Rubric

Q2: Representations (50 points)

Q2.1 Task 1 (10 points)

1. (+10) Correct solution
2. (+3) Uses `LogisticRegression` with `penalty=None` and `random_state=2025`, fits on `X, y`, calculates predictions and accuracy
3. (+2) Calls `plot_decision_boundary(X, y, lr_model)` with fitted model
4. (+2) Reports numerical accuracy value (~0.5–0.6)
5. (+3) Explanation mentions:
    - data is non-linearly separable/circular
    - logistic regression only learns linear boundaries
    - linear boundary cannot separate circles

Q2.2 Task 2 (10 points)

1. (+10) Correct solution
2. (+3) `compute_X_rep` method returns distance from origin, reshaped to 2D
3. (+2) Creates `RepModel` instance, calls `.fit(X, y)`, makes predictions
4. (+1) Reports accuracy ≥ 0.70
5. (+2) Calls `plot_decision_boundary(X, y, rep_model)`
6. (+2) Explanation: distance feature makes inner/outer circles linearly separable in 1D

Q2.3 Task 3 (8 points)

1. (+8) Correct solution
2. (+3) Uses `MLPClassifier(hidden_layer_sizes=(8,), random_state=2025, max_iter=2000, alpha=0.0)`
3. (+2) Fits model on `X, y`, makes predictions
4. (+1) Reports accuracy ≥ 0.70
5. (+2) Calls `plot_decision_boundary(X, y, mlp_model)`

Q2.4 Task 4 (12 points)

1. (+12) Correct solution
2. (+2) Uses `train_test_split(test_size=0.5, random_state=2025)`
3. (+2) Creates small MLP with `(8,)` hidden layer, correct parameters
4. (+2) Creates large MLP with `(1000,1000)` hidden layers, correct parameters
5. (+2) Reports train and test accuracy for both networks (4 values total)
6. (+2) Shows decision boundaries for both models using training data
7. (+2) Correctly calculates parameter counts:
    - Small network ≈ 24 parameters
    - Large network ≈ 1,000,000+ parameters

Q2.5 Task 5 (10 points)

1. (+10) Correct solution
2. (+3) Uses `MLPClassifier(hidden_layer_sizes=(1000,1000), early_stopping=True, validation_fraction=0.4, n_iter_no_change=10)`
3. (+2) Sets `random_state=2025, max_iter=2000, alpha=0.0`
4. (+2) Fits early stopping model on training data from Task 4 split
5. (+2) Reports both train and test accuracy
6. (+1) Calls `plot_decision_boundary` with early stopping model on training data

---

Q3: MLP (50 points)

Q3.1 Task 1 (9 points)

1. (+9) Correct solution
2. (+3) Uses `MLPClassifier(hidden_layer_sizes` between `(50,)` and `(200,)`, `max_iter=500, random_state=2025`)
3. (+2) Calls `.fit(X_train, y_train)`, `.predict(X_train)`, `.predict(X_test)`
4. (+2) Reports training accuracy ≥ 90%
5. (+2) Reports numerical test accuracy

Q3.2 Task 2 (19 points)

1. (+19) Correct solution
2. (+2) `__init__` initializes `W1`, `W2`, sets hyperparameters as attributes
3. (+2) `forward` implements `X @ W1 → relu() → @ W2 → softmax()`
4. (+2) `predict_proba` calls `forward`, `predict` uses `argmax`
5. (+3) `backward` contains correct gradients: `a1.T @ (y_pred - y_onehot)` and ReLU derivative
6. (+5) Implements helper methods: `relu`, `softmax`, `to_onehot`, `shuffle_data`, `get_batch`
7. (+3) `fit` has epoch loop, batch processing, weight updates with learning rate
8. (+2) Cross-entropy formula present: `- y_onehot * log(y_pred)`, averaged

Q3.3 Task 3 (10 points)

1. (+10) Correct solution
2. (+3) Creates `NeuralNet(in_size=64, out_size=10, h_size=100–500, lr=0.005–0.01, batch_size=16–64)`
3. (+2) Calls `fit(X_train, y_train, verbose=True)`, shows training progress
4. (+3) Reports training accuracy ≥ 80%
5. (+2) Reports numerical test accuracy with `predict`

Q3.4 Task 4 (12 points)

1. (+12) Correct solution
2. (+3) Tests learning rates `[0.0001, 0.001, 0.01]` with verbose training
3. (+3) Tests batch sizes `[16, 128, 1024]` with verbose training
4. (+3) Fills table with epochs needed to reach >80% accuracy (or “N/A”)
5. (+2) Shows evidence of systematic experimentation (code/output)
6. (+1) Records results in some form (printed outputs, comments, or table)