Feature Selection for Classification using Particle Swarm Optimization (PSO)

Numerical Methods & Optimization – ENSIA

Project Team:

Bouabdelli Moulay Mohamed - G1

Chaalel Omar Ziyad - G5

Frihaoui Mohammed - G1

Supervised by: Dr. Lakehal Soumaya

May 9, 2025

Contents

1 Introduction

Feature selection is a very popular data pre-processing technique that simplifies models by removing unimportant or redundant features, making them more interpretable and stable in learning in some cases. The chosen subset can then be tested against a fitness function to assess its optimality.

As the number of dimensions (features) increase, finding said optimal subset through search becomes an intractable problem [3]. And many problems that are related or can be reduced to this problem are NP hard problems[1].

The dataset we will use for this project is the *Breast Cancer Wisconsin (Diagnostic) Dataset* first introduced in [7] obtained from the *scikit-learn API*. It contains 30 features related to breast cancer diagnoses, where the goal is to predict whether a tumor is malignant or benign.

Particle Swarm Optimization (PSO) would be beneficial in this case to select a near-optimal subset of features that most impact the perfect classifier decisions without having to explore an intractable search space.

2 Problem Definition and Mathematical Model

2.1 Objective

The objective is to find the optimal subset of features that results in the highest performance of a classifier. Performance measure is *accuracy* defined as :

$$\text{Accuracy} = \frac{TP + TN}{TN + TP + FP + FN}$$

where {TP, TN, FP, FN} are the components of a binary classifier confusion matrix.

2.2 Mathematical Formulation

Let:

- n: number of features.
- $x = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$: binary mask vector indicating selected features.
- $f(x)$: classification accuracy of model using features selected by x.

Optimization Problem:

$$\max_{x \in \{0,1\}^n} f(x)$$

Constraints: Optional constraint to limit number of selected features:

$$\sum_{i=1}^{n} x_i \leq k \quad \text{(optional)}$$

Alternatively, we will reformulate the problem slightly into minimization by using the fact that

$$\max_{x \in \{0,1\}^n} f(x) \leq 1$$

and factoring in the penalty from the constraint as follows :

$$\min_{x \in \{0,1\}^n} S(x)$$

Where :

$$S(x) = 1 - f(x) + 0.01 \cdot \frac{\text{num\_features(x)}}{\text{total\_features}} \cdot \mathbb{1}_{num\_features(x) > 20}(x)$$

Problem Complexity Analysis

The feature selection task described above is a form of *subset selection* over n binary decision variables. Since the number of possible subsets is 2^n, the search space grows exponentially with the number of features. This combinatorial explosion renders exhaustive search infeasible for even moderately sized n.

Furthermore, the general subset selection problem for optimizing a classification performance metric (such as accuracy) is known to be NP-Hard [1]. This is because evaluating each subset requires training and validating a classifier, and no known polynomial-time algorithm can guarantee the global optimum. Thus, heuristic and metaheuristic methods like Particle Swarm Optimization (PSO) are commonly employed to explore this space efficiently.

3 Particle Swarm Optimization (PSO)

3.1 PSO Overview

Particle Swarm Optimization (PSO) is a computational method designed to optimize problems by iteratively refining candidate solutions with respect to a specified measure of quality while requiring minimal parametrization[5].

This paradigm was initially introduced by James Kennedy and Russel Ebhart in 1995[5], it is inspired by *Swarm Intelligence* which is a social behavioral pattern known in many animals that live in cooperative societies particularly fish and birds flocking [2].

The algorithm can be thought of as an analogy to birds flocking in the following manner, a bird flock (swarm of particles consituting a set of states) travels around its habitat (the problem state space) using group and individual learning looking for a place for them to settle down in which shelters them and provides them with food ..., the latter can be abstracted to the notion of a *fitness function* which the algorithm optimizes [6].

3.2 Binary PSO

Binary Particle Swarm Optimization (BPSO) is a variant of the standard PSO algorithm adapted for discrete or binary search spaces. It is particularly used for subset selection problems, where the solution space consists of binary vectors indicating the inclusion (1) or exclusion (0) of features.

In BPSO, the position of each particle is represented by a binary vector, and the velocity reflects the probability of a bit being 1. Unlike standard PSO, which updates positions based on continuous values, BPSO uses a probabilistic mechanism to map the continuous function to a binary output based on a threshold, which reduces the problem to searching in a finite space which can in theory speed up the convergence of particles [8].

- **Velocity update formula**:

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (p_i - x_i(t)) + c_2 \cdot r_2 \cdot (g - x_i(t))$$

  where:

  - $v_i(t)$: velocity of particle i at time t
  - $x_i(t)$: current position of particle i
  - p_i: best known position of particle i (personal best)
  - g: global best position among all particles
  - $r_1, r_2 \sim U(0,1)$: random numbers
  - w: inertia weight
  - c_1, c_2: cognitive and social coefficients

- **Sigmoid transfer function** (used to squash velocity into a probability range [0,1]):

$$S(v_i) = \frac{1}{1+e^{-v_i}}$$

- **Position update rule** (based on probability output by the sigmoid):

$$x_i = \begin{cases} 1, & \text{if } r < S(v_i) \\ 0, & \text{otherwise} \end{cases} \quad \text{where } r \sim U(0,1)$$

4 Implementation Details

4.1 Programming Environment

- Language: Python
- Libraries: NumPy (Matrix-Like Data Manipulations), scikit-learn (Dataset, Classifier & Metrics).

Why Python ? Python allows for faster implementations and testing of ideas due to its high level nature, abstracting significant boilerplate code and memory management that would otherwise be accounted for in many other technologies like C++ or MATLAB.

Futhermore, has a flourishing ecosystem of libraries for computation and machine learning in general. And through tools like Cython, Python code can be compiled into C++/C to offload high-performance components, leaving the focus on algorithm design, evaluation and visualization rather than technology dependent optimization tweaks.

4.2 Fitness Function & BPSO

- Accuracy of classifier on validation data.
- Include penalty for number of selected features to prevent the swarm from favoring solutions with excessive number of features.
- We set BPSO to run for 50 iterations with an early stopping patience parameter set to 20, i.e if the algorithm does not find any improvement 20 iterations in a row, we stop it.
- The swarm is set to contain 30 particles.
- We use the minimization formulation of the problem and include the penalty as we discussed above, penalty multiplier is set to 0.01.
- All these *hyper-parameters* can be subject to fine tuning through learning in a continuous state space via gradient descent.

4.3 Classifier Used

- Random Forest with 10 estimators each using *gini* index for loss.

5 Results and Analysis

In this section, we present the results obtained from applying Particle Swarm Optimization (PSO) for feature selection and its effect on classifier performance. We provide an in-depth analysis of the optimization process, the performance metrics, and convergence behavior. Additionally, we include visualizations and tables to better illustrate the outcomes.

5.1 Performance Metrics

The optimization process led to significant improvements in the classification performance. Here are the key performance metrics:

- **Best Accuracy Achieved**: 0.9942 (99.42%)
- **Number of Iterations**: 45
- **Optimization Time**: 17.61 seconds
- **Early Stopping**: Triggered due to lack of further improvement

The achieved accuracy of 99.42% indicates that the selected subset of features was highly discriminative, leading to excellent model performance on the dataset.

5.2 Convergence Behavior

The convergence curve shows how the accuracy of the classifier evolved over the iterations of the PSO algorithm. Below is the plot representing the accuracy across iterations:

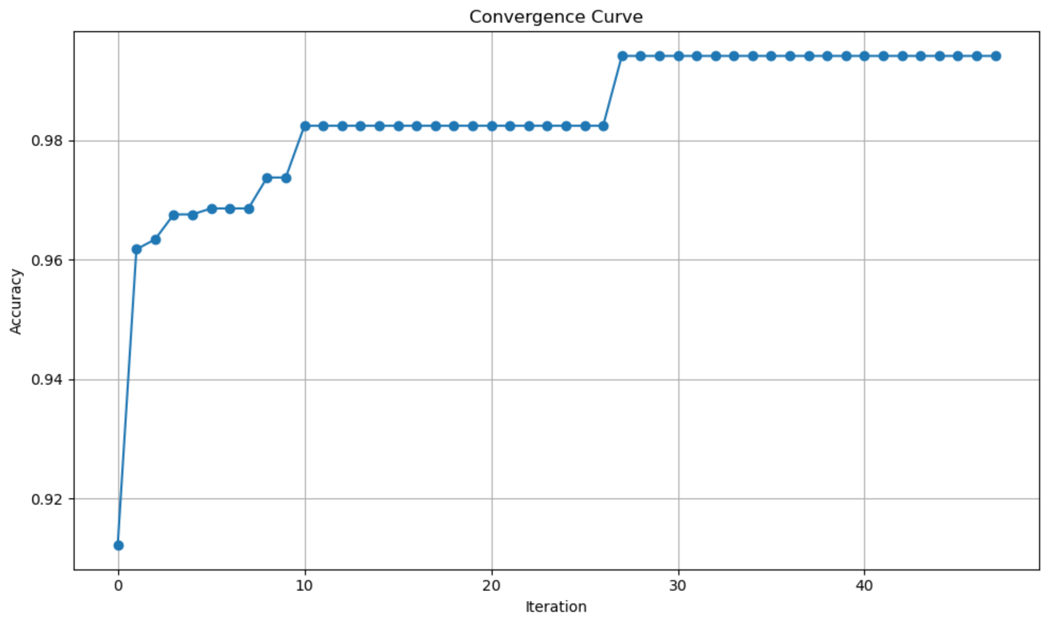

Figure 1: Convergence curve showing accuracy over iterations. The accuracy increases rapidly during the early iterations, reaching a plateau around iteration 28. This plateau reflects the algorithm's convergence to a local or global optimum.

The **convergence curve** (Figure 1) shows rapid improvements in accuracy during the initial iterations, indicating that the swarm was effectively exploring the feature space. The accuracy then stabilizes after iteration 28, which suggests that the optimization process has found a near-optimal subset of features. Early stopping was triggered after 45 iterations, signaling that further exploration would not yield significant improvements.

5.3 Feature Selection Frequency

We also analyzed the frequency with which individual features were selected across the particles in the swarm. This is visualized in the following bar chart:

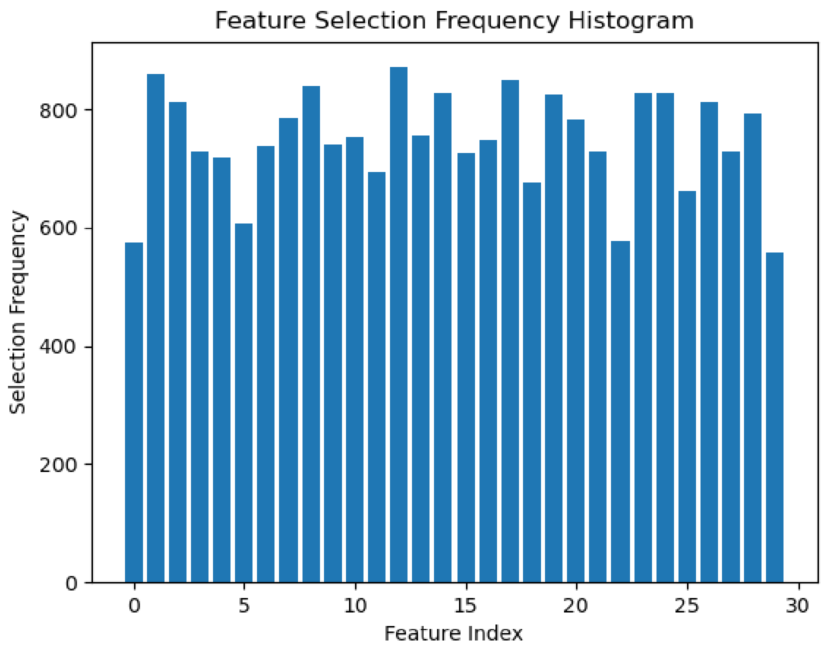

Figure 2: Bar chart of feature selection frequency across particles.

5.4 Feature Selection Frequency

The feature selection frequency plot (Figure 2) reveals that the distribution of feature selections is nearly uniform, with slight fluctuations in the selection frequency of individual features. This indicates that the PSO algorithm is efficiently exploring the feature set space, taking most features into consideration for the training, this might also suggest that no one feature is significantly more dominant than the others in influencing the model decisions.

5.5 Comparison of Accuracy with Different Swarm Sizes and Iteration Counts

The table below shows the performance of the PSO algorithm with varying swarm sizes and iteration counts. We present the best achieved accuracy for different combinations of these parameters:

Table 1: Best Accuracy Achieved for Different Combinations of Iterations and Swarm Sizes

Iterations \ Swarm Size	**10**	**25**	**50**	**100**
10	0.9766	0.9766	0.9766	0.9796
20	0.9883	0.9825	0.9942	0.9883
50	0.9883	0.9942	1.0000	1.0000
100	0.9942	0.9883	0.9942	1.0000

The **accuracy comparison table** (Table 1) demonstrates that increasing both the swarm size and the number of iterations leads to higher classification accuracy. The highest accuracy (1.0000) was achieved when the swarm size was 50 or 100, and the number of iterations was set to 50 or 100. This shows that larger swarm sizes enhance the exploration capabilities, while more iterations improve convergence towards the optimal solution which comes at the expense of having a very long running time but we can also see that our configuration gives the .

5.6 Optimization Time and Efficiency

The optimization time of 17.61 seconds indicates that PSO can provide near-optimal feature subsets in a relatively short period. The effectiveness of early stopping (which halted the algorithm after 45 iterations) also contributed to this efficiency, preventing unnecessary computation.

—

5.7 Summary of Observations

- **Early Exploration and Rapid Convergence**: The PSO algorithm showed rapid improvements in accuracy during the initial iterations, effectively exploring the feature space before converging to an optimal or near-optimal subset of features.

- **Impact of Swarm Size and Iterations**: Larger swarm sizes and more iterations consistently led to better performance, with 50 or more particles and at least 50 iterations achieving near-perfect classification accuracy.

- **Optimization Time**: The process completed in under 20 seconds, making PSO a relatively fast method for feature selection, especially with early stopping incorporated.

By restructuring the results and analysis section with comments on each figure and table, we now have a clear narrative that explains the significance of each visualization, the relationship between the swarm size, iterations, and classifier performance, and the efficiency of the PSO algorithm.

6 Challenges

Sub-optimal Convergence : BPSO is prone to converging too quickly to local optima and the search space is saturated with plateaus occurring in a huge variety of subsets which makes convergences to true optima less and less likely.

Computational Time : In a world where decisions are taken in split seconds, the algorithm running time is impacted by the major time consuming factor in the optimization pipeline which is evaluating the fitness function, so better implementations which could support parallelism should be considered.

Parameter Sensitivity : PSO is notably sensitive to its parameter settings, which significantly impacts its performance in convergence speed and optimality. The main parameters within PSO, such as the inertia weight, acceleration coefficients, and the total number of particles, play critical roles and therefore selecting them is a non trivial task which could also be done through various search mechanisms to avoid all aforementioned problems [4].

7 Conclusion

In this project, we tackled the problem of feature selection for classification tasks using Binary Particle Swarm Optimization (BPSO). By formulating the task as a combinatorial optimization problem, we aimed to find the most informative subset of features from the Breast Cancer Wisconsin Diagnostic Dataset to maximize classification accuracy while minimizing model complexity.

Our results demonstrate that BPSO is an effective and computationally efficient technique for feature selection. It achieved high classification accuracy (up to 99.42%) within a reasonable time frame and showed strong convergence behavior. We observed that larger swarm sizes and iteration counts led to improved results (100%), though with increased computational cost.

However, we also encountered several challenges, including sensitivity to the considerable number of hyperparameters, risk of premature convergence, and the high computational cost of evaluating the fitness function. These highlight opportunities for future work, such as:

- Using adaptive parameter tuning methods to improve convergence.
- Leveraging parallel computing to reduce runtime.
- Exploring alternative fitness functions and penalty designs that better balance accuracy and sparsity of feature selection.

Overall, our study reaffirms the potential of metaheuristic algorithms like PSO in solving high-dimensional optimization problems, especially in the context of feature selection in machine learning pipelines.

Appendix

Code Snippet: Velocity and Position Update

```
velocities[i][j] = (w * velocities[i][j] +
                    c1 * r1 * (pbest[i][j] - swarm[i][j]) +
                    c2 * r2 * (gbest[j] - swarm[i][j]))
swarm[i][j] = 1 if random.random() < sigmoid(velocities[i][j]) else 0
```

Code Snippet: Minimization Objective Function

```
acc = accuracy_score(y_test, predictions)
penalty = 0.001 * curr_features / all_features * (curr_features > 17)
score = 1 - acc + penalty
```

Full Python Code Repository

GitHub Repository Link

References

[1] Manuel Blum et al. Np-complete problems for networks of processors. *Journal Placeholder*, 1:1–10, 1992.

[2] S. Cheng, H. Lu, X. Lei, and Y. Shi. A quarter century of particle swarm optimization. *Complex Amp; Intelligent Systems*, 4:227–239, 2018.

[3] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[4] A. Mostafa, O. E. Nahas, and M. Mekky. An adaptive particle swarm based compressive sensing technique. *Menoufia Journal of Electronic Engineering Research*, 31:100–106, 2022.

[5] Gonçalo Pereira. Particle swarm optimization. 05 2011.

[6] S. Sengupta, S. Basak, and R. Peters. Particle swarm optimization: a survey of historical and recent developments with hybridization perspectives. *Machine Learning and Knowledge Extraction*, 1:157–191, 2018.

[7] William Nick Street, William H. Wolberg, and Olvi L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *Electronic imaging*, 1993.

[8] I. F. M. Zain and S. Y. Shin. Distributed localization for wireless sensor networks using binary particle swarm optimization (bpso). *2014 IEEE 79th Vehicular Technology Conference (VTC Spring)*, pages 1–5, 2014.