Department of Computer Science

Course Title: Evolutionary Computing (2025)

Assignment 3: Training Logistic Regression with Evolution Strategies (ES)

1. Introduction

Evolution Strategies (ES) are evolutionary optimization methods designed for continuous parameters. In this assignment, you will use ES to train a binary logistic regression classifier on the Heart Disease dataset. Because ES are derivative-free, they are applicable even when the objective is noisy or non-differentiable. Here, we will use ES to optimize the model parameters by minimizing cross-entropy loss (with optional L2 regularization) and track performance on separate train and test splits.

---

2. Problem Definition

- Dataset: Heart Disease dataset (binary classification: presence/absence of heart disease; target ∈ {0,1}).
- Task: Predict whether a patient has heart disease based on clinical features (e.g., age, sex, chest pain type, exercise-induced angina, ST depression, etc.).
- Model: Binary logistic regression that maps input features $x \in \mathbb{R}^d$ to a probability $\hat{y} \in [0, 1]$ using the logistic sigmoid:

$$\hat{y} = \sigma(W^\top x + b),$$

where $W \in \mathbb{R}^d$ is the weight vector, $b \in \mathbb{R}$ is the bias, and $\sigma(z) = \frac{1}{1+e^{-z}}$.

- Goal: Implement an Evolution Strategy that learns (W, b) to minimize cross-entropy loss (with optional L2 regularization) and achieve good classification performance.

You will evaluate your final model on a test split and report accuracy, precision, recall, F1-score, and a confusion matrix, along with several diagnostic plots.

3. Data & Preprocessing

1. **Dataset**
    - **Use a standard Heart Disease dataset for binary classification (e.g., the Cleveland subset or equivalent), where the target label is 0 (no disease) or 1 (disease).**
2. **Train/ test Split**
    - **Split the dataset into:**
        - **70% training data**
        - **30% test data**
    - **The split must be stratified with respect to the target label to preserve the class distribution in both train and test sets.**
3. **Feature Scaling**
    - **Apply feature scaling to all continuous input features (and optionally to ordinal ones), for example using z-score standardization:**

$$x_j^{\text{scaled}} = \frac{x_j - \mu_j}{\sigma_j},$$

where μ_j and σ_j are the mean and standard deviation of feature j computed on the training set only.

    - **Apply the same μ_j and σ_j to scale the test set. Do not recompute them on the test data.**

4. Algorithm Design

4.1 Representation

- **Let d be the number of input features.**
- **Model parameters:**

$$\boldsymbol{\theta} = [\boldsymbol{W}, \boldsymbol{b}] \in \mathbb{R}^{d+1},$$

i.e., $\theta$is the flattened concatenation of weights $W$and bias b.

- **Strategy parameters:**
  - **A vector $\sigma \in \mathbb{R}^{d+1}$of mutation step sizes, one per parameter.**
- **Individual (candidate solution):**

$$\text{individual} = [\theta, \sigma],$$

with total length $2(d+1)$.

4.2 Initialization

- **Initialize model parameters:**

$$W, b \sim \mathcal{U}(-0.1, 0.1).$$

- **Initialize step sizes:**

$$\sigma_i \sim \mathcal{U}(0.01, 0.1) \text{for each } i = 1, \ldots, d+1.$$

4.3 Mutation (Self-Adaptive ES)

We use self-adaptive mutation with one global and one local learning rate.

1. **Learning Rates**

Let $n = |\theta|$be the dimensionality of the parameter vector θ(here $n = d+1$). Define:

$$\tau = \frac{1}{\sqrt{2n}}, \tau' = \frac{1}{\sqrt{2\sqrt{n}}}.$$

- **τ is the global learning rate (shared across all dimensions).**
- **$\tau'$is the local learning rate (dimension-specific).**

2. **Strategy Parameter Update**

For each individual, sample:

- **One global perturbation $N(0,1)$,**
- **Independent local perturbations $N_i(0,1)$for each dimension i.**

Then update the step sizes:

$$\sigma_i' = \sigma_i \cdot \exp\left(\tau \cdot N(0,1) + \tau' \cdot N_i(0,1)\right), i = 1, \ldots, n.$$

3. **Object Parameter Update**

Using the updated step sizes σ', update the parameters:

$$\theta_i' = \theta_i + \sigma_i' \cdot N_i(0,1), i = 1, \ldots, n.$$

4. **Stability Constraints**

- **Parameter clipping: clip each component of $\theta'$to the range $[-5, 5]$.**
- **Step size lower bound: enforce**

$$\sigma_i' \geq 10^{-6} \text{for all } i.$$

4.4 Evaluation / Fitness

1. **Model Output**

For an input x, the model predicts:

$$\hat{y} = \sigma(W^\top x + b)$$ **, where $\sigma(\cdot)$is the logistic sigmoid.**

2. **Loss Function**

Given a training set $\{(x^{(k)}, y^{(k)})\}_{k=1}^{N}$with labels $y^{(k)} \in \{0,1\}$, define:

- **Cross-entropy loss:**

$$\mathrm{CE}(\theta) = -\frac{1}{N}\sum_{k=1}^{N}\left[y^{(k)}\log\hat{y}^{(k)} + (1-y^{(k)})\log(1-\hat{y}^{(k)})\right].$$

- **Optional L2 regularization (on weights only):**

$$\mathrm{L2}(W) = \| W \|_2^2.$$

The total loss is:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathrm{CE}(\boldsymbol{\theta}) + \lambda_{\mathrm{reg}} \parallel \boldsymbol{W} \parallel_2^2,$$

where $\lambda_{\mathrm{reg}} \geq 0$is the L2 regularization coefficient.
(Note: we use $\lambda_{\mathrm{reg}}$to avoid confusion with λ, the number of offspring in ES.)

3. **Fitness Function**

Define the fitness to maximize as:

$$\mathrm{fitness}(\boldsymbol{\theta}) = -\mathcal{L}(\boldsymbol{\theta}),$$

i.e., lower loss corresponds to higher fitness. This provides a smoother learning signal than accuracy.

4.5 Selection

Implement one of the following:

- (μ, λ)**-ES**
  - **From μparents, generate λoffspring ($\lambda > \mu$).**
  - **Only offspring are considered for selection: sort offspring by fitness and choose the best μindividuals as parents for the next generation.**
- $(\mu + \lambda)$**-ES**
  - **From μparents, generate λoffspring.**
  - **Combine parents and offspring ($\mu + \lambda$ individuals), sort by fitness, and select the best μas the next generation.**
- **Example configuration:**

$$\mu = 30, \lambda = 210.$$

You should implement elitism by sorting individuals by fitness and keeping the best ones according to the chosen scheme.

---

5. Outputs & Visualization

Your implementation must produce (at least) the following plots and metrics.

5.1 Plot 1: Best and Mean Fitness vs. Generations (Training Loss)

- **For each generation:**
  - **Compute the training loss $\mathcal{L}(\theta)$ for all individuals in the population.**
  - **Record:**
    - **the best (minimum) training loss among all individuals,**
    - **the mean training loss across the population.**
- **Plot these two curves (best and mean) as a function of the generation index (or evaluation count).**

5.2 Plot 2: Training Accuracy vs. Generations

- **For each generation, identify the best individual in that generation (the one with the lowest training loss).**
- **Compute the classification accuracy on the training set for this best individual.**
- **Plot training accuracy versus generation index.**

5.3 Plot 3: Final Confusion Matrix on the test Set

- **After the ES algorithm terminates, select the best overall individual (e.g., the one with the lowest training loss across all generations or the best test performance).**
- **Evaluate this final model on the test set and report:**
  - **the confusion matrix,**
  - **and the corresponding test metrics:**
    - **accuracy,**
    - **precision,**
    - **recall,**
    - **F1-score.**

5.4 Optional: test Accuracy vs. Generations

- **Optionally, at each generation, evaluate the best individual on the test set and compute test accuracy.**
- **Plot test accuracy vs. generation index (in the same figure as training accuracy or in a separate figure) to analyze generalization and potential overfitting.**

---

6. Experiments & Analysis

In your report, include a short analysis of your experimental results:

1. **Convergence Behavior**
    - **Discuss the trend of best/mean fitness over generations.**
    - **Does the algorithm converge? Is there evidence of stagnation?**
2. **Sensitivity Analysis**
    - **Briefly analyze the effect of varying:**
        - **population size (μ, λ),**
        - **learning rates τ, τ',**
        - **regularization coefficient λ_{reg}.**
    - **You do not need an exhaustive grid search, but show at least a few different configurations and compare their performance qualitatively.**
3. **Error Analysis**
    - **Use the final confusion matrix to comment on typical misclassifications.**
    - **Are false positives or false negatives more common?**
    - **What might this imply in a medical context?**

---

7. Constraints & Deliverables

1. **Allowed Libraries**
    - **You may use:**
        - **Python**
        - **NumPy**

        - **Pandas**
        - **Matplotlib**
    - **You may use scikit-learn only for:**
        - **data utilities (e.g., train_test_split with stratification), and**
        - **evaluation metrics (e.g., accuracy_score, precision_recall_fscore_support, confusion_matrix).**
    - **You must not use scikit-learn's built-in LogisticRegression**

2. Deliverables

- **Runnable code (Python scripts or a Jupyter notebook) implementing:**
    - **data loading and preprocessing,**
    - **ES algorithm (representation, mutation, selection),**
    - **evaluation and plotting.**
- **PDF report that includes:**
    - **a short description of your algorithm design,**
    - **hyperparameters used ($\mu, \lambda, \tau, \tau', \lambda_{\text{reg}}$, number of generations, etc.),**
    - **all required plots (fitness curves, training accuracy curve, final confusion matrix, optional test accuracy),**
    - **experimental analysis and conclusions.**
- **README file with:**
    - **instructions on how to run your code,**
    - **environment/dependency information,**
    - **any notes needed for reproduction.**