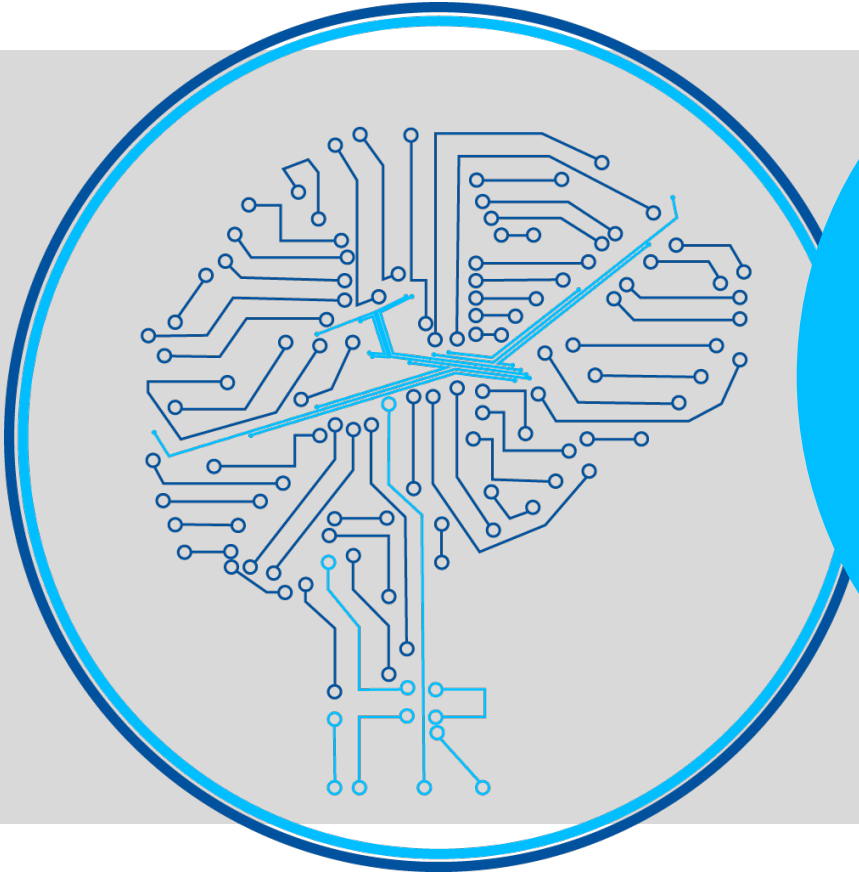
University of Stuttgart
Flight Mechanics and Controls Lab



Population Based Optimization

Fabian Schimpf

Housekeeping

Thursday, December 17th 5.30pm:

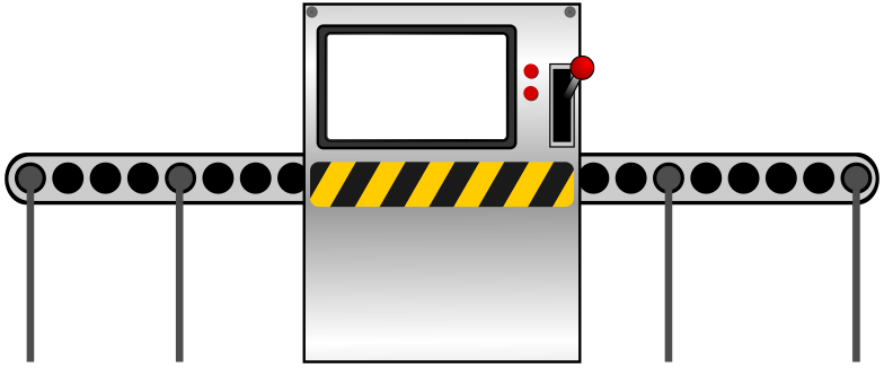
Optimierung des Primärenergiebedarfs und der Gesamtkosten (Total Cost of Ownership) von Gebäuden mit Schwarmoptimierung (PSO)

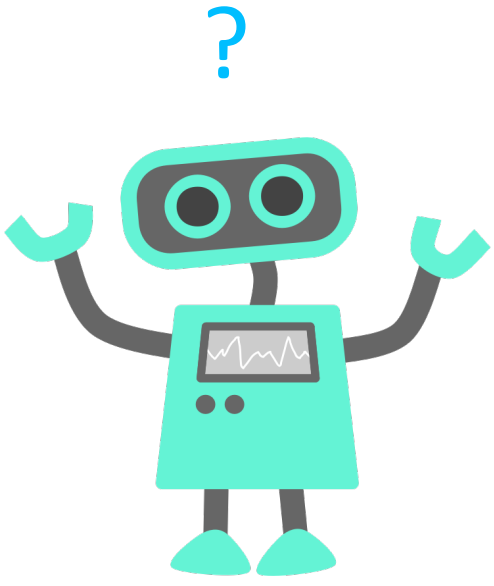
Ralf Wagner, CTO LTG AG

Introduction to Optimization

Developing Intuitions

- Given a complex machine (like an engine) imagine you should find the set of lever or knob positions to derive an optimal performance (like minimum fuel consumption or optimal through put).





Introduction to optimization

Terminology

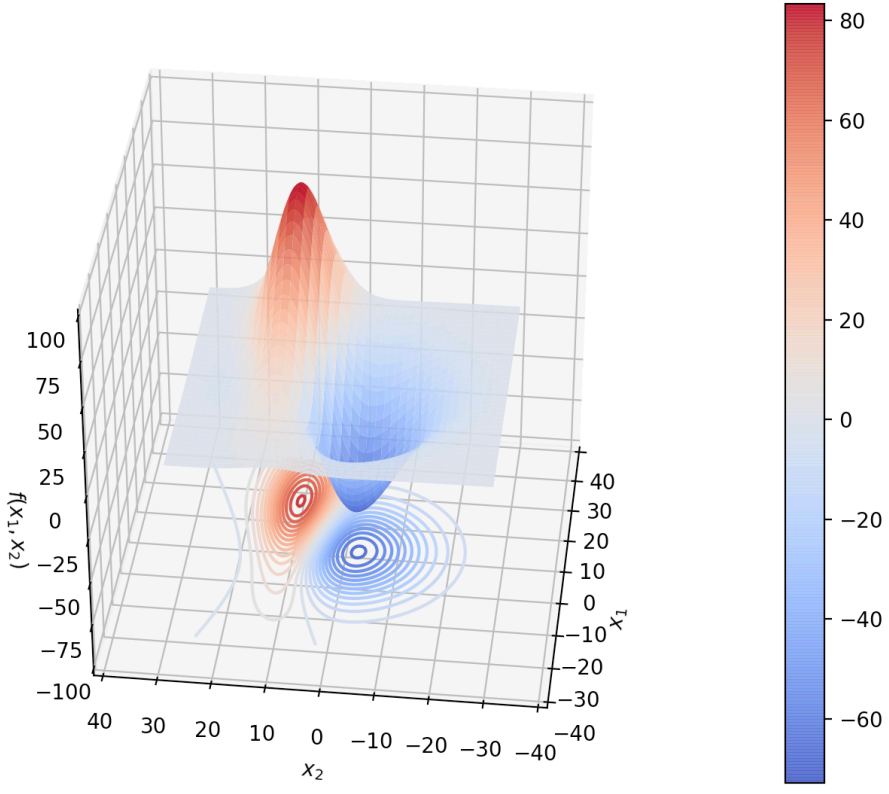
- Usually defined as the minimization of a given real valued **objective function** f, with respect to a **parameter vector** x under a set of **constraints** g and h.
  - Definition allows for standardized algorithms to solve these problems
  - Maximization can be reframed by multiplying the objective function with -1
  - In ML the parameter vector is often denoted as Θ

$$\begin{aligned} \underset{\vec{x}}{\text{minimize}} \quad & f(\vec{x}) \\ \text{subject to} \quad & g_i(\vec{x}) \leq 0, \text{for } i = 1, ..., n \\ & h_j(\vec{x}) = 0, \text{for } i = 1, ..., m \end{aligned}$$

Introduction to optimization

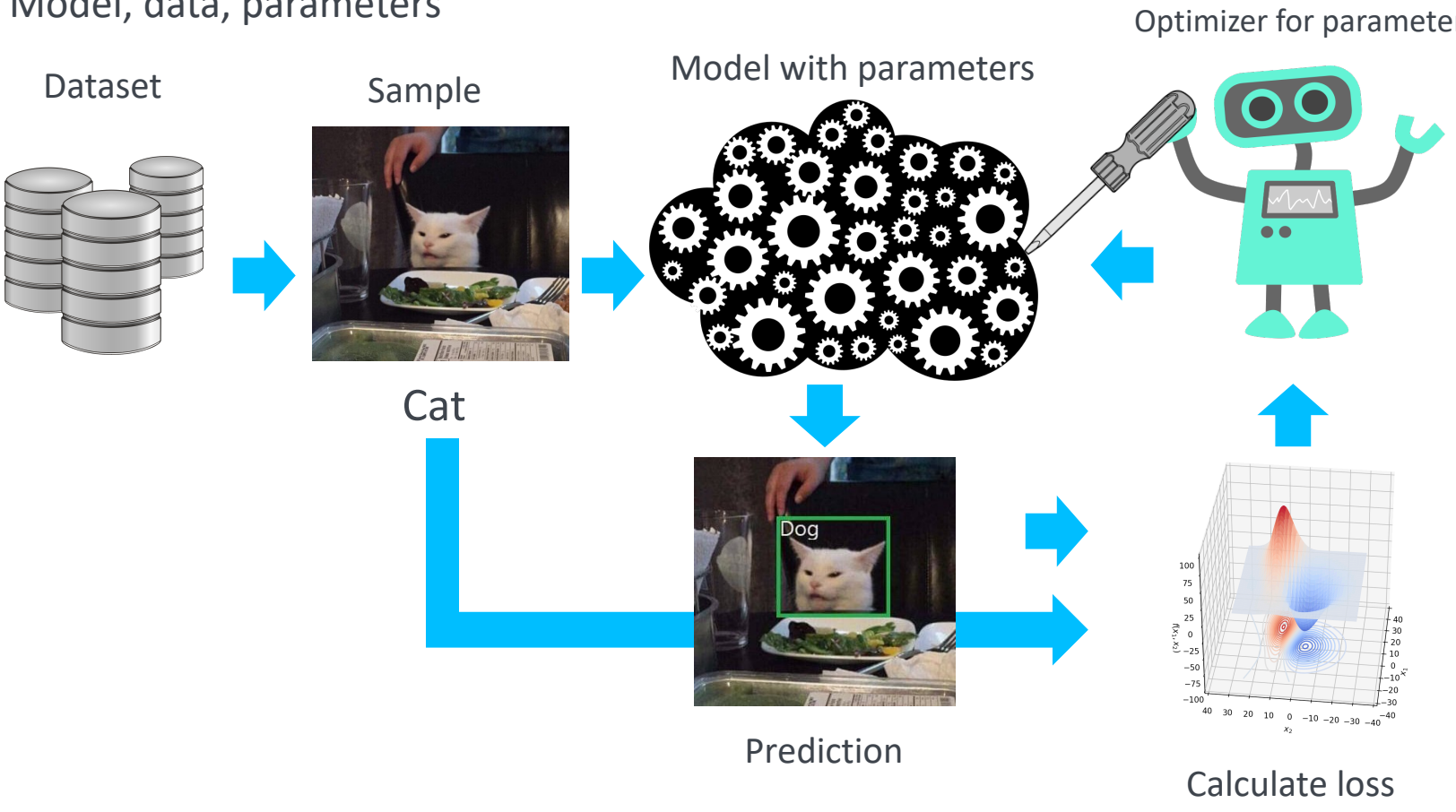
Loss and loss landscapes

- **Loss** is commonly used to refer to the objective function
- A **loss landscape** is a representation of the objective function
- Common representations are
  - Surface plots
  - Contour plots
- Contour plots can be understood as the interception of horizontal plane with surface plots of the objective function



Introduction to Optimization

Model, data, parameters



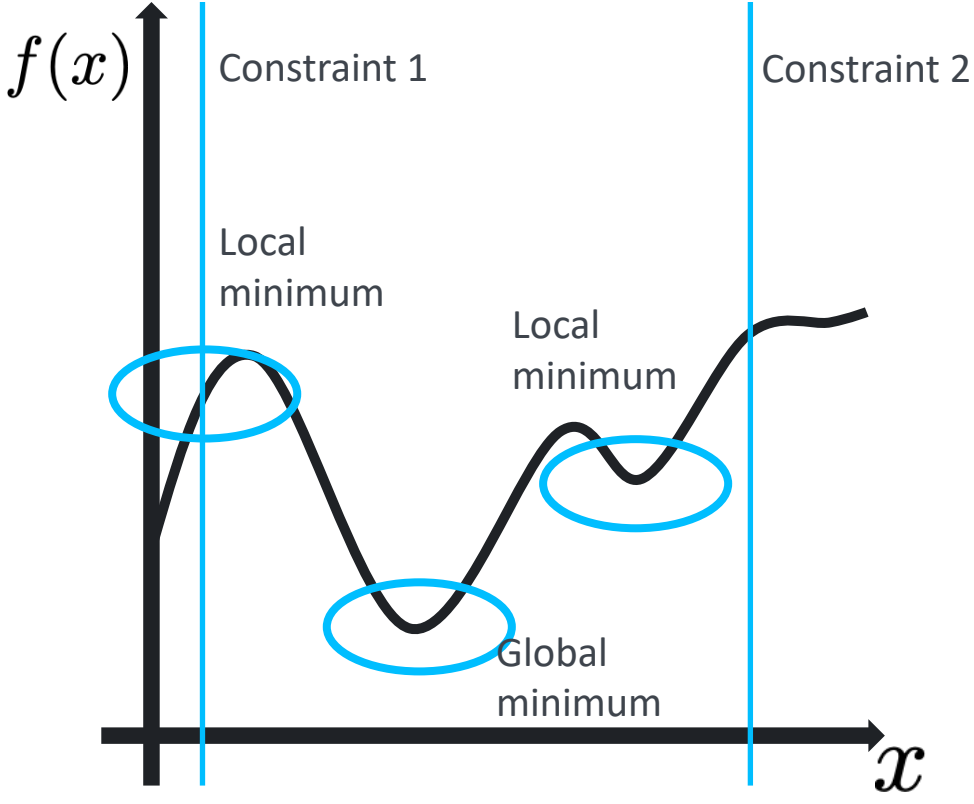
Introduction to optimization

Difference between search and optimization

- Number of atoms in the observable universe: 10^{78} to 10^{82}
- **Combinatorial explosion**
  - Picking a headquarter in one of 50 cities is mathematically feasible
  - Finding the quickest connection between 50 cities has about as many permutations as there are atoms in the universe (50!)
- **Curse of dimensionality**
  - 40 parameters with 100 points has more permutations than there are atoms in the universe – neural networks have up to billions of parameters
  - It is clear that search does not scale to some problems engineers care about

Introduction to optimization

Local vs. global optima

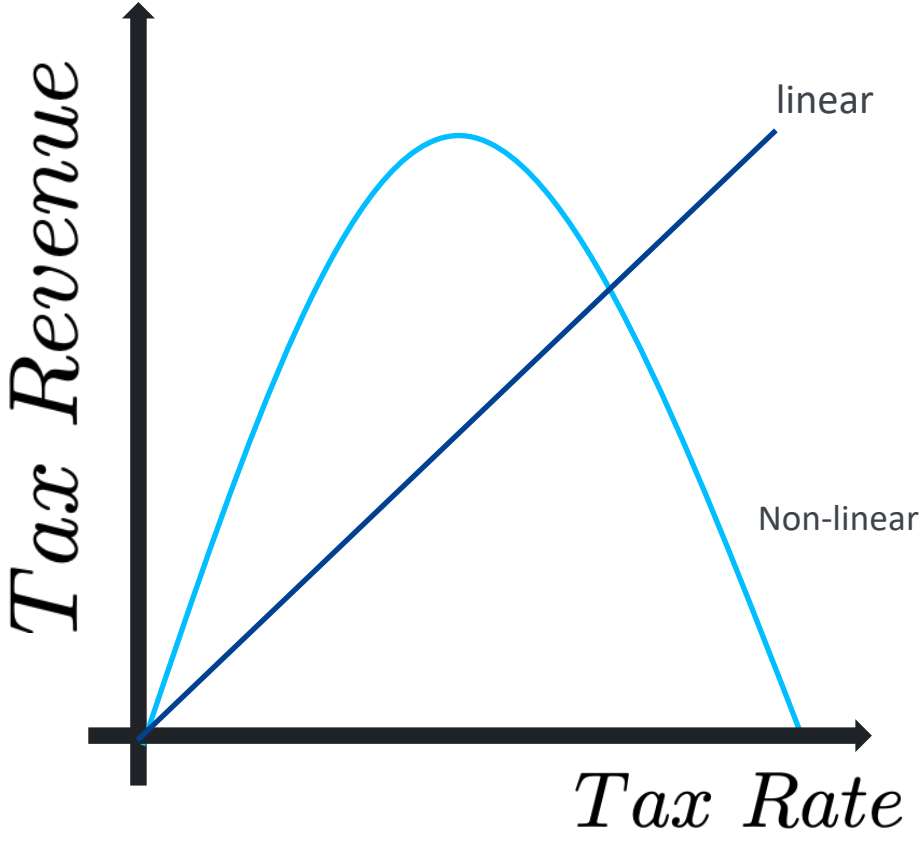


A point is considered to be an optimum if it

- Is optimal compared to all feasible (i.e. satisfying all constraints) points in a neighborhood → **local optimum**
- Is optimal compared to all feasible (i.e. satisfying all constraints) points → **global optimum**

Introduction to optimization

Linear vs. nonlinear optimization



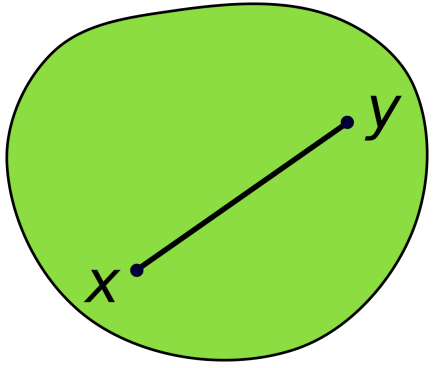
- Linear optimization
  - Local optima are global optima
  - Optimum always at constraints
- Non-linear optimization
  - Optima can be in interior
  - Optima can be at constraints
  - Local optima are not necessarily global optima
  - Update towards optimum depends strongly on where the update starts
  - Example: Laffer curve

Introduction to optimization

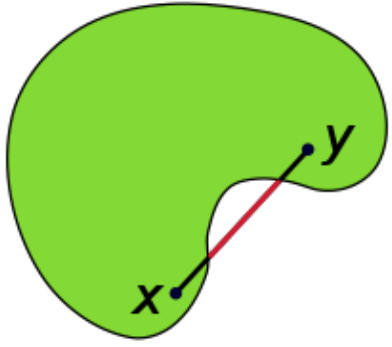
Convex vs. non-convex optimization

- Necessary condition for a function: **Hessian is positive semidefinite and defined for all values in the domain of x**
- Convex Optimization only if the function and all constraints form convex sets!
- 2nd order optimization like Newtons Method only work in convex settings
- For a convex functions local optima are global optima

Convex



Non-Convex



Introduction to Optimization

Goodhart's Law

”Any observed statistical regularity will tend to collapse once pressure is placed upon it for control purposes” — Charles Goodhart

“When a measure becomes a target, it ceases to be a good measure.” — Marilyn Strathern

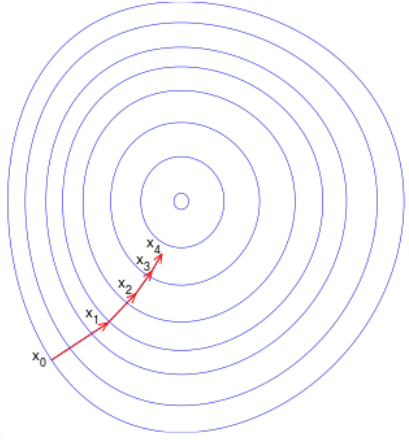
- Has been theme since Goethe ("The Sorcerer's Apprentice")
- Highlights importance of defining precisely what a system’s goal should be (which is hard)
  - Accuracy, false negative, generalization, adversarial robustness, human feedback, …
- Prevalent in reinforcement learning with ”poorly” defined reward functions

Introduction to Optimization

Main Approaches

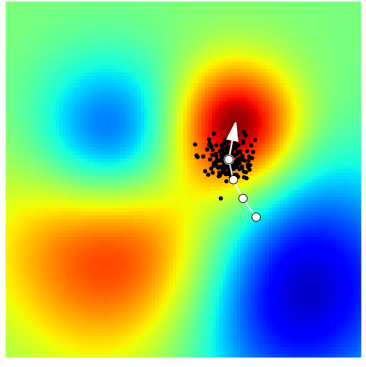
Gradient based

- Needs a differentiable function / a differentiable parameterized model
- Efficient for large parameter set
- Intuition: Ball rolling down a hill



Population / Heuristic based

- Fewer assumptions about the loss function needed
- Works well for small parameter sets
- Intuitions often derived from nature

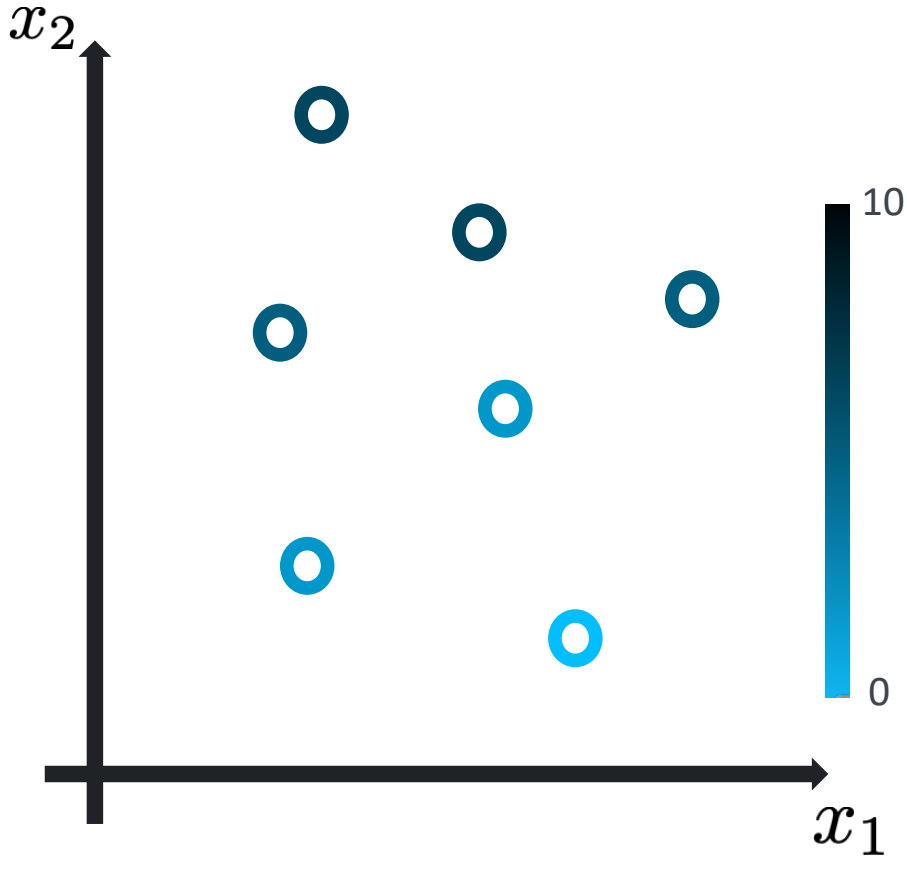


Population Based Optimization Algorithms

Introduction to Optimization

Characteristics of population based optimization algorithms

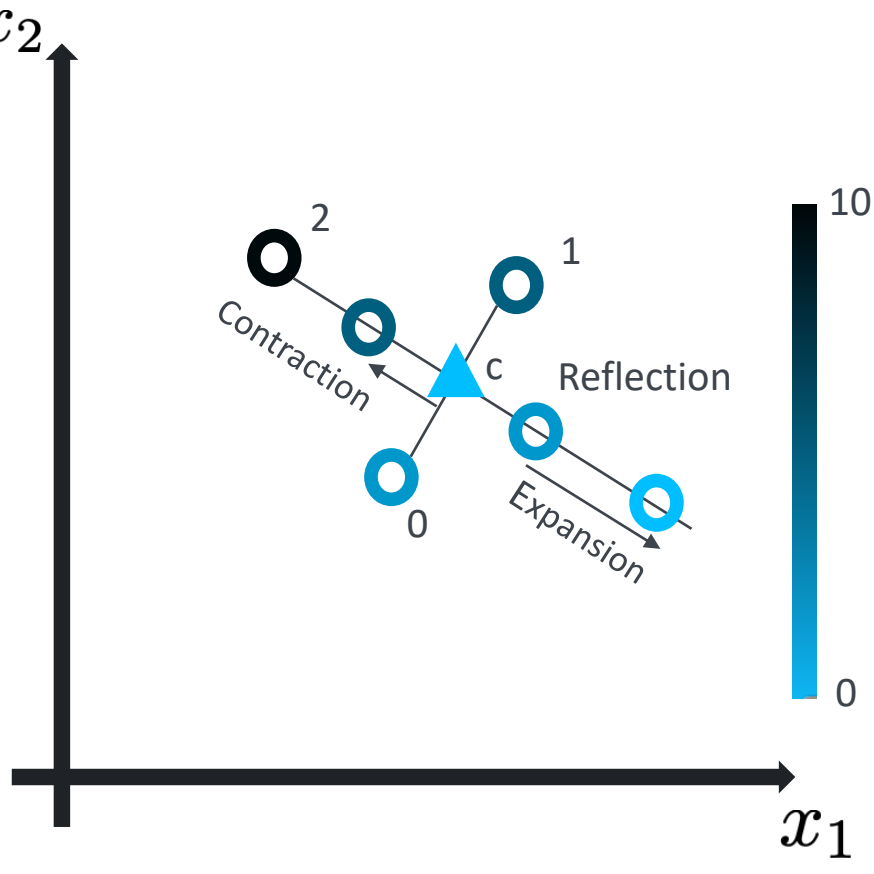
- Ability to evaluate the loss function at discrete point
- Ability to have multiple tries to inform the next set of tests
  - Set of tested points is called population
  - Updates of the population is often also called generations
- No assumptions / knowledge about the loss function or model needed



Nelder Mead

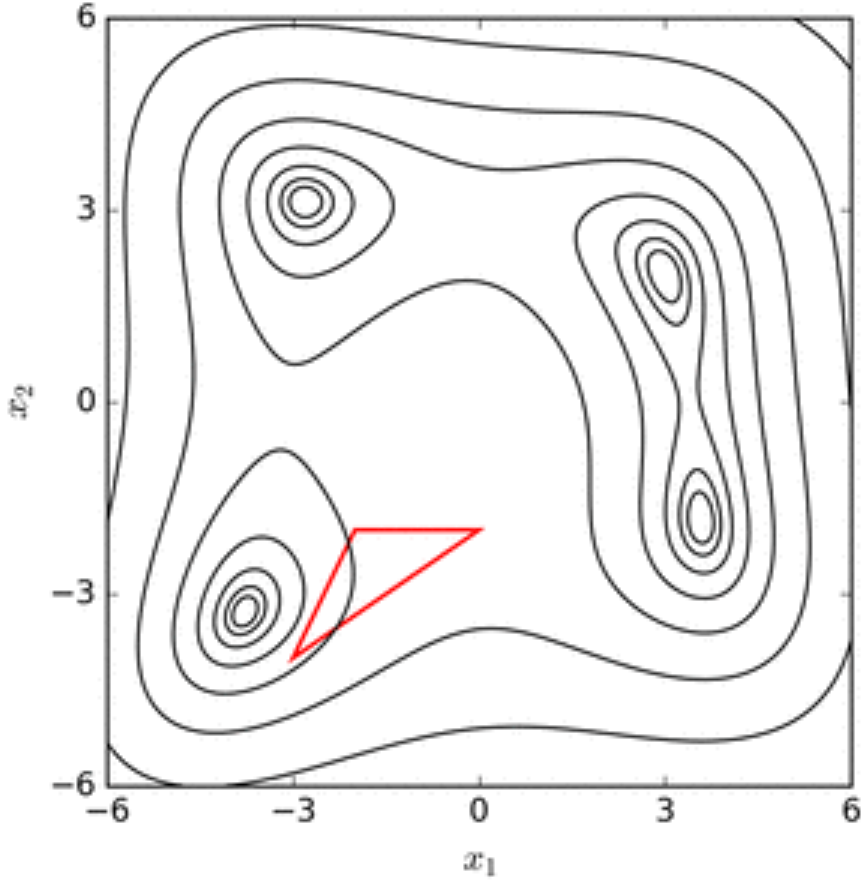
Algorithm

1. Initialize d+1 points where d is the dimensionality of the function
2. Evaluate points
3. Sort by loss (1st best nth worst)
4. Calculate centroid c of n-1 best
5. Update n-th sample along the vector $\overrightarrow{c - x_n}$
   - Reflection (outside of polyeder)
     - Expansions if new point is best so far
     - Contraction if new point is worse than n-1 best
   - Line search
6. Evaluate terminal condition
   1. Finish
   2. Jump to 2.

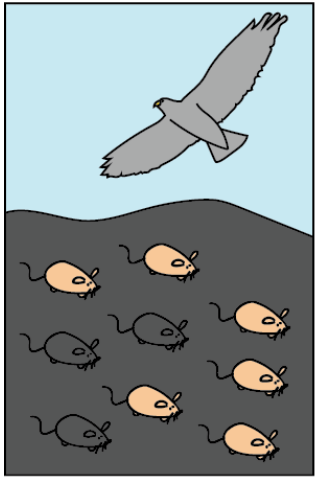


Nelder Mead

Algorithm

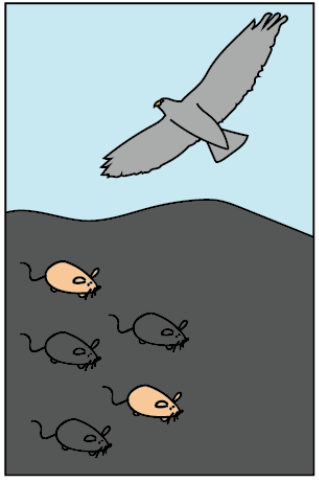


Evolution Strategies



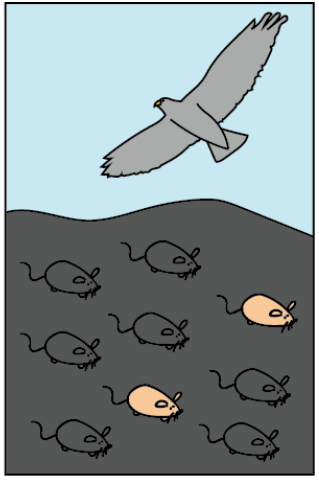
A population of mice has moved into a new area where the rocks are very dark. Due to natural genetic variation, some mice are black, while others are tan.

Some mice are eaten by birds



Tan mice are more visible to predatory birds than black mice. Thus, tan mice are eaten at higher frequency than black mice. Only the surviving mice reach reproductive age and leave offspring.

Mice reproduce, giving next generation



Because black mice had a higher chance of leaving offspring than tan mice, the next generation contains a higher fraction of black mice than the previous generation.

1+1 ES

Initialize the non-negative mutation variance σ^2
$x_0 \leftarrow$ randomly generated individual
While not(termination criterion)
    Generate a random vector r with $r_i \sim N(0, \sigma^2)$ for $i \in [1, n]$
    $x_1 \leftarrow x_0 + r$
    If x_1 is better than x_0 then
        $x_0 \leftarrow x_1$
    End if
Next generation

Adaptive 1+1 ES

Initialize the non-negative mutation variance σ^2
$x_0 \leftarrow$ randomly generated individual
While not(termination criterion)
    Generate a random vector r with $r_i \sim N(0, \sigma^2)$ for $i \in [1, n]$
    $x_1 \leftarrow x_0 + r$
    If x_1 is better than x_0 then
        $x_0 \leftarrow x_1$
    End if
    $\phi \leftarrow$ proportion of successful mutations during the past G generations
    If $\phi < 1/5$
        $\sigma \leftarrow c^2 \sigma$
    else if $\phi > 1/5$
        $\sigma \leftarrow \sigma / c^2$
    End if
Next generation

Dim(x)

Gaussian distribution

$G = \min(n, 30)$

$c = 0.817$

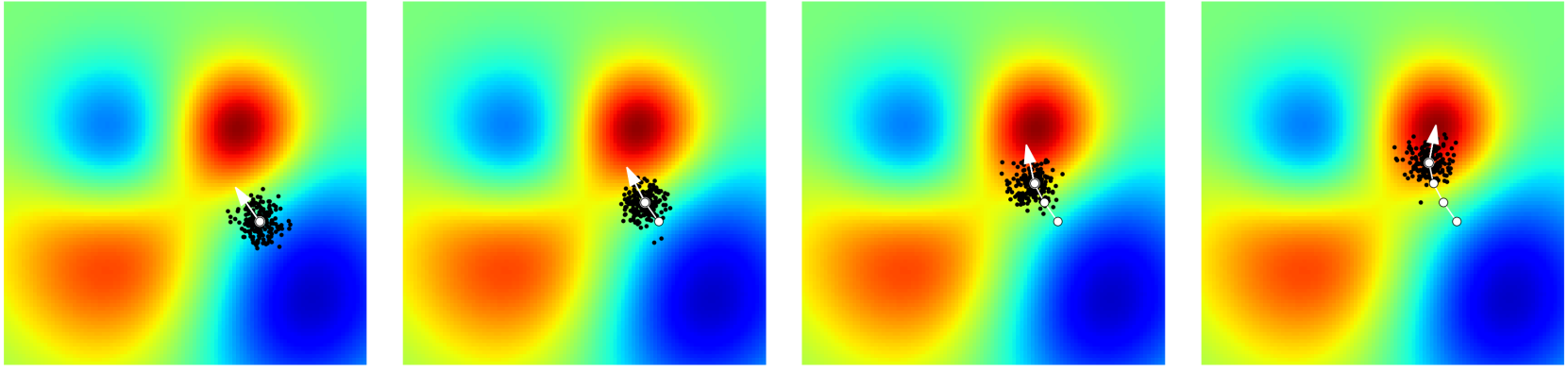
1+1 ES

Simulated Annealing

T = initial temperature > 0
$\alpha(T)$ = cooling function: $\alpha(T) \in [0, T]$ for all T
Initialize a candidate solution x_0 to minimization problem $f(x)$
While not(termination criterion)
    Generate a candidate solution x
    If $f(x) < f(x_0)$
        $x_0 \leftarrow x$
    else
        $r \leftarrow U[0, 1]$
        If $r < \exp[(f(x_0) - f(x))/T]$ then
            $x_0 \leftarrow x$
        End if
    End if
    $T \leftarrow \alpha(T)$
Next iteration

- Inspired by the annealing process in crystals
- Tuning parameters
  - Initial temperature T: relative importance of exploration
  - Cooling schedule α: Convergence rate
    - Linear cooling: $\alpha(T) = \max(T_0 - \eta k, T_{\min})$
    - Exponential cooling: $\alpha(T) = aT$
    - Inverse cooling: $\alpha(T) = T/(1 + \beta T)$
    - …
- Possible extensions
  - Separate cooling for each dimension
  - Reinitialize temperature
  - Memory of best solution

Evolution Strategies



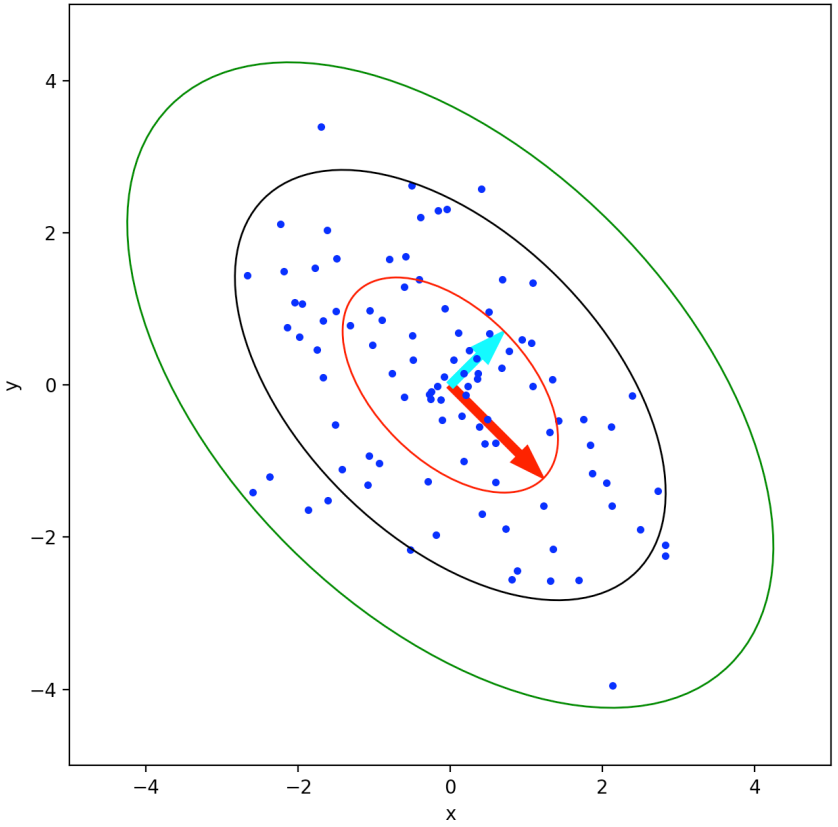
Simple Algorithm:

1. Set variance and mean
2. Create Distribution around mean
3. Evaluate population
4. Change mean to best particle
5. Evaluate terminal condition
   - Jump to 2.

Problem:

- Keeping the covariance matrix constant can be inefficient
  - Variance trades of exploration against exploitation fine coverage of the loss landscape

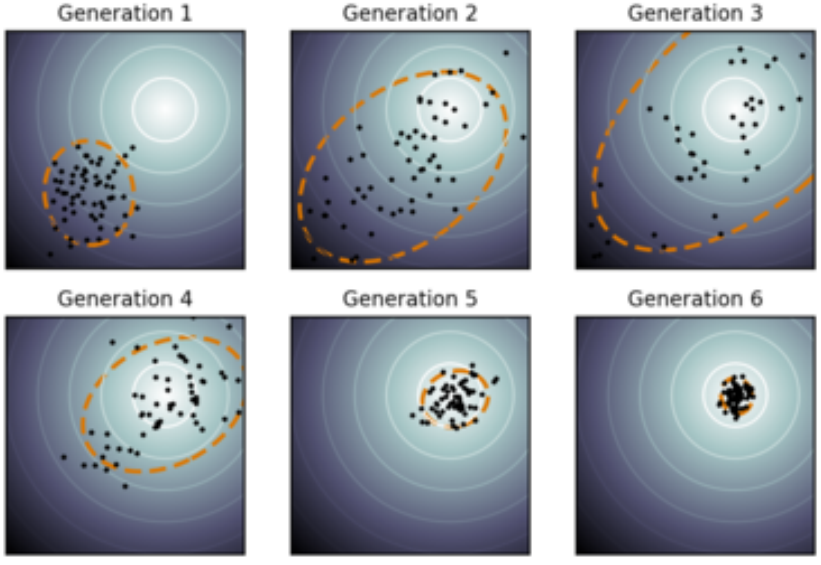
Interpretation of Covariance Matrices as Error Ellipses



- Data drawn from a distribution with
  - Mean μ
  - Covariance matrix C
- 2* sqrt(Eigenvalues) * normalized eigenvectors are width and height of the ellipse
- Note, it is possible to estimate the mean and the covariance matrix from data!

$$\sigma_{xx}^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_x)^2 \qquad \sigma_{xy}^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y)$$

Simplified CMA - ES



To do this algorithm justice we would need almost a full lecture. Therefore, a simplified version is presented. Check out "The CMA Evolution Strategy: A Tutorial" by N. Hansen for more details.

Simplified Algorithm

- Calculate loss for each particle in generation g
- Determine best 25%, their number is N_{best}
- Update the mean of the distribution

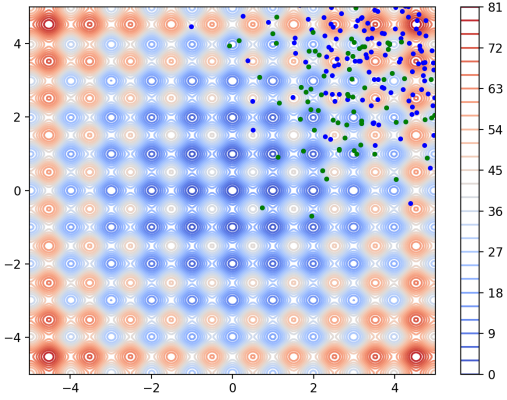
$$\mu^{g+1} = \frac{1}{N_{best}} \sum_{i=1}^{N_{best}} x_i$$

- Calculate covariance matrix C^{g+1} using the best solutions for the current generation and the mean from the last generation μ^g
- Sample new population by sampling the distribution given by μ^{g+1} and C^{g+1}

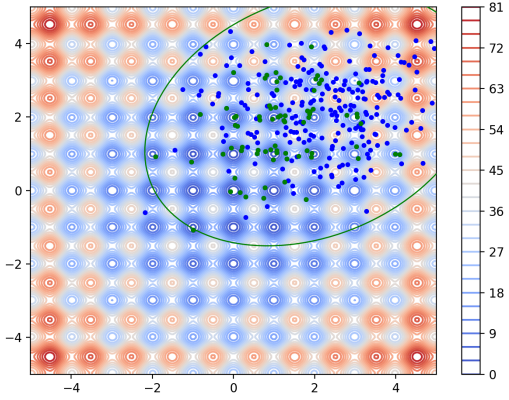
Simplified CMA ES

Results

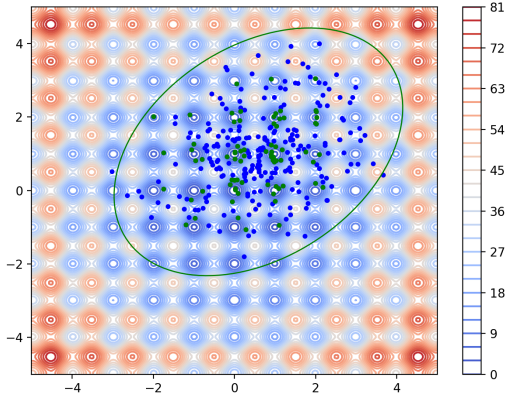
Generation 0



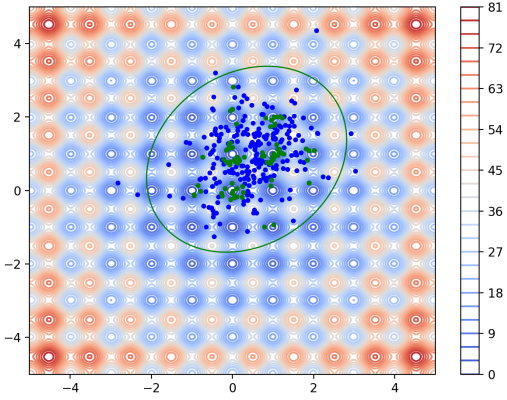
Generation 2



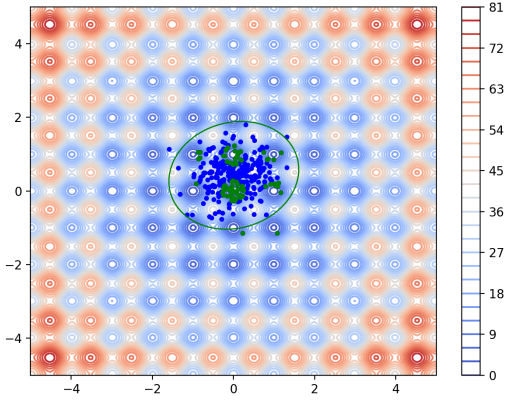
Generation 4



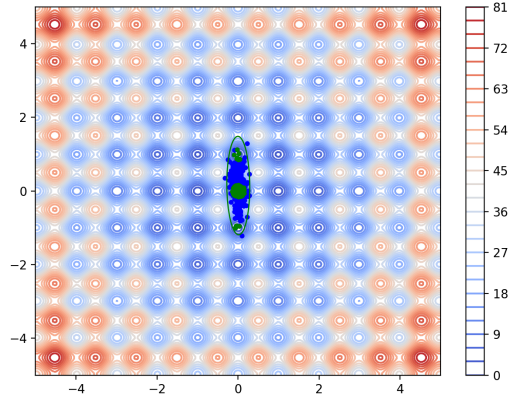
Generation 6



Generation 16



Generation 24



Particle Swarm Optimization (PSO)

Ideas and Principles

- Particles persist over generations to develop trajectories
- Each particle has a position in the parameter space, a velocity (inertia) and a remembers its best parameters so far

Each particle is also influenced by

- The global state of the swarm (influence by society)
- Its local neighborhood (influence should be correlate to “closeness”)

Particle Swarm Optimization (PSO)

Algorithm

Initialize a random population of individuals $\{x_i\}$, $i \in [1, N]$
Initialize each individual's n-element velocity vector v_i, $i \in [1, N]$
Initialize the best-so-far position of each individual: $b_i \leftarrow x_i$, $i \in [1, N]$
Define the neighborhood size $\sigma < N$
Define the maximum influence values $\phi_{1,\max}$ and $\phi_{2,\max}$
Define the maximum velocity $v_{\max}$
While not(termination criterion)
    For each individual x_i, $i \in [1, N]$
        $H_i \leftarrow \{\sigma$ nearest neighbors of $x_i\}$
        $h_i \leftarrow \arg\min_x \{f(x) : x \in H_i\}$
        Generate a random vector ϕ_1 with $\phi_1(k) \sim U[0, \phi_{1,\max}]$ for $k \in [1, n]$
        Generate a random vector ϕ_2 with $\phi_2(k) \sim U[0, \phi_{2,\max}]$ for $k \in [1, n]$
        $v_i \leftarrow v_i + \phi_1 \circ (b_i - x_i) + \phi_2 \circ (h_i - x_i)$
        If $|v_i| > v_{\max}$ then
            $v_i \leftarrow v_i v_{\max} / |v_i|$
        End if
        $x_i \leftarrow x_i + v_i$
        $b_i \leftarrow \arg\min\{f(x_i), f(b_i)\}$
    Next individual
Next generation

Size of population

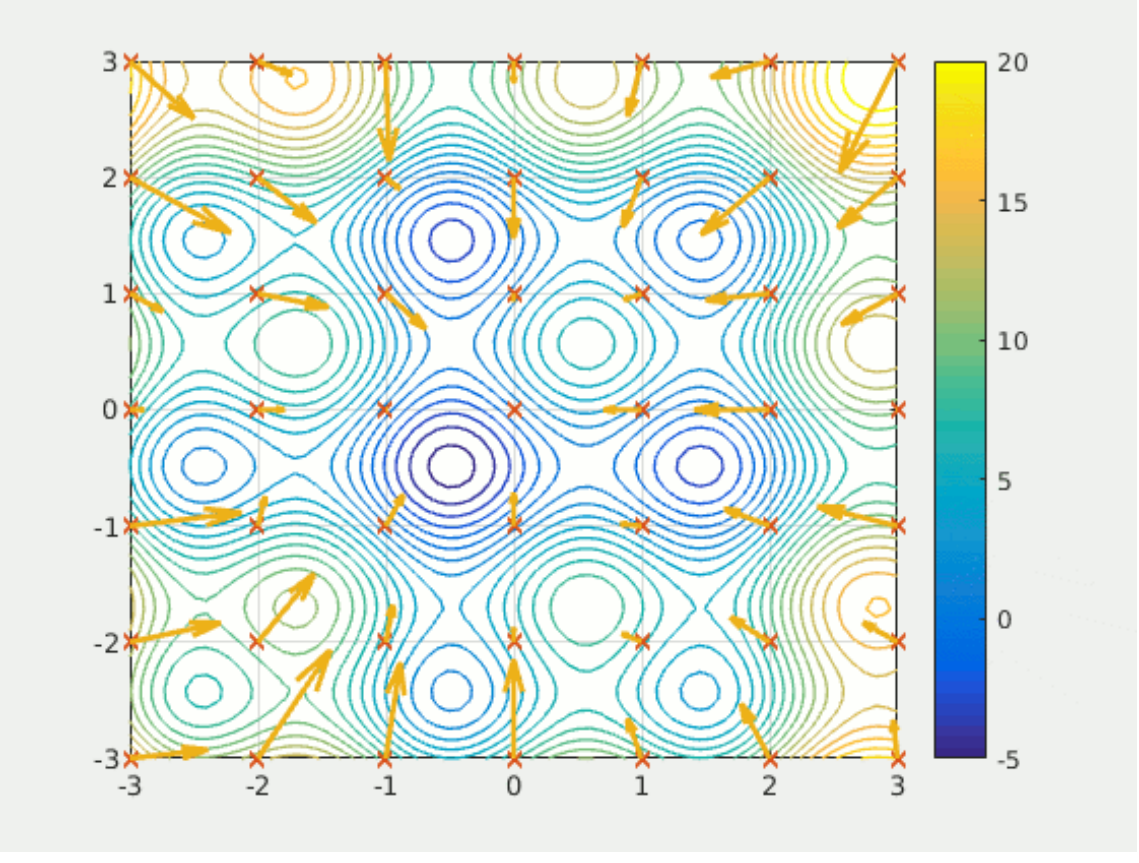
Dim(x)

Elementwise multiplication

Potential extensions:

- Inertia weighting: Decay inertia over time
- Global velocity updates
- "new PSO": Avoidance of bad behavior
- "Catfish PSO": Reset bad particles if the solution is stagnating
- Adding mutation-like operators to avoid premature convergence
- …

Particle Swarm Optimization (PSO)



Comments on population based optimization

Housekeeping

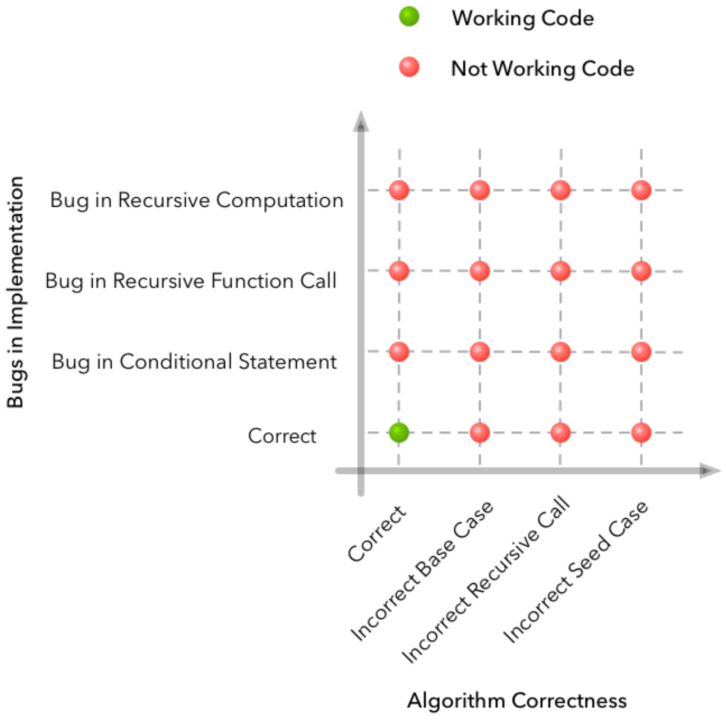
Thursday, December 17th 5.30pm:

Optimierung des Primärenergiebedarfs und der Gesamtkosten (Total Cost of Ownership) von Gebäuden mit Schwarmoptimierung (PSO)

Ralf Wagner, CTO LTG AG

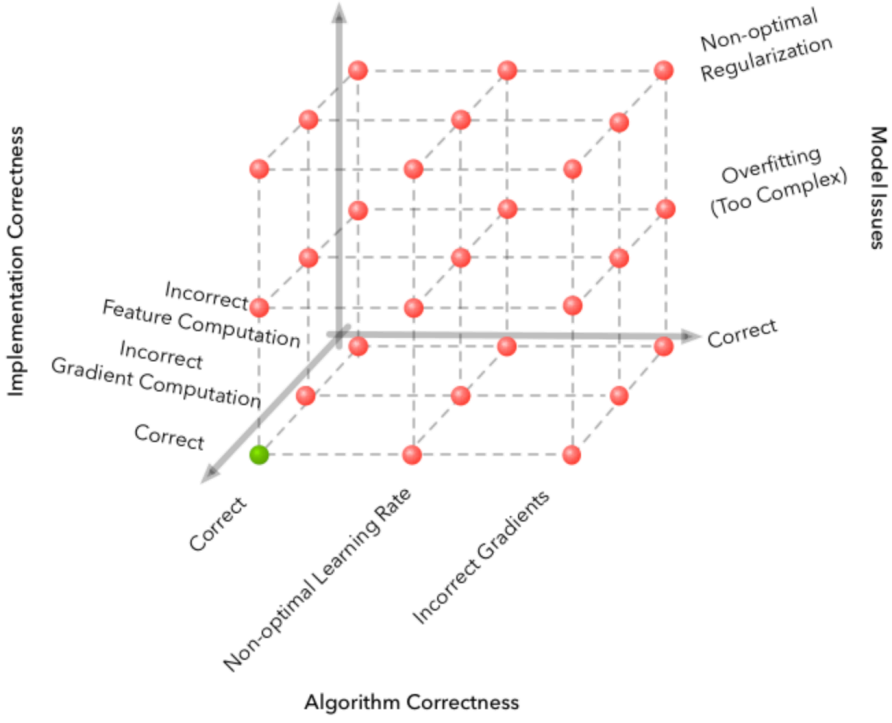
Intuitions why ML is hard

Error cases in classical software engineering



Intuitions why ML is hard

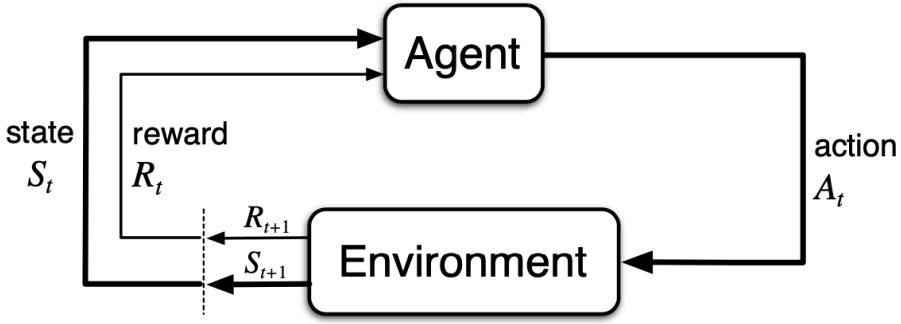
Error cases in machine learning



Intuitions why ML is hard

Error cases in machine learning

How population based optimization is used in RL



Agent aims to maximize expected discounted future cumulative reward:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Policy maps state to action:

$$\pi(a|s) = \mathbb{P}\left[a_t = a | s_t = s\right]$$

Optimizing the policy with population based optimization and –G as objective function is a simple viable solution!

Advantages:

- No need to learn value function (expected reward in given state)
- Works for sparse rewards
- Can be highly parallelized

Disadvantage

- As of now: Does not scale to big problems

Advice for debugging

- Write generic optimizers that can be tested on simplified problems
- Test your code on problems with
  - Lower dimensionality
  - Known solutions / optima
  - A linear or convex shaped loss landscape

If you can’t solve easy problems don’t progress to harder ones!

- Spend significant effort testing the code yielding your loss function
  - There are tons of tested optimization libraries available
  - Domain specific code is empirically error prone

Hyperparameter search

- Optimization loop around model and optimizer to tune optimizer settings
- Often implemented as grid search or Bayesian optimization to efficiently span the parameter space
- Can be extremely time consuming for large datasets as models have to be trained first to evaluate performance of optimizer
- Identifying key parameters to tune is crucial (curse of dimensionality)
- Subfield of ML dedicated to finding integrated approaches: AutoML

Benefits of population based optimization

- Highly applicable for Engineering problems (switches, discrete gears etc., no need to model complex systems)
- Work generally reasonably well for local minima and non-linear problems
- Usually very easy to implement
- Simplify some key problems in Reinforcement Learning
  - Time dependencies
  - Fewer hyperparameters
- Some algorithms can be highly parallelized to leverage massive computational resources

Caveats of population based optimization

- Literature is cluttered with the same concept being reintroduced under a different name
- Heuristic based approach lack mathematic proofs especially for nonlinear problems
- Tuning of algorithms can be tough for a broad range of problem settings
- Hard to determine general performance as random numbers play a key role in the many algorithms
- Inefficient for problems with known loss functions

More information on the presented topics

Lectures

- Nichtlineare Optimierung – Dr. Grimm

Further reading

- Dan Simon - Evolutionary Optimization Algorithms

Thank you!

Fabian Schimpf

e-mail Fabian.schimpf@ifr.uni-stuttgart.de
phone +49 (0) 711 685-
fax +49 (0) 711 685-

University of Stuttgart
Flight Mechanics and Controls Lab
Pfaffenwaldring 27, 70569 Stuttgart