Cannibals
2. 2A: Feature engineering
3. 2B: Feature scaling
4. 2C: Loss function
5. 2D: Gradient descent
6. 2E: Guess equation
7. 3A: Split train/test
8. 3B: Loss function
9. 3C: Gradient loss function
10. 3D: Logistic regression classification
11. 3E: Confusion matrix
12. 3F: Metrics (Precision, Recall, F1)

ML

Backpropagation

```
# reset gradients to 0
optimiser.zero_grad()
# get predictions
y_pred = model(x)
# compute loss
loss = loss_fn(y_pred, y)
# backpropagate
loss.backward()
# update the model weights
optimiser.step()
```

Print PyTorch NN weights

```
print(model.state_dict())
```

Explained variance of PCA

```
print(sum(pca.explained_variance_ratio_))
```