Spaceship Titanic

Automated neural network architecture generation using genetic algorithms

Spaceship Titanic

- Problem statement:
  - Spaceship with 13k passengers encounters spacetime anomaly
  - Half of the passengers transported to alternate dimension
  - Challenge: predict which passengers were transported using records recovered from the spaceship's damaged computer system
- Kaggle competition
  - https://www.kaggle.com/competitions/spaceship-titanic/overview

Dataset

- 12 features (categorical and numeric)
- Predict whether passenger is transported (0 or 1)
- Training data - 8700 rows
  - Split 80/20 between train and eval
- Test data - 4300 rows

HomePlanet		CryoSleep		Cabin		Destination	
Earth	53%		true 1544 36% false 2640 62% [null] 93 2%	[null]	2%	TRAPPIST-1e	69%
Europa	23%			G/160/P	0%	55 Cancri e	20%
Other (1012)	24%			Other (4169)	97%	Other (480)	11%
Earth		True		G/3/S		TRAPPIST-1e	
Earth		False		F/4/S		TRAPPIST-1e	
Europa		True		C/0/S		55 Cancri e	
Europa		False		C/1/S		TRAPPIST-1e	
Earth		False		F/5/S		TRAPPIST-1e	
Earth		False		F/7/P		TRAPPIST-1e	

Neural Network Design

- Feed-forward network architecture
  - Depth: 1-6 dense blocks
  - Width per block: 16-512
  - Activations: ReLU / GeLU / LeakyReLU
- Feed-forward network optimization hyper parameters
  - Optimizer: Adam / AdamW
  - Learning rate, weight decay
  - Batch size, epochs cap

Genetic Algorithm Design

- Individual genome example:
  - *"num_layers": 4,*
  - *"layer_widths": [256, 128, 64, 32],*
  - *"activation": "relu",*
  - *"dropout": 0.3,*
  - *"optimizer": "adamw",*
  - *"learning_rate": 0.001,*
  - *"batch_size": 256,*
  - *...*

Genetic Algorithm Design

- Fitness functions:
  - Accuracy on eval dataset
  - Accuracy on test dataset (limited to 10 runs / day)
    - Test dataset only available on kaggle
- Convergence / stopping criteria
  - First fitness function - fitness stagnation
  - Second fitness function - iteration limit

Genetic Algorithm Design

- Selection algorithm
  - Elitism - start with 10 candidates, keep 2-4
- Mutation method
  - Micro-mutations
    - Small tweaks to existing values
    - Applies to numerical or enum parameters like learning rate, dropout or optimizer
  - Macro-mutations
    - Structural changes
    - Add/remove layer, rescale widths, activation cascade, etc
- Parameters
  - Per individual mutation rate, per gene mutation rate, macro-mutation rate