Cómputo evolutivo para la optimización de hiperparámetros en algoritmos de Aprendizaje Automático

Jorge Ramos-Frutos

27 de junio del 2025

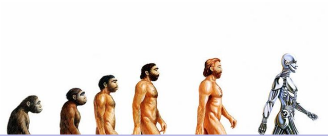
Definitions

Evolutionary Computation

Bio-inspired optimization paradigm using:

- Natural selection
- Genetic recombination
- Random mutation

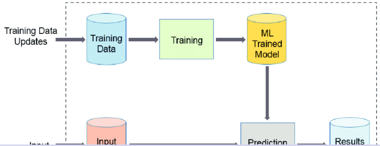
to evolve solutions iteratively.



Machine Learning

Algorithms that learn patterns from data through:

- Mathematical optimization
- Statistical generalization
- Parameter adjustment



Hyperparameter Optimization

What are Hyperparameters?

Parameters that are **not learned** during training but are **set before** the learning process begins. They control the model's architecture and learning behavior.

Some algorithms:

- XGBoost:
  - max_depth (tree complexity)
  - learning_rate (step size)
  - n_estimators (number of trees)
- Neural Networks:
  - learning rate
- SVM:
  - C (regularization)
  - kernel type
  - gamma (RBF influence)
- Random Forest:
  - n_estimators
  - max_features

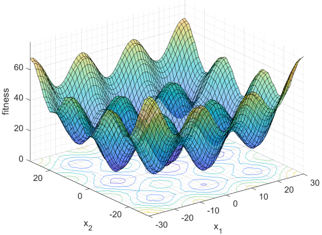
Hyperparameter Optimization

Baking the Perfect Cake

- Like perfecting a recipe:
  - Oven temperature
  - Baking time
  - Ingredients ratio
- Poor ratios $\Rightarrow$ Bad results

Optimization Techniques

- **Grid Search**: Exhaustive combinatorial
- **Random Search**: Stochastic sampling
- **Bayesian Opt.**: Gaussian processes
- **Genetic Algorithms**: Evolutionary approach (as implemented)



XGBoost: Overview

- **Ensemble learning** algorithm based on decision trees
- **Gradient Boosting**: builds models sequentially by correcting errors
- Regularized optimization to prevent overfitting
- Highly efficient and parallelizable

Objective Function in XGBoost

The general objective function:

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

where:

- l: loss function (log-loss for classification)
- Ω: regularization term
- f_k: k-th tree

For binary classification:

$$l(y_i, \hat{y}_i) = -[y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

with $p_i = \frac{1}{1+e^{-\hat{y}_i}}$

Tree Construction

XGBoost uses second-order Taylor expansion:

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{n}[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)] + \Omega(f_t)$$

where:

- $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$
- $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

The regularization term:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

with T number of leaves and w leaf scores.

Optimized Parameters

Parameters optimized by the GA and their mathematical meaning:

Parameter	Mathematical Meaning
max_depth	Maximum depth d of each tree
learning_rate	η: weight of each new tree
n_estimators	K: total number of trees
gamma	γ: minimum loss reduction
min_child_weight	w_{min}: minimum child node weight
subsample	Fraction of data to train each tree
colsample_bytree	Fraction of features for each tree

Exercise

Manual hyperparameter adjustment

Adjust XGBoost hyperparameter manually using the code given in the next address:
`https://github.com/JARF1095/Delfines-2025-U-de-G`

Basic Concepts

- Biological evolution inspiration (natural selection)
- Population algorithm (several vectors)
- Each individual contains a **fitness** that measures the solution quality
- Genetic operators:
  - **Selection**: Survival of the fittest
  - **Crossover**: Combine parent information
  - **Mutation**: Introduce variations within the individual

Mathematical Representation for Hyperparameter Optimization

In our code:

- Each individual is a 7-parameter vector:

$$\mathbf{x} = [d, \eta, n, \gamma, w, s_s, s_c]$$

  where:

  - d: max_depth (maximum tree depth)
  - η: learning_rate (learning rate)
  - n: n_estimators (number of trees)
  - γ: min_split_loss (minimum loss reduction for split)
  - w: min_child_weight (minimum child node weight)
  - s_s: subsample (fraction of samples used)
  - s_c: colsample_bytree (fraction of features used)

Fitness Function

The evaluation function in the code:

$$f(\mathbf{x}) = \text{accuracy}(M(\mathbf{x}), X_{val}, y_{val})$$

where:

- $M(\mathbf{x})$ is the XGBoost model with parameters $\mathbf{x}$
- X_{val}, y_{val} are validation data
- accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$

Implemented constraints:

- $\gamma \geq 0$
- $\eta \geq 0{,}001$
- $s_s, s_c \in [0{,}5, 1{,}0]$

Genetic Operators Used

- **Selection**: Tournament (selTournament)
- **Crossover**: Simulated Binary Crossover (cxSimulatedBinaryBounded)

$$c_1 = \frac{1}{2}[(1+\beta)p_1 + (1-\beta)p_2]$$

$$c_2 = \frac{1}{2}[(1-\beta)p_1 + (1+\beta)p_2]$$

- **Mutation**: Polynomial Bounded Mutation (mutPolynomialBounded)

$$x' = x + \delta(\bar{x} - x)$$

Optimization Strategy

- Random initial population (20 individuals)
- 15 generations
- Crossover probability: 70 %
- Mutation probability: 30 %
- Elitism (HallOfFame) preserves the best individual

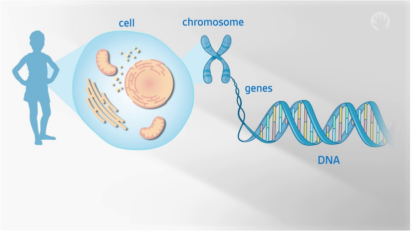


Figura: Fitness evolution across generations

Program Flow

1. Data preprocessing (normalization, encoding)
2. Train/val/test split (60 %/20 %/20 %)
3. Genetic algorithm configuration:
   - Parameter search space
   - Genetic operators
4. Hyperparameter optimization
5. Final model training
6. Evaluation and metrics

Evaluation Metrics

The code calculates multiple metrics:

- **AUC-ROC**: Area Under the ROC Curve

$$\text{AUC} = \int_0^1 TPR(FPR^{-1}(x))dx$$

- **Gini**: Related to AUC

$$Gini = 2 \times AUC - 1$$

- **Accuracy**: Overall correctness
- **Precision**: $\frac{TP}{TP+FP}$
- **Recall/Sensitivity**: $\frac{TP}{TP+FN}$
- **Specificity**: $\frac{TN}{TN+FP}$
- **F1-score**: Harmonic mean of precision and recall

Exercise

Genetic algorithm configuration

Play with the next GA parameters:

- population size
- max generations
- cross probability
- mutation probability

using the code on the next address:
`https://github.com/JARF1095/Delfines-2025-U-de-G`

Key Takeaways

- The genetic algorithm provides systematic hyperparameter search
- XGBoost is a powerful model combining multiple trees with regularization
- The combination leverages both strengths:
  - GA for parameter space exploration
  - XGBoost for precise modeling with overfitting protection
- Multiple metrics provide comprehensive performance insight

Recommendations

- For further improvement:
  - Increase GA population and generations (with computational cost)
  - Consider alternative fitness functions (e.g., AUC instead of accuracy)
  - Add more parameters to optimize (e.g., reg_lambda, reg_alpha)
- Validate with additional medical datasets
- Clinically interpret the most important variables

References

- Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System.
- Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning.
- DEAP documentation: `https://deap.readthedocs.io`
- XGBoost documentation: `https://xgboost.readthedocs.io`