Maastricht University

Department of Data Science and Knowledge Engineering

Autonomous Robotic Systems

Master Course

Assignment
Particle Swarm Optimization

Strategy and advice for assignments

- Do first programming in class (so you can ask questions if you get stuck early)
- Prepare for assignments at home (read lecture slides again, read material)
- Finish programming at home
- **Write simple code**
- To get graded:

  Upload code and video demonstration to CANVAS

Task

- Evaluate properties of PSO and compare PSO against gradient descent
- Use simulations to convince a customer/project leader/colleague of the pros and cons of PSO
- Find out if it makes sense to combine PSO and gradient descent

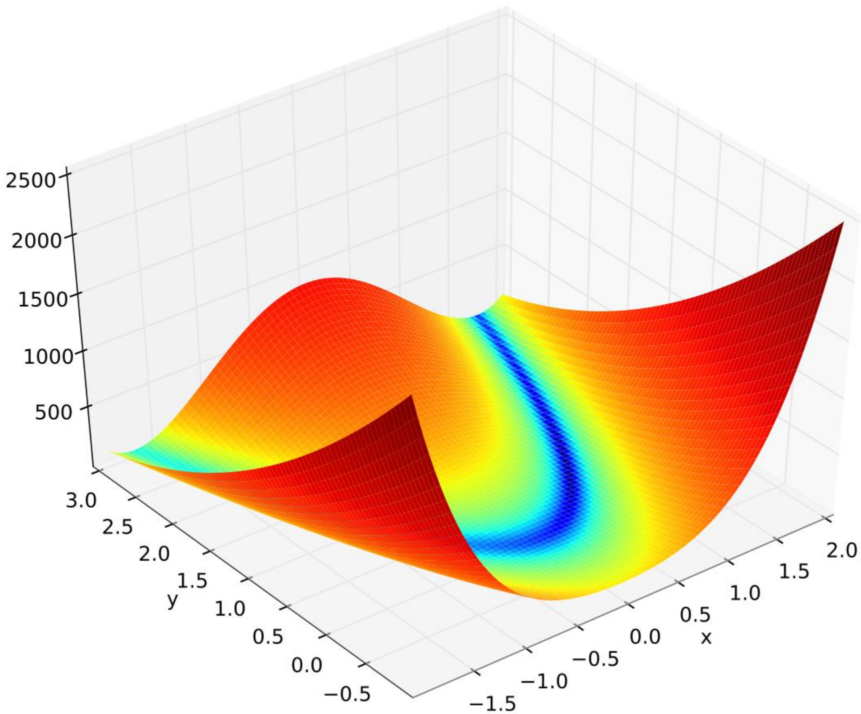
How would you do such experiments?

(You only have to implement PSO!)

Motivation to use benchmark functions

- Need test function also for future algorithms
- Final tasks of robot navigation is too time consuming to debug the algorithm
- Benchmark functions help us to understand if our algorithm works

Benchmark function: Rosenbrock

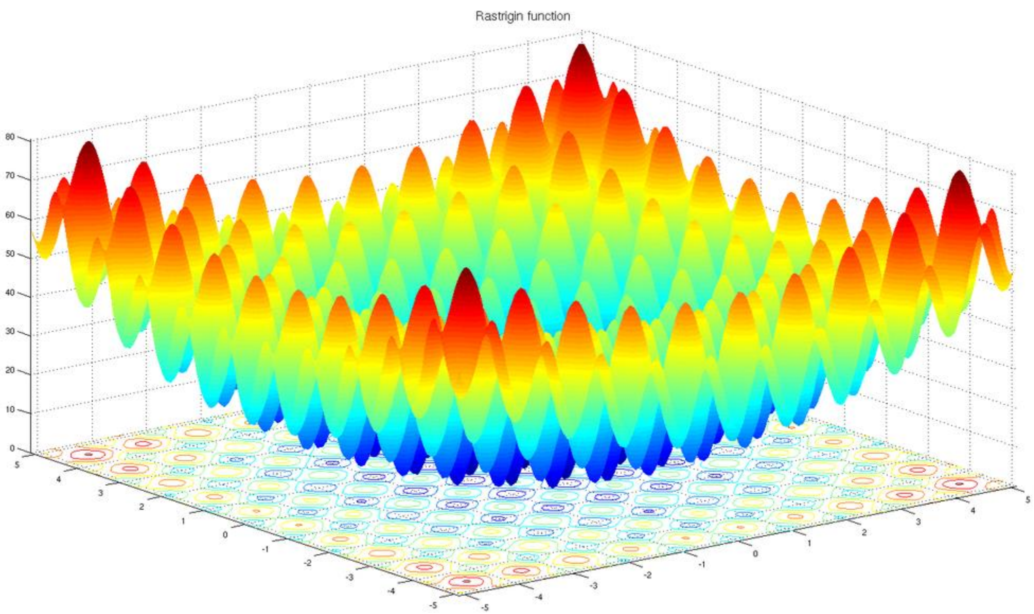


Rosenbrock function with two variables

$$f(x,y) = (a-x)^2 + b(y-x^2)^2$$

- Deep valley with known minimum minimum at (a,a^2)
- We use a=0
- Easy to find valley
- Difficult to find global minimum
- Also defined for multiple dimensions (see Wikipedia)

Benchmark function: Rastrigin



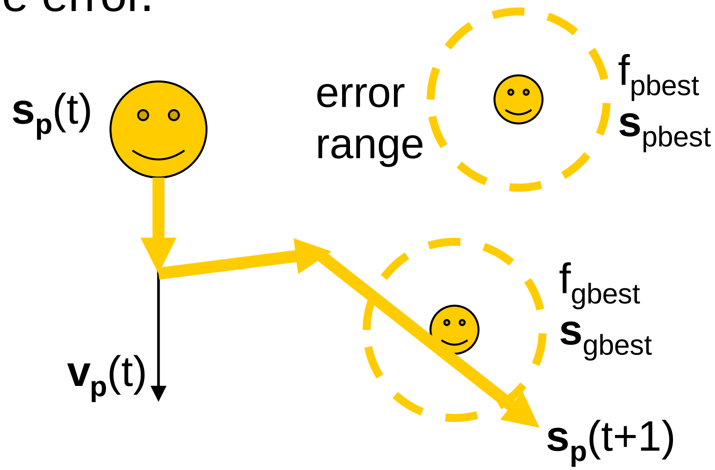
Rastrigin function with two variables

$$f_2(\boldsymbol{x}) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10cos(2\pi x_i) \right)$$

- Many local minima
- Known global minimum at **x=0**

Particle's actions

A particle computes the next position by taking into account a fraction of its current velocity **v**, the direction to its previous best location pbest, and the direction to the location of the best neighbor gbest. The movement towards other particles has some error.



$$\boldsymbol{v}_p(t+1) = \boldsymbol{a} \cdot \boldsymbol{v}_p(t) + b \cdot R \cdot \left(\boldsymbol{s}_{pbest} - \boldsymbol{s}_p(t)\right) + c \cdot R \cdot \left(\boldsymbol{s}_{gbest} - \boldsymbol{s}_p(t)\right)$$

$\boldsymbol{s}_p(t) = \boldsymbol{s}_p(t-1) + \boldsymbol{v}_p\,(t) \cdot \Delta t$ (Euler Integration, here $\Delta t = 1$)

where a, b, c are learning constants often between 0 and 1 (but see next slide)
R is a random number between 0 and 1

Parameter tuning → read papers

Parameters have been found empirically. E.g. Russell C. Eberhart suggested for best tradeoff between global and local exploration.

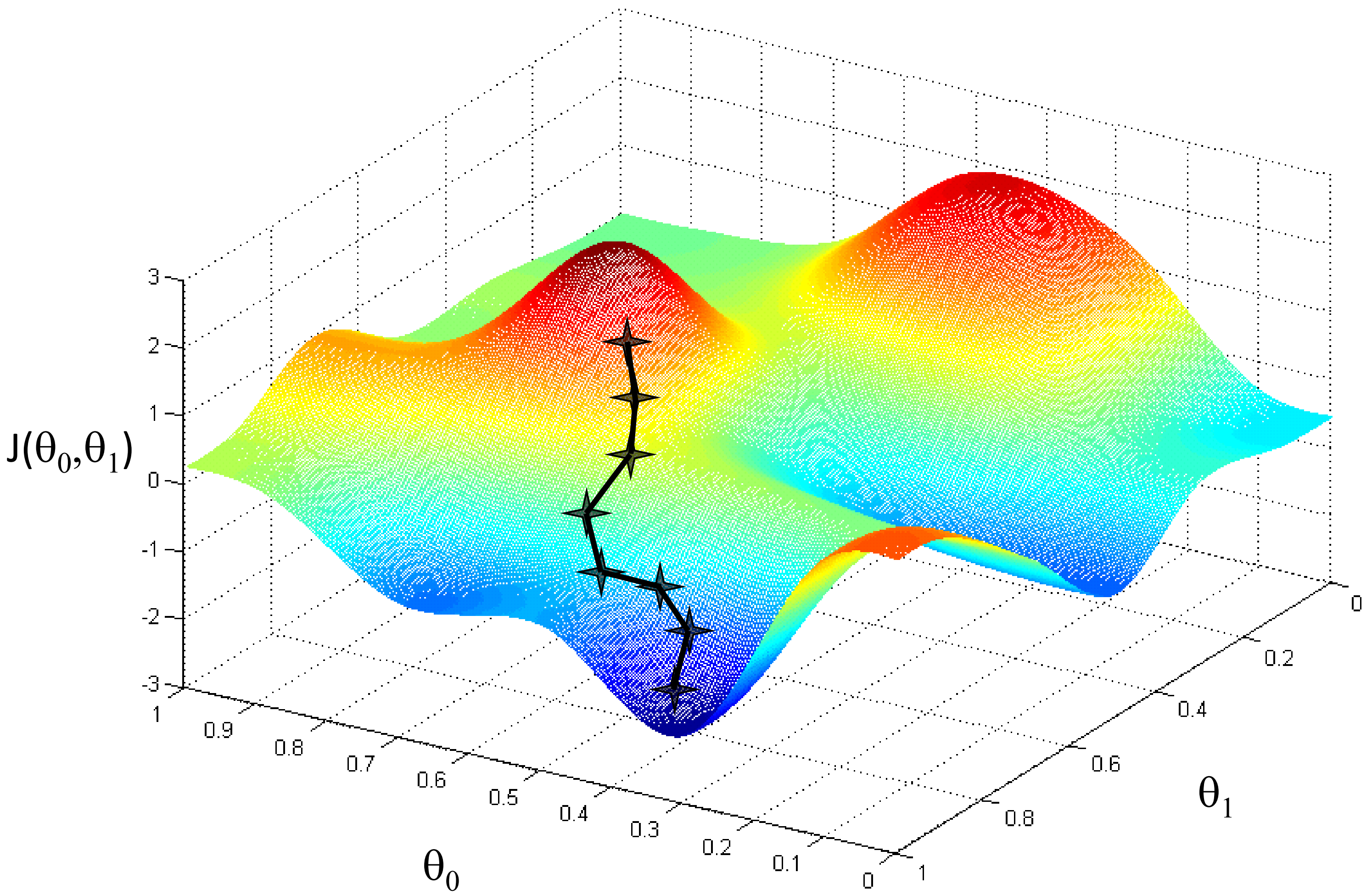
- Good approach is to reduce inertia weight *a* during run (i.e., from 0.9 to 0.4 over 1000 generations)
- Then usually set *b* and *c* to 2

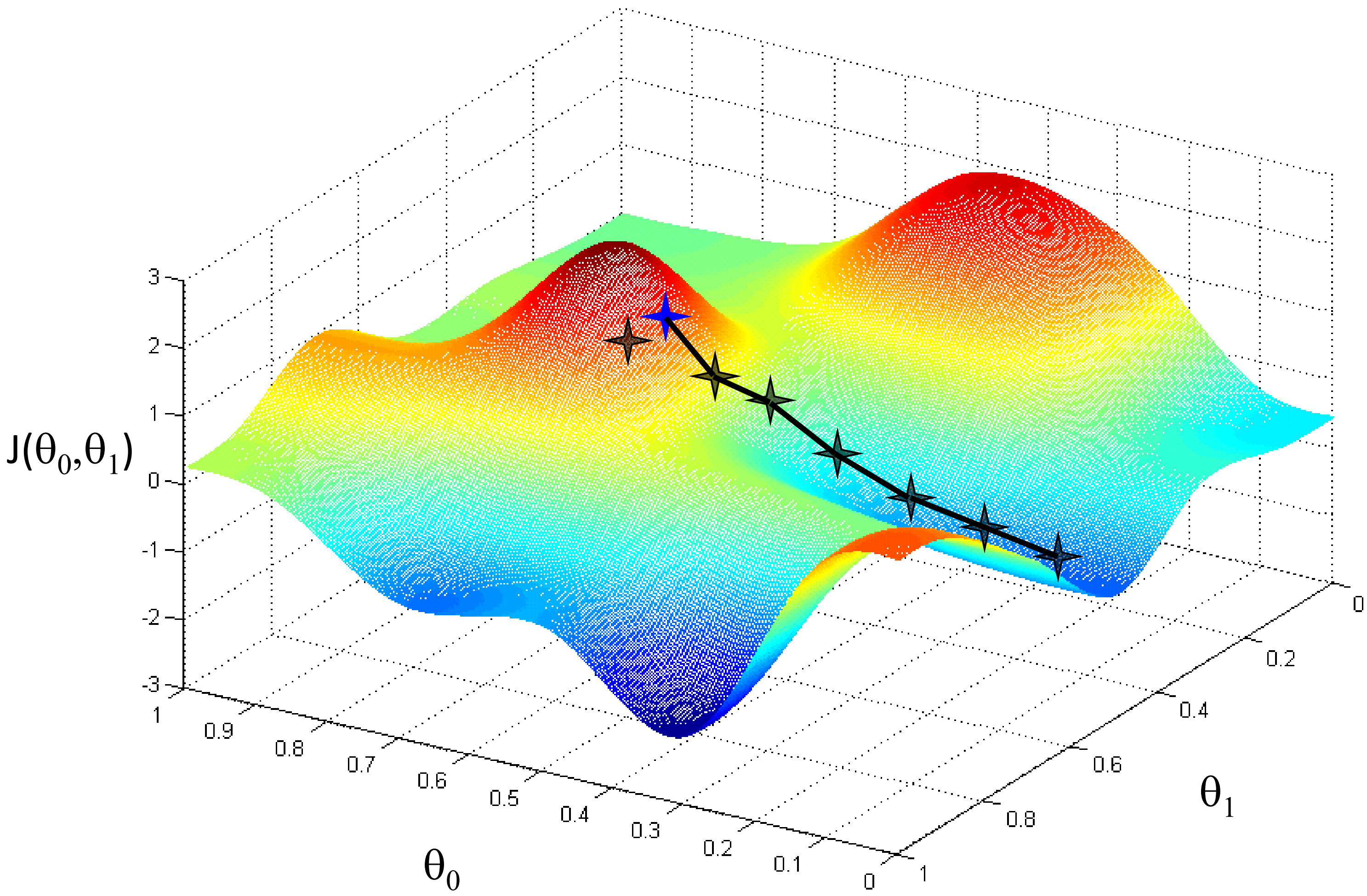
Gradient Descent

- Very well-known algorithm for finding (local) minima and maxima.
- Used for many optimization problems – not just regression.
- There are other and way more sophisticated methods

Iterative algorithm for finding min/max of function

1. Start with some θ_0, θ_1
2. Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$
   until you reach a minimum (hopefully)


J(θ_0,θ_1)
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1



$J(\theta_0,\theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1

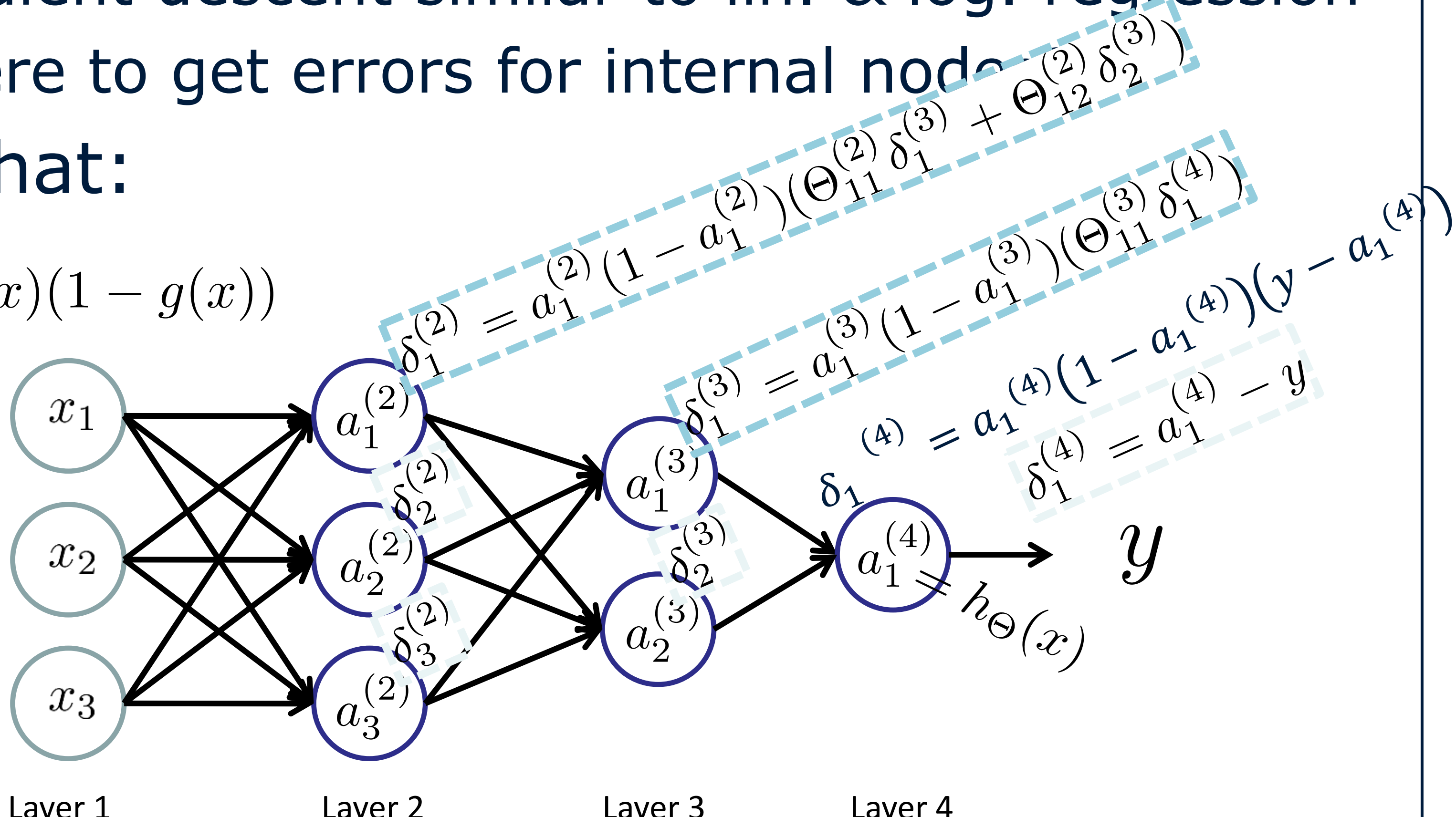
Learning the weights

Backpropagation

- gradient descent similar to lin. & log. regression
- where to get errors for internal nodes

Given that:

$$\frac{d}{dx}g(x) = g(x)(1 - g(x))$$



Get some deep understanding

- What do you think will happen if you run PSO on the benchmark functions?
- Which algorithm will have better performance (PSO or gradient descent)? – why?
- Under which circumstances does is make sense to combine PSO and gradient descent? How would it be best to combine these?

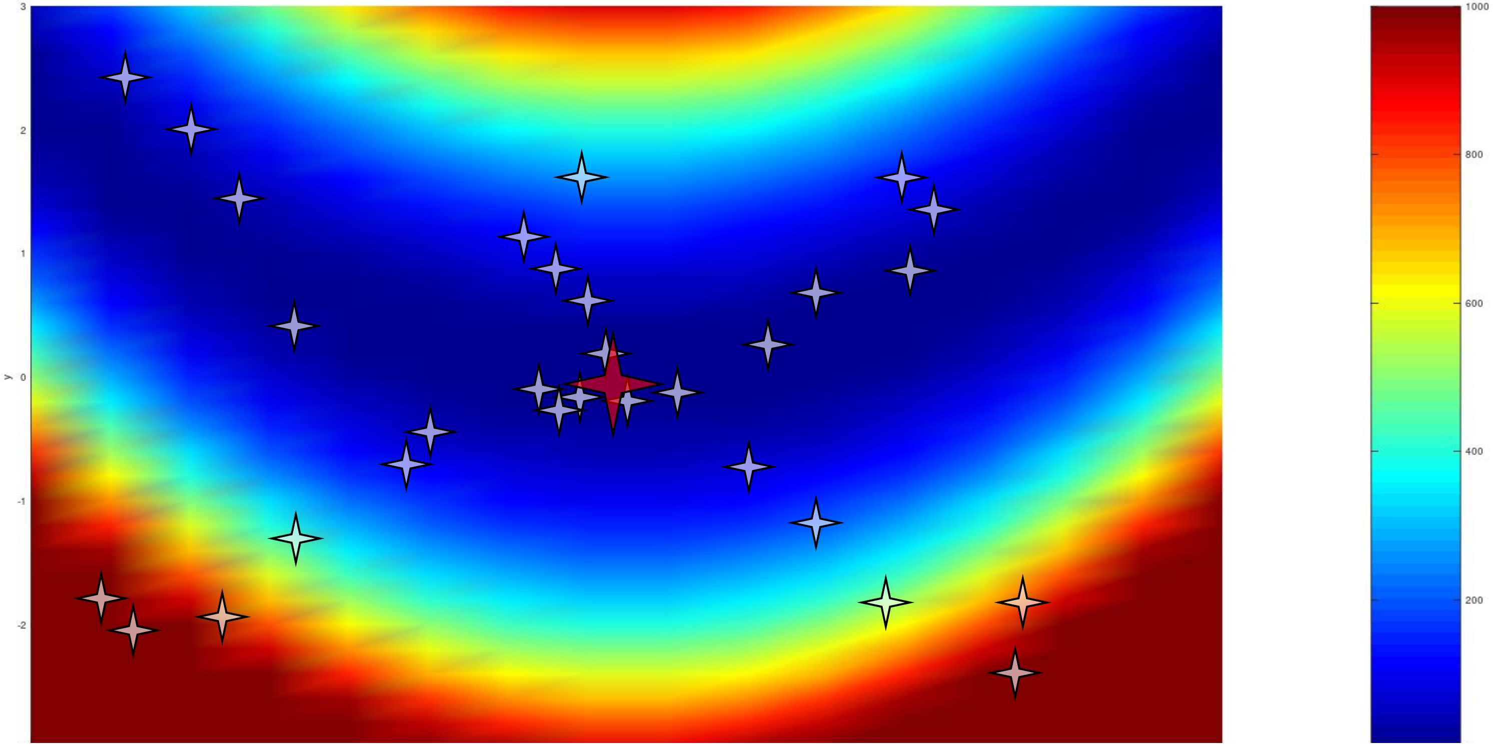
Summary: Your task

- Implement Rosenbrock and Rastrigin functions with two parameters
- Test PSO on both
- Reason about how Gradient Descent would perform on these functions (Gradient Descent does not have to be implemented – but you can if you want)
- Upload documented code
- Demonstrate simulations **(video demonstration)**
  - Show how particles move on benchmark function
  - Show plot of benchmark function at the same time
  - **Provide real simulation recordings (as in video game), not just pictures, no slides!**

Hand in

- Documented code (Python, C++, C, Java, Matlab)
  - Make sure that each group member codes something, add names to code (who did what?)
  - **Upload zip archive** containing all code (no other types of archives)
- Recommendation: use language for simple prototyping and plotting (e.g. Python-Anaconda, Matlab/Octave)
- Video demonstration: (mp4 only!, 5-6 minutes max, 150MB max)
  - Explain with your own voice - all team members must present something
  - Perform meaningful experiments, document experimental setup (parameters), explain results

Expected demo (but for actual PSO and both benchmark function)



Show how particles move together at every iteration

Plagiarism

- Do not copy and hand in code from colleagues
- You can talk to colleagues and help each other, but you cannot exchange code between groups.
- Write your own software

- **You must implement PSO yourself (no use of libraries here!)**

Homework until Tuesday

- Hand in **before** lecture start

Find favourite colleagues

- Group assignments (2-3 people per group)
  - Add your names here, so we can form groups on CANVAS
  - You can change your group again for coming assignment

- Always make sure that you understand all aspects of an assignment
- Several exam questions will be based on assignments!

https://docs.google.com/spreadsheets/d/1Zwho72baI0YQf8oAYn5Hs-PRx6U53OQz57NhqbSrLcc/edit?usp=sharing