Introduction to Evolutionary Algorithms

Risto Miikkulainen

November 11, 2024

Introduction to Evolutionary Algorithms

- Evolutionary Algorithms (EAs) are optimization techniques inspired by Darwinian evolution.
- EAs mimic natural selection and survival of the fittest.
- EAs start with a population of random solutions and evolve over generations.

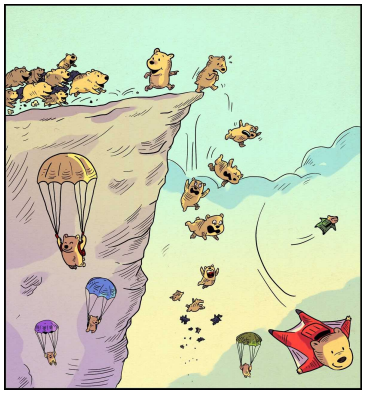
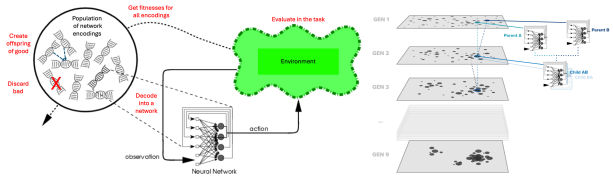


Figure: Survival of the fittest.

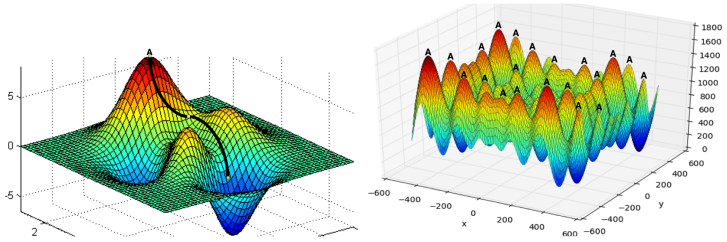
Mechanism of Evolutionary Algorithms

- Each solution is evaluated based on a fitness function.
- The best solutions (parents) are selected for the next generation.
- Variations (mutations) and recombination create new offspring.
- Over time, solutions become increasingly fit for the task.



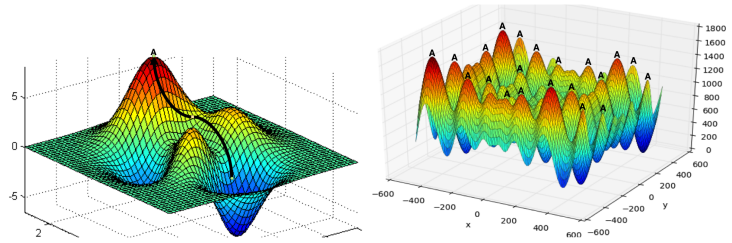
Evolutionary Algorithms vs. Gradient-based Methods

- Suitable for complex problems where no single perfect solution exists.
- Do not require a clearly defined error function, unlike backpropagation.
- Applicable in areas where traditional gradient-based optimization is challenging.



Evolutionary Algorithms vs. Reinforcement Learning

- Reinforcement Learning (RL) improves a single solution by estimating gradients
- RL struggles with sparse rewards, local optima, noisy evaluations.
- RL can refine solutions, but cannot explore widely.



Local Optimum in RL

Demo:

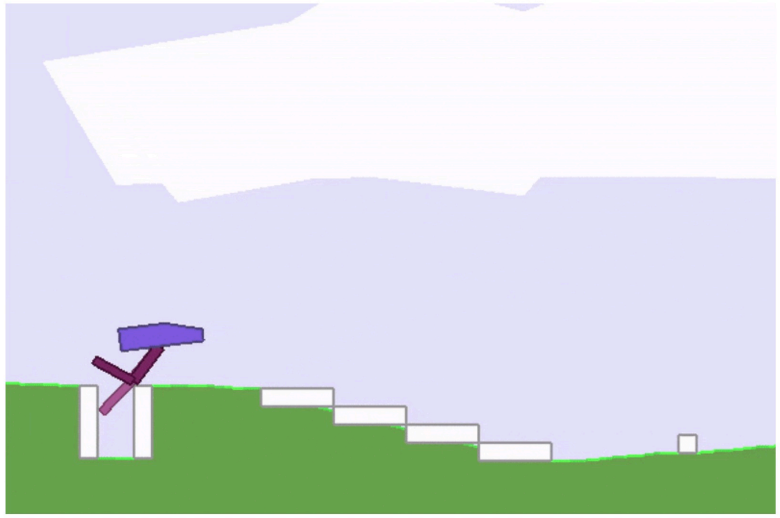
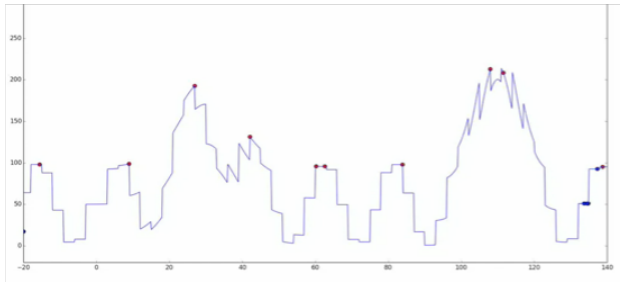


Figure: Bipedal walker agent stuck in a local optimum.

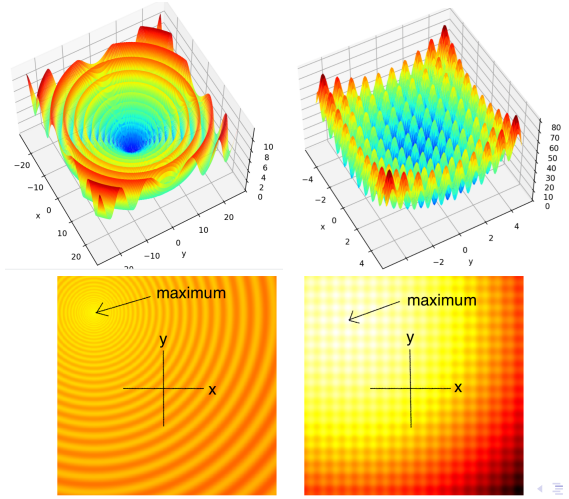
Advantages of Evolutionary Algorithms

- EAs provide an alternative: a population-based search instead of improving a single solution.
- Can be more creative.
- Can scale-up to
  - Large spaces (many possible solutions)
  - High-dimensional spaces (many knobs to adjust)
  - Deceptive spaces (hard to search)
- Many implementations



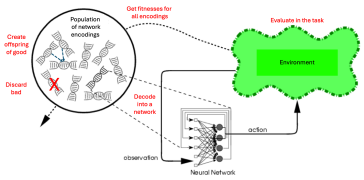
Illustrating Evolutionary Optimization

- Shaffer and Rastrigin functions in 2D
  - Highly deceptive and difficult for other methods
- Demonstrating Genetic Algorithms and Evolutionary Strategy
  - Two most common methods for evolving neural networks



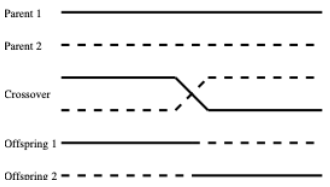
Overview of Genetic Algorithms (GAs)

- Genetic Algorithms (GAs) mimic natural selection and were introduced by John Holland in the 1970s.
- In GAs, each individual is represented as a chromosome, i.e. a string of binary or real numbers or other suitable forms.
- The initial population is generated randomly or heuristically to provide a diverse set of starting solutions.



GA Operators: Selection upon Variation

- **Selection**: Determines which individuals contribute genetic material to the next generation.
  - **Roulette Wheel Selection**: Probability-based selection based on fitness.
  - **Tournament Selection**: Choose fittest from a random group.
  - **Rank-Based Selection**: Individuals ranked by fitness, selection probabilities assigned by rank.
- **Variation**: Generate new, diverse individuals.
  - **Crossover**: Combines genetic material from two parents to create offspring.
    - Single-Point, Two-Point, and Uniform Crossover techniques.
  - **Mutation**: Introduces small random changes to maintain diversity and avoid local optima.



Simple GA Process

- Simple GA keeps only 10% of the best solutions, discarding the rest.
- New solutions are sampled by recombining parameters from surviving solutions.
- Gaussian noise is added to new solutions to maintain diversity.

Demo: Demo:

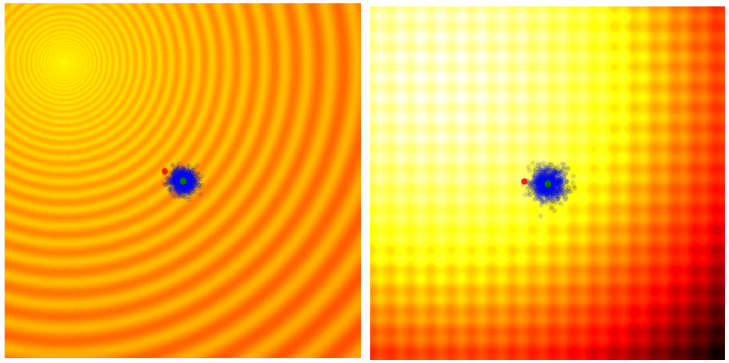


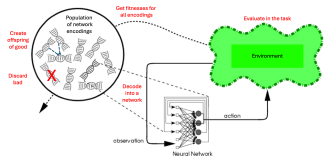
Figure: Simple GA progress over 20 steps. The green dots represent members of the elite population from the previous generation, the blue dots are the offsprings to form the set of candidate solutions, and the red dot is the best solution.

Overview of Evolution Strategy (ES)

- Evolution Strategy (ES) was developed in the 1960s and 1970s by Ingo Rechenberg and Hans-Paul Schwefel.
- ES typically operates on real-valued vectors, optimizing continuous functions.
- ES typically employs only mutation operations.
- Unlike GAs, ES focuses on continuous optimization rather than binary or symbolic representations.

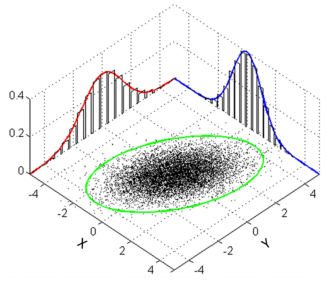
Mechanism of Evolution Strategy

- Each individual is represented by a vector of real numbers (solution parameters).
- Initial population is generated randomly or based on prior knowledge.
- Selection is deterministic, with the best individuals producing the next generation.
- **(μ, λ) Selection**: μ parents produce λ offspring, best λ selected.
- **$(\mu + \lambda)$ Selection**: Best μ individuals from both parents and offspring selected.



Variation in Evolution Strategy

- Variation primarily through mutation, adding a normally distributed random vector.
- Mutation strength (σ) controls the magnitude of perturbations.
- Crossover is less common but can combine parameter vectors from multiple parents.
- ES samples solutions from a normal distribution around a mean μ with standard deviation σ.



Simple ES Process

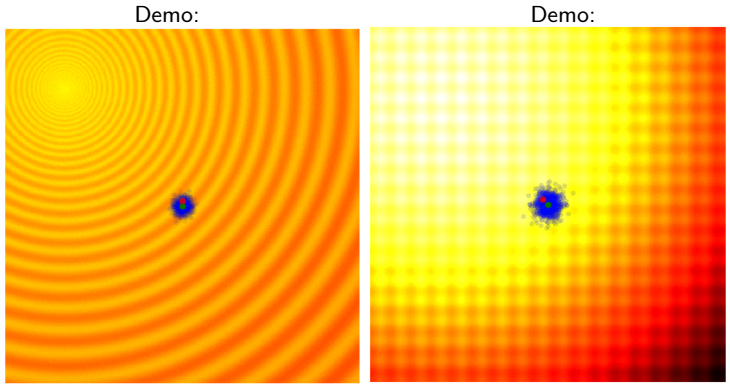
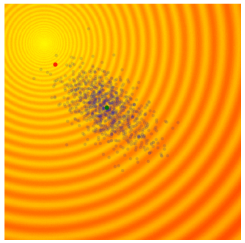


Figure: Simple ES progress over 20 steps. The green dot indicates the mean, blue dots are sampled solutions, and the red dot is the best solution.

Extension to CMA-ES

- CMA-ES addresses the limitation of fixed standard deviation in Simple ES and GA.
- It adapts the search space by adjusting the mean μ and covariance matrix, thus capturing dependencies between dimensions.
- Ideal for exploring large search spaces and fine-tuning near optimal solutions.
- Particularly useful for neuroevolution: captures how weights depend on each other.



Covariance Matrix Estimation

- To adapt the search space, CMA-ES uses the N_{best}=25% of the population to estimate the covariance matrix, and the current generation's mean, $\mu^{(g)}$.
- First calculate the new mean:

$$\mu_x^{(g+1)} = \frac{1}{N_{best}} \sum_{i=1}^{N_{best}} x_i, \qquad \mu_y^{(g+1)} = \frac{1}{N_{best}} \sum_{i=1}^{N_{best}} y_i$$

- Then the 2D covariance matrix entries:

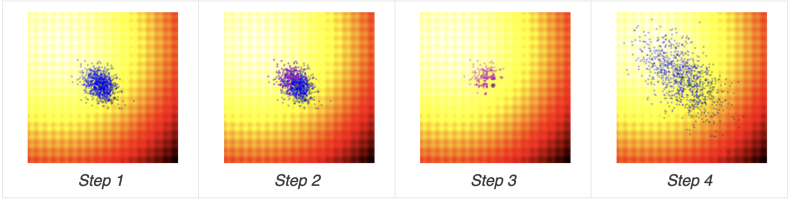
$$\sigma_x^{2,(g+1)} = \frac{1}{N_{best}} \sum_{i=1}^{N_{best}} (x_i - \mu_x^{(g)})^2,$$

$$\sigma_y^{2,(g+1)} = \frac{1}{N_{best}} \sum_{i=1}^{N_{best}} (y_i - \mu_y^{(g)})^2$$

$$\sigma_{xy}^{(g+1)} = \frac{1}{N_{best}} \sum_{i=1}^{N_{best}} (x_i - \mu_x^{(g)})(y_i - \mu_y^{(g)})$$

CMA-ES Step-by-Step Illustration

1. Calculate fitness scores for each solution.
2. Select the best 25% of the population.
3. Use these best solutions to update the covariance matrix and mean for the next generation.
4. Sample new candidate solutions from the updated distribution.



CMA-ES Process

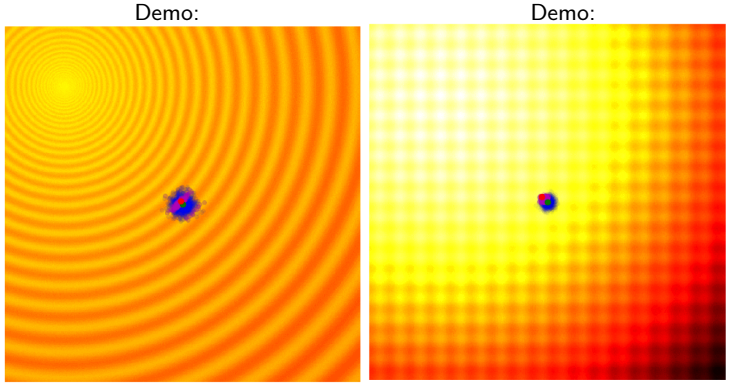


Figure: CMA-ES adapts both mean and covariance matrix over time, allowing the search process to explore broadly or refine narrowly as needed.

Conclusion on Basic Evolutionary Algorithms

- EAs are powerful, adaptive algorithms for nonlinear optimization.
- Population-based search allows for extensive exploration, making it possible to find surprising and creative solutions.
- It also makes it possible to scale up optimization to high dimensions, high dimensionality, and deceptive search spaces.
- Many implementations:
  - Genetic algorithms (GA), evolution strategies (ES,CMA-ES), genetic programming (GP), estimation of distribution algorithms (EDA), particle-swarm optimization (PSO), differential evolution (DE), evolutionary programming (EP), genetic cartesian programming (GCP)...
  - GAs and CMA-ES most commonly used for neuroevolution.

They Can Get Over Local Minima!

Demo:

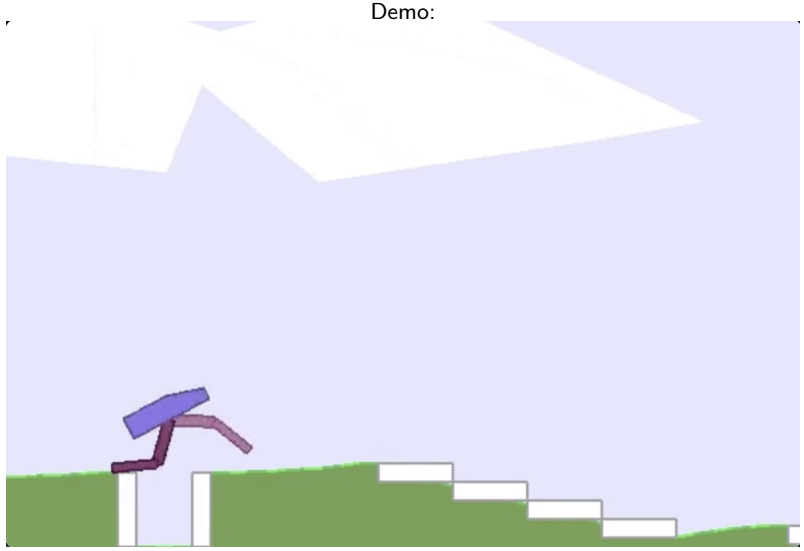


Figure: Bipedal walker agent evolved with CMA-ES.

Introduction to Exercises

- Hands-on experience is the best way to understand EAs.
- First exercises on Numpy and EvoJAX on Google Colab.
- Different algorithms can be easily evaluated, e.g.:

```
#solver = es.SimpleGA(...)
#solver = es.SimpleES(...)
solver = es.CMAES(...)

while True:
   solutions = solver.ask()
   fitness_list = np.zeros(solver.popsize)

   for i in range(solver.popsize):
     fitness_list[i] = evaluate(solutions[i])

   solver.tell(fitness_list)
   result = solver.result()

   if result[1] > MY_REQUIRED_FITNESS:
     break
```