```
from sklearn.datasets import load_iris
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report


#Load dataset
data = load_iris()
X = data.data
y = data.target

#Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

#Define model and parameters
mlp = MLPClassifier(max_iter=1000)

parameters = {
    'hidden_layer_sizes': [(10,), (50,), (100,)],
    'activation': ['relu', 'tanh', 'logistic'],
    'learning_rate': ['constant', 'invscaling', 'adaptive']
}

#Grid search
cv = GridSearchCV(mlp, parameters, cv=5)
cv.fit(X_train, y_train)

#Print results
print("Best Parameters:", cv.best_params_)
print("Classification Report:\n", classification_report(y_test, cv.predict(X_test)))
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic
```

```
              0       1.00      1.00      1.00        19
              1       1.00      1.00      1.00        13
              2       1.00      1.00      1.00        13

       accuracy                           1.00        45
      macro avg       1.00      1.00      1.00        45
   weighted avg       1.00      1.00      1.00        45

/usr/local/lib/python3.11/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic
  warnings.warn(
```

1) What is the difference between the learning rates: Constant, Invscaling, and Adaptive?

Constant: Keeps the same learning rate throughout training.

Invscaling: Decreases the learning rate as training progresses.

Adaptive: Lowers learning rate only when the model stops improving.

2) What is feed-forward vs backpropagation in ANN?

Feed-forward: Data passes from input → hidden → output layer to make predictions.

Backpropagation: Model calculates errors, then adjusts weights by moving backward through the network.

3) What is Batch vs Online training?

Batch: Model trains on the full dataset at once or in mini-batches.

Online: Model updates after every individual data point. Good for streaming data.

4) What is Gradient Descent and what does it do?

Gradient Descent is an optimization algorithm.

It updates the model's weights to minimize the error by moving step-by-step toward the lowest loss value.