LAB 5

Grey Wolf Optimizer (GWO):

Application:

Implementation of Grey Wolf Optimizer (GWO) for feature selection in machine learning. The objective is to select the best subset of features that minimizes the classification error.

Code:

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Objective function for feature selection
def feature_selection_fitness(solution, X_train, X_test, y_train, y_test):
    selected_features = np.where(solution > 0.5)[0]  # Select features based on threshold
    if len(selected_features) == 0:  # If no features are selected, return a high error
        return 1.0
    # Use selected features for training and testing
    X_train_sel = X_train[:, selected_features]
    X_test_sel = X_test[:, selected_features]
    # Train SVM classifier
    classifier = SVC()
    classifier.fit(X_train_sel, y_train)
    y_pred = classifier.predict(X_test_sel)
    # Return fitness (1 - accuracy)
    accuracy = accuracy_score(y_test, y_pred)
    return 1 - accuracy

# Grey Wolf Optimizer for feature selection
class GreyWolfOptimizer:
    def __init__(self, fitness_func, num_features, num_wolves=10, max_iter=20):
        self.fitness_func = fitness_func
        self.num_features = num_features
        self.num_wolves = num_wolves
        self.max_iter = max_iter
        self.alpha_pos = None  # Best solution (alpha wolf)
        self.alpha_score = float("inf")
        self.beta_pos = None
        self.beta_score = float("inf")
        self.delta_pos = None
        self.delta_score = float("inf")

    def optimize(self, X_train, X_test, y_train, y_test):
        # Initialize population (wolves' positions)
        wolves = np.random.rand(self.num_wolves, self.num_features)
        for iteration in range(self.max_iter):
            for i in range(self.num_wolves):
                # Evaluate fitness of each wolf
                fitness = self.fitness_func(wolves[i], X_train, X_test,
y_train, y_test)
                # Update alpha, beta, and delta wolves
                if fitness < self.alpha_score:
                    self.alpha_score, self.alpha_pos = fitness,
wolves[i].copy()
                elif fitness < self.beta_score:
                    self.beta_score, self.beta_pos = fitness, wolves[i].copy()
                elif fitness < self.delta_score:
                    self.delta_score, self.delta_pos = fitness,
wolves[i].copy()

            # Update positions of wolves
            a = 2 - iteration * (2 / self.max_iter)  # Linearly decreasing
parameter
            for i in range(self.num_wolves):
                r1, r2 = np.random.rand(), np.random.rand()
                A1, C1 = 2 * a * r1 - a, 2 * r2
                D_alpha = abs(C1 * self.alpha_pos - wolves[i])
                X1 = self.alpha_pos - A1 * D_alpha

                r1, r2 = np.random.rand(), np.random.rand()
                A2, C2 = 2 * a * r1 - a, 2 * r2
                D_beta = abs(C2 * self.beta_pos - wolves[i])
                X2 = self.beta_pos - A2 * D_beta

                r1, r2 = np.random.rand(), np.random.rand()
                A3, C3 = 2 * a * r1 - a, 2 * r2
                D_delta = abs(C3 * self.delta_pos - wolves[i])
                X3 = self.delta_pos - A3 * D_delta

                wolves[i] = (X1 + X2 + X3) / 3  # Average position
                # Ensure wolves stay within bounds
                wolves[i] = np.clip(wolves[i], 0, 1)

        # Return the best solution
        return self.alpha_pos, 1 - self.alpha_score  # Features and accuracy

# Main function
def main():
    # Ask the user for inputs
    print("Welcome to the Grey Wolf Optimizer for Feature Selection!")
    dataset_choice = input("Choose dataset: (1) Iris [default], (2) Custom: ")
    if dataset_choice == "1" or dataset_choice == "":
        data = load_iris()
        X, y = data.data, data.target
    else:
        print("Custom dataset support not yet implemented. Using Iris dataset as default.")
        data = load_iris()
        X, y = data.data, data.target

    population_size = int(input("Enter the number of wolves in the population (e.g., 10): ") or 10)
    max_iterations = int(input("Enter the maximum number of iterations (e.g., 20): ") or 20)

    # Split data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

    # Define GWO parameters
    num_features = X.shape[1]
    gwo = GreyWolfOptimizer(
        fitness_func=feature_selection_fitness,
        num_features=num_features,
        num_wolves=population_size,
        max_iter=max_iterations,
    )

    # Run optimization
    best_solution, best_accuracy = gwo.optimize(X_train, X_test, y_train, y_test)

    # Output results
    selected_features = np.where(best_solution > 0.5)[0]
    print("\nOptimization Results:")
    print(f"Selected Features: {selected_features}")
    print(f"Best Accuracy: {best_accuracy:.4f}")
```

```python
if __name__ == "__main__":
    main()
```

Output:

```
Welcome to the Grey Wolf Optimizer for Feature Selection!
Choose dataset: (1) Iris [default], (2) Custom: 1
Enter the number of wolves in the population (e.g., 10): 10
Enter the maximum number of iterations (e.g., 20): 20

Optimization Results:
Selected Features: [0 2 3]
Best Accuracy: 1.0000
```