Computer simulations in Physics

Optimization

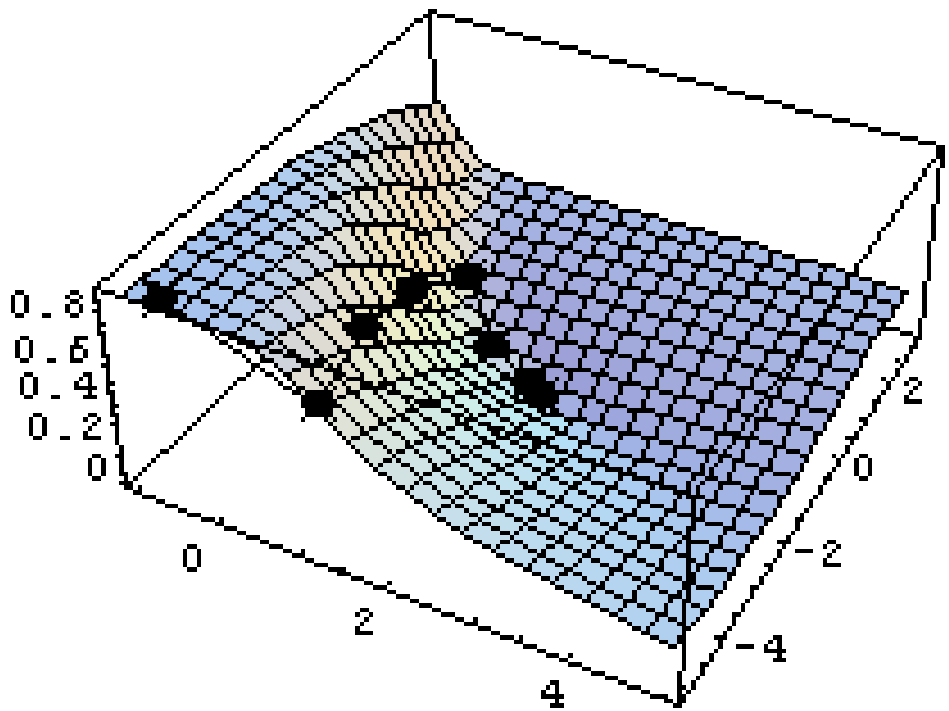
Janos Török

Department of Theoretical Physics

April 6, 2021

Optimization

- General problem of finding the ground state
- Phase-space:
- Arbitrary number of dimensions
- Methods:
  - Gradient methods
  - Simulated Annealing
  - Genetic algorithm



Optimization

- General problem of finding the ground state
- Phase-space:
- Arbitrary number of dimensions
- Methods:
  - Gradient methods
- Implementation
  - C: GSL
  - python: scipy.optimize
  - Both are very flexible and can be used with numerical or analytical derivatives

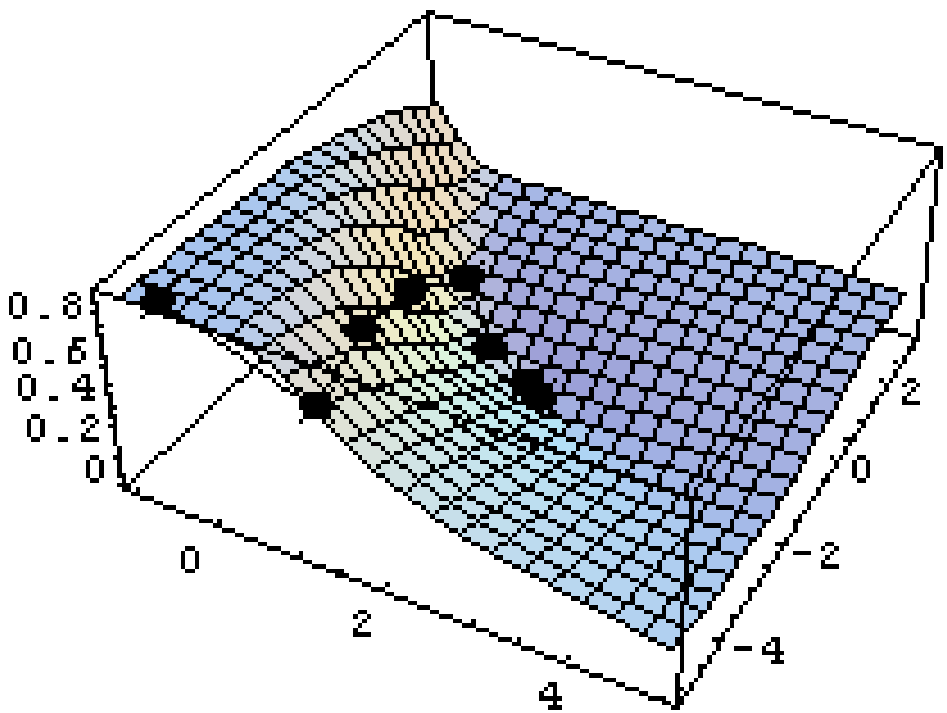
Gradient based optimization

- Given $f(\mathrm{x})$, with $\mathrm{x} = \{x_1, x_2, \ldots x_n\}$
- Gradient $\nabla f(\mathrm{x}) \equiv \mathrm{g}(\mathrm{x}) = \{\partial_1 f, \partial_2 f, \ldots \partial_n f\}$
- Second order partial derivatives: square symmetric matrix called the *Hessian matrix*:

$$\nabla^2 f(\mathrm{x}) \equiv H(\mathrm{x}) \equiv \begin{pmatrix} \partial_1\partial_1 f & \ldots & \partial_1\partial_n f \\ \vdots & \ddots & \vdots \\ \partial_1\partial_n f & \ldots & \partial_n\partial_n f \end{pmatrix}$$

General Gradient Algorithm

1. Test for convergence
2. Compute a search direction
3. Compute a step length
4. Update x



Steepest descent algorithm

1. Start from x_0
2. Compute $g(x_k) \equiv \nabla f(x_k)$. If $||g(x_k)|| \leq \varepsilon_g$ then stop, otherwise, compute normalized search direction $p_k = -g(x_k)/||g(x_k)||$
3. Compute α_k such that $f(x_k + \alpha p_k)$ is minimized
4. New point: $x_{k+1} = x_k + \alpha p_k$
5. Test for $|f(x_{k+1} - f(x_k))| \leq \varepsilon_a + \varepsilon_r |f(x_k)|$ and stop if fulfilled in two successive iterations, otherwise go to 2.

Conjugate Gradient Method

- The iteration
$$x_{n+1} = x_k - \gamma_n \nabla f(x_k),$$
- We can select γ such that if the function is quadratic in all directions it goes immediately into the minimum
- Idea: almost all minima are quadratic close to the minimum

$$\gamma_n = \frac{|(x_n - x_{n-1})^T [\nabla f(x_n) - \nabla f(x_{n-1})]|}{||\nabla f(x_n) - \nabla f(x_{n-1})||^2}$$

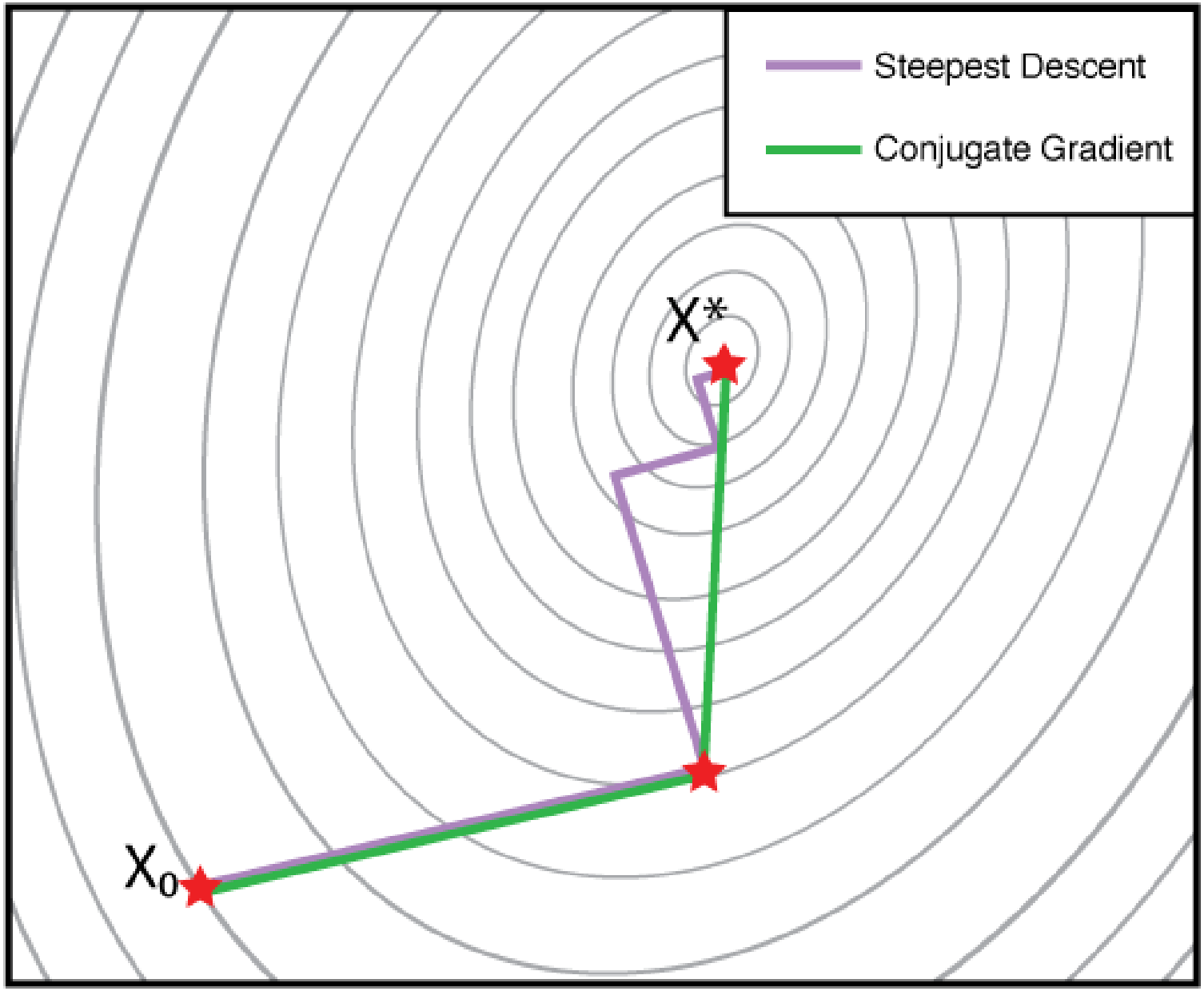
Conjugate Gradient Method

1. Start from x_0
2. Compute $g(x_k) \equiv \nabla f(x_k)$. If $||g(x_k)|| \leq \varepsilon_g$ then stop, otherwise Go to 6
3. $p_0 = -g_0$
4. Compute $g(x_k) \equiv \nabla f(x_k)$. If $||g(x_k)|| \leq \varepsilon_g$ then stop, otherwise continue
5. Compute the new conjugate gradient direction $p_k = -g_k + \beta_k p_{k-1}$, where

$$\beta = \left(\frac{||g_k||}{||g_{k-1}||} \right)^2$$

6. Compute α_k such that $f(x_k + \alpha p_k)$ is minimized
7. New point: $x_{k+1} = x_k + \alpha p_k$
8. Test for $|f(x_{k+1} - f(x_k))| \leq \varepsilon_a + \varepsilon_r |f(x_k)|$ and stop if fulfilled in two successive iterations, otherwise go to 4.

Conjugate Gradient Algorithm



Modified Newton's method

Second order method

1. Start from x_0
2. Compute $g(x_k) \equiv \nabla f(x_k)$. If $||g(x_k)|| \leq \varepsilon_g$ then stop, otherwise, continue
3. Compute $H(x_k) \equiv \nabla^2 f(x_k)$ and the search direction $p_k = -H^{-1} g_k$
4. Compute α_k such that $f(x_k + \alpha p_k)$ is minimized
5. New point: $x_{k+1} = x_k + \alpha p_k$
6. Go to 2.

Optimization

- General optimization
- Parameters of the system x (input)
  - for networks: adjacency matrix, degree distribution
  - for pattern recognition: data, or processed data (e.g Fourier spectrum, etc.)
- Optimized property: $y = f(\mathrm{x})$, we search for $f(.)$ which gives the desired y
  - any measurable quantity
  - classification of data (e.g. $y = 1$ for cat, $y = 2$ for dog, etc.)
- Loss function, $L(f)$, the quantity to be minimized (Energy/Hamiltonian)
  - Least square: $L(f) = [y - f(\mathrm{x})]^2$
  - Hamming distance: $L(f) = \begin{cases} 1 & \text{if } f(\mathrm{x}) = y \\ 0 & \text{otherwise} \end{cases}$

Linear regression

- Assume linear form for the loss function:

$$f_{\mathrm{w}}(\mathrm{x}) = w^T \mathrm{x}$$

- n data points
- m variables
- The problem must not be linear, e.g. polynomial fit w contains the coefficients of the polynomial (i indexes data points):

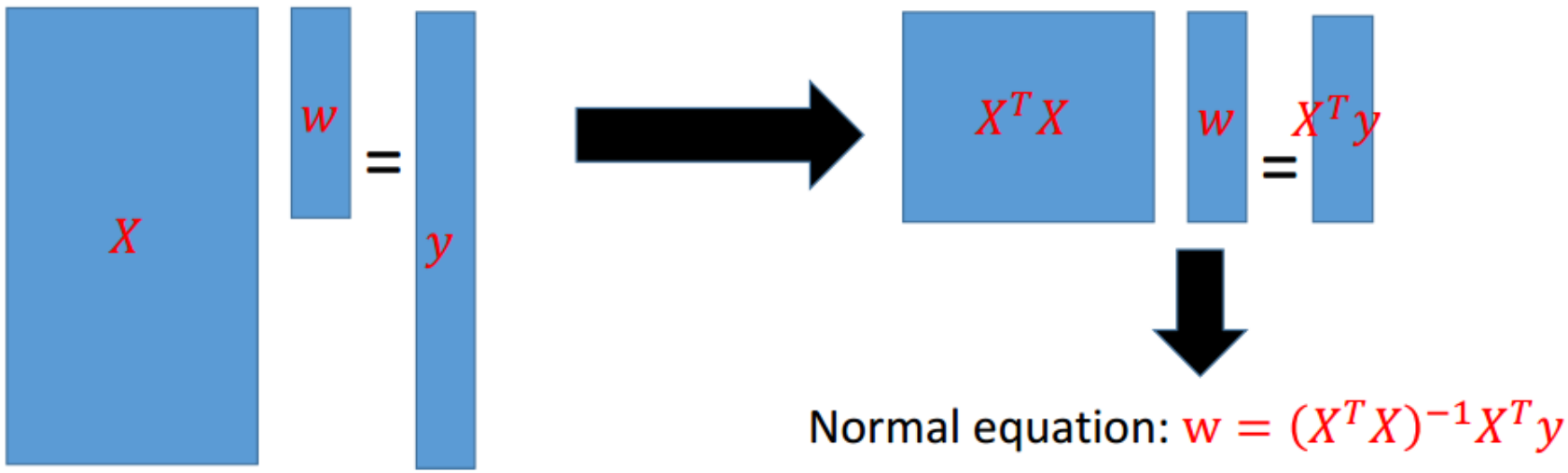
$$y_i = w_0 + w_1 x_i^{(1)} + w_2 x_i^{(2)} + \cdots + w_m x_i^{(m)} + \varepsilon_i$$

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(m)} \\ 1 & x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(m)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n^{(1)} & x_n^{(2)} & \dots & x_n^{(m)} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

- If X is a square matrix than $\mathrm{w} = X^{-1}\mathrm{y}$
- Otherwise $\mathrm{w} = (X^T X)^{-1} X^T \mathrm{y}$

Linear regression

- If X is a square matrix than $\mathrm{w} = X^{-1}\mathrm{y}$
- Otherwise $\mathrm{w} = (X^T X)^{-1} X^T \mathrm{y}$



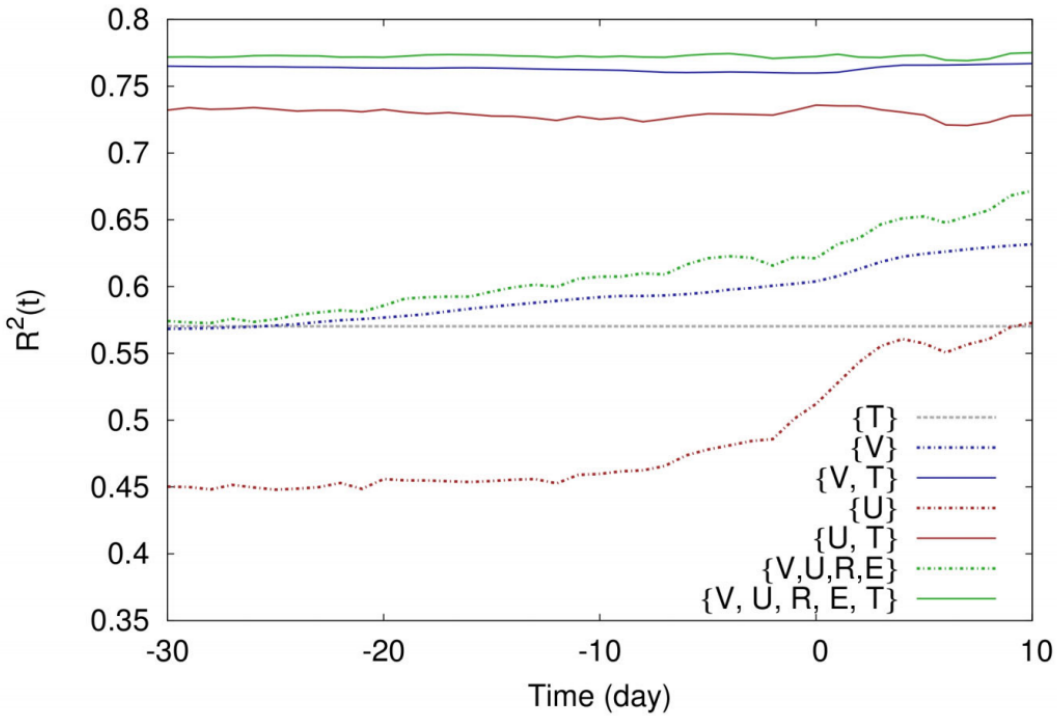
Linear regression

Solution: $\mathrm{W} = (X^T X)^{-1} X^T \mathrm{y}$

- Minimize $\varepsilon^2 = (\mathrm{Y} - \mathrm{XW})^T (\mathrm{Y} - \mathrm{XW})$ with respect to W
- Derivate it with respect to W:
  $\frac{d}{d\mathrm{W}} (\mathrm{Y} - \mathrm{XW})^T (\mathrm{Y} - \mathrm{XW}) = -2\mathrm{X}^t (\mathrm{Y} - \mathrm{WW}) = 0$
- Solve for W: $\mathrm{X}^T \mathrm{Y} = (\mathrm{X}^T \mathrm{X})\mathrm{W}$

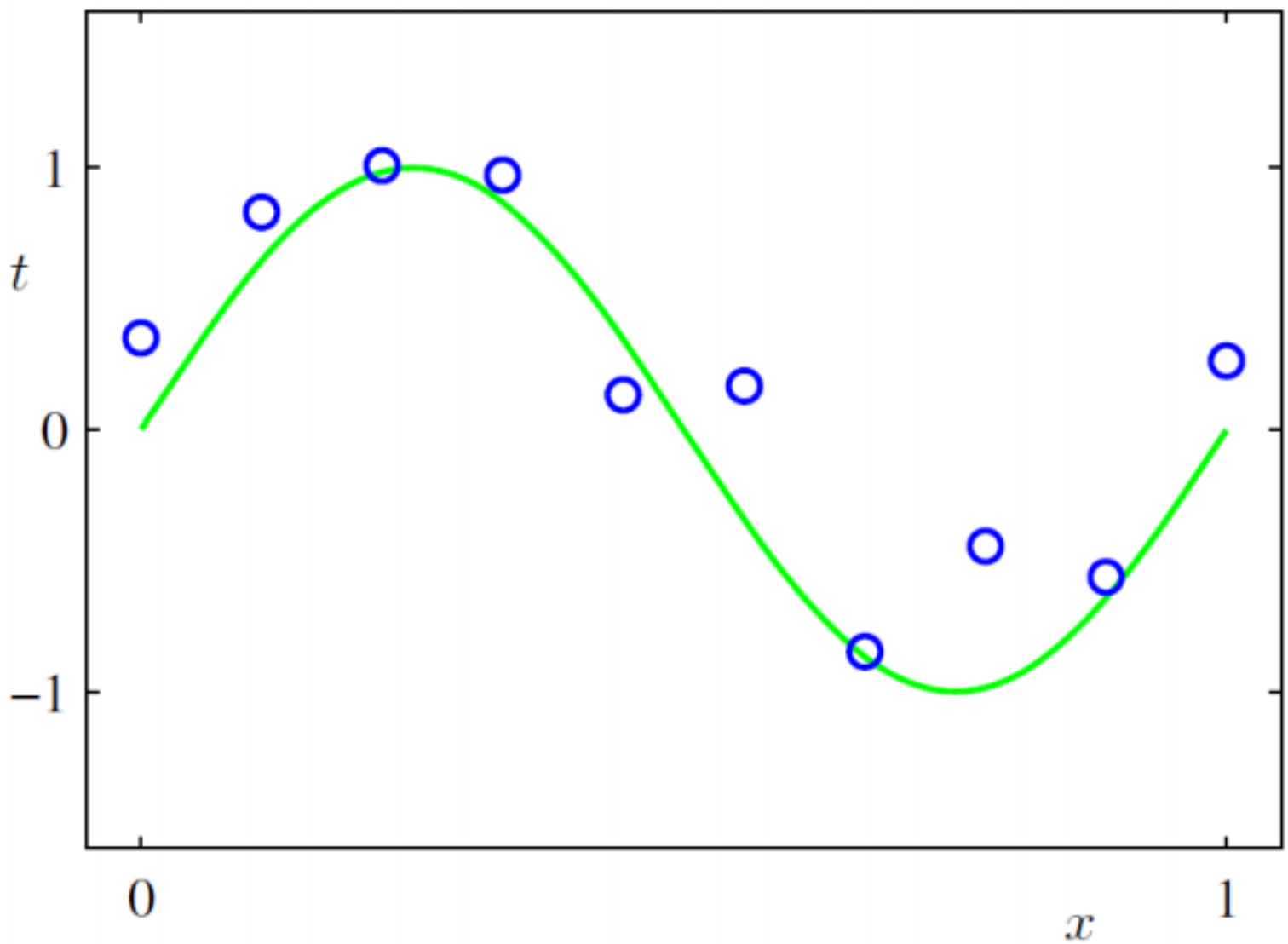
Linear regression: Example

- Movie success prediction
- Variables:
  - V Number of views of the Wikipedia page
  - U Number of editors of the Wikipedia page
  - E Number of edits made on the Wikipedia page
  - R Collaborative rigor of Wikipedia editing
  - T Number of theaters that screen the movie
- Time=0 day of release

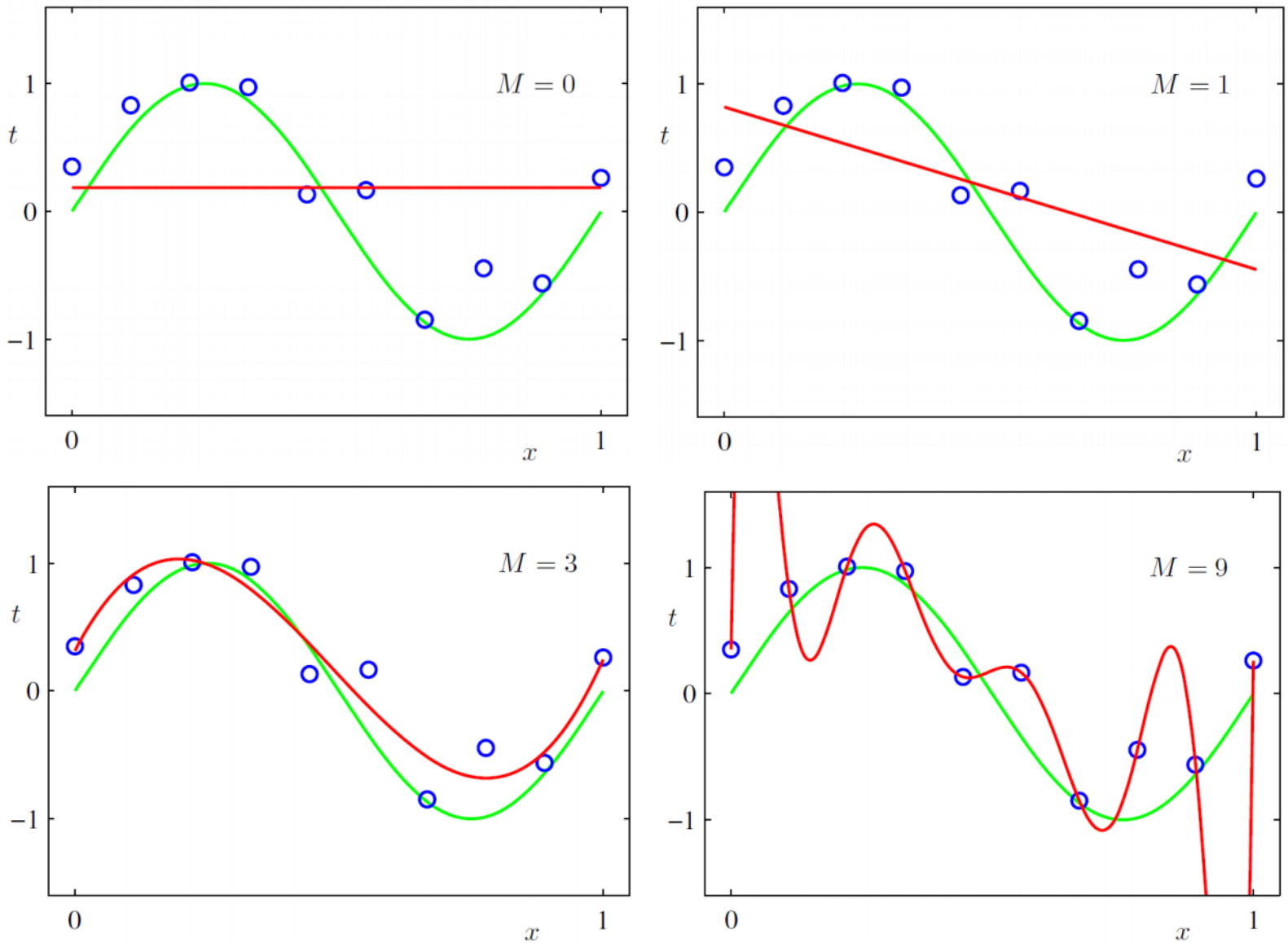


Linear regression: Over fitting

$$t = \sin(2\pi x) + \xi$$



Linear regression: Over fitting



Maximum likelihood

- Unknown parameter θ which maximizes the likelihood to obtain the given data
- We know the probability distribution $f(x_i, \theta)$ of random variables x_i
- Likelihood:

$$L(\theta) = f(x_1, \theta) f(x_2, \theta) \cdots f(x_n, \theta)$$

- θ parameter vector for the probability distribution
- Generally maximize $\log L(\theta)$
- Can be solved in many cases
- Probability distributions must be known in advance
- Parameters obtained through equations.
- Main usage: model parameter estimation

Maximum likelihood, example

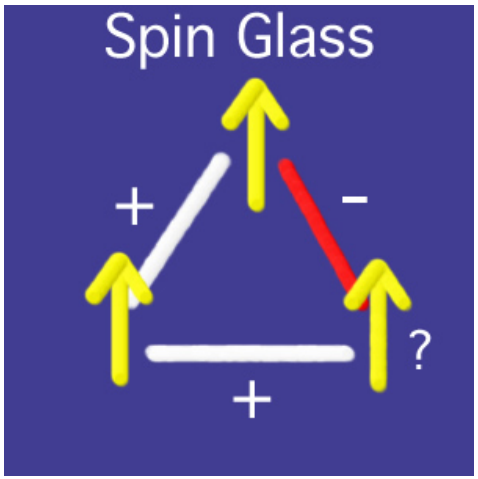
- Two dice, one normal 6 side, the other has probabilities of $p(1,2,3,4,5,6) = (1/12, 1/6, 1/6, 1/6, 1/6, 1/4)$
- We have rolled 10 times and the result is $1,2,6,5,6,5,3,6,4,2$.
- What is the likelihood that we chose the fake dice?
- $p_{\text{norm}} = \frac{1}{6^{10}} = 1.65 \cdot 10^{-8}$
- $p_{\text{fake}} = \frac{1}{6^6 \cdot 4^3 \cdot 12} = 2.79 \cdot 10^{-8}$
- What is the dice which gave this with the highest probability $(1/10, 1/5, 1/10, 1/10, 1/5, 3/10)$

Glassy behavior, frustration

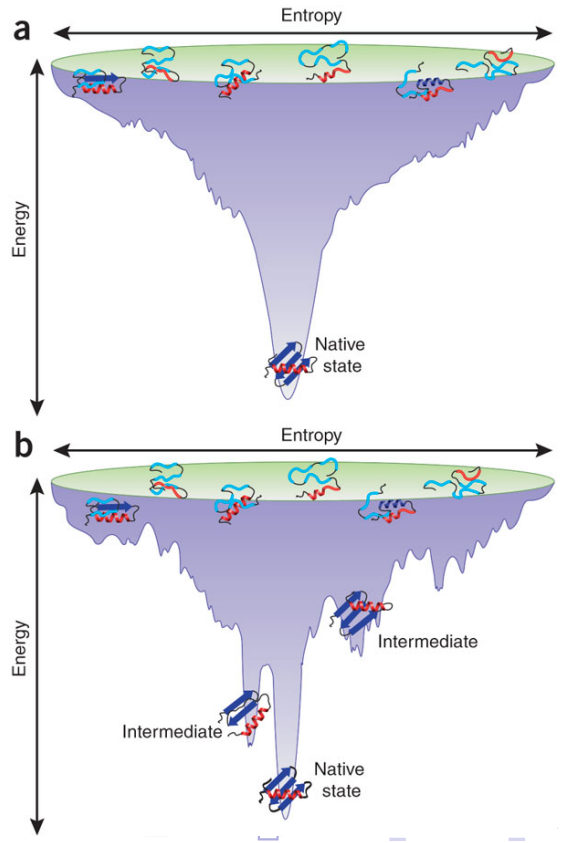
- Model glass: spin glass:

$$H = -\frac{1}{2}\sum_{\langle i,j\rangle} J_{ij} S_i S_j$$

- where J_{ij} are random quenched variables with 0 mean (e.g. $\pm J$ with probability half)

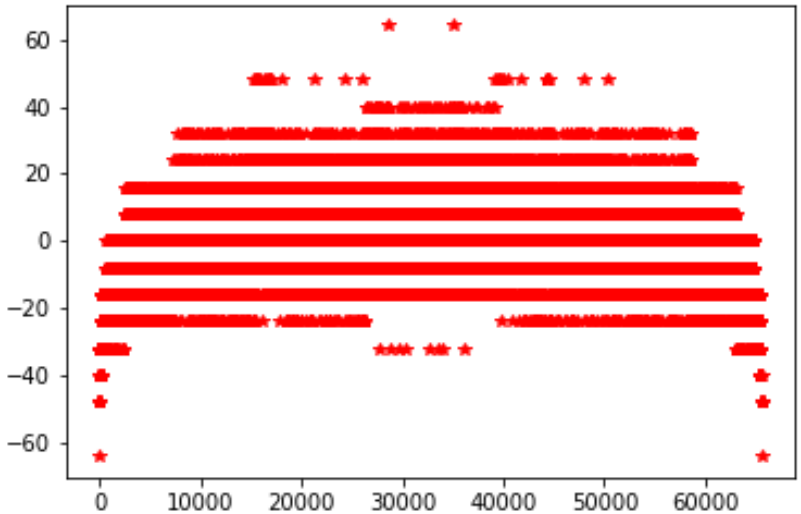


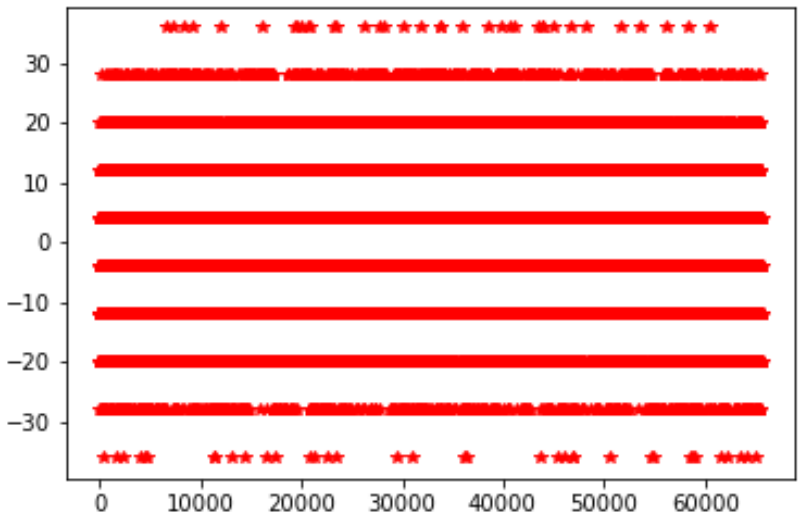
Rugged energy landscape.



Energy landscape

Ising vs. spin glass (X Axis: binary representation of number)

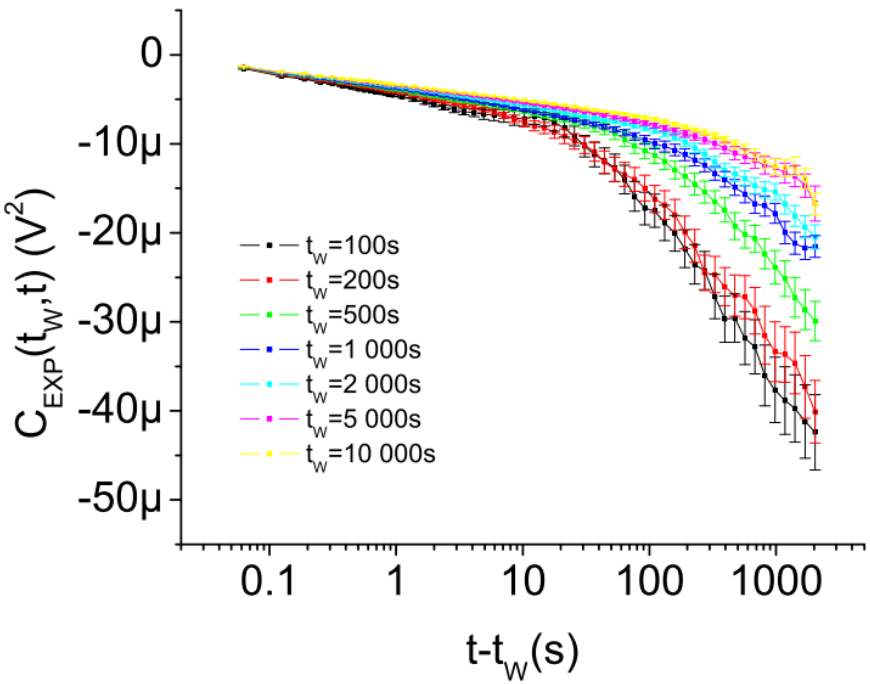




Spin glass: Aging

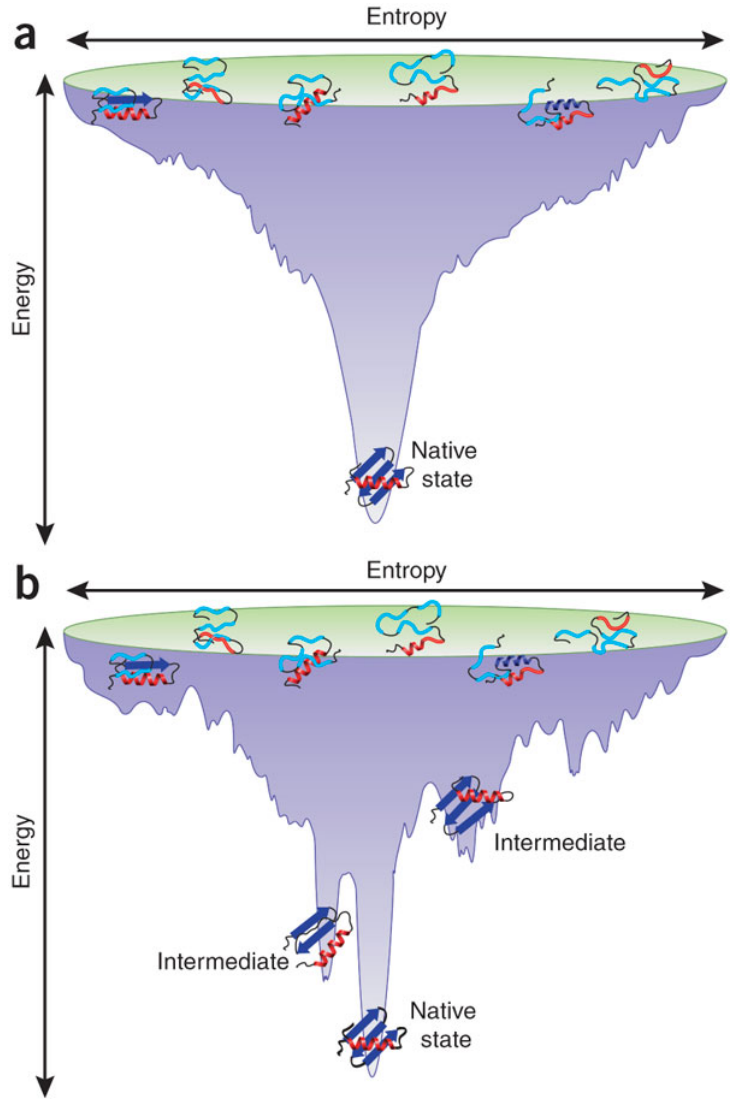
- Heat up the sample where it equilibrates fast
- Quench it below T_c
- Wait t_w
- Switch off the magnetic field
- Measure a parameter $q(t_w, t_w + t)$
- Often q is a covariance (X observable):

$$q(s,t) = E(X_t X_s) - E(X_t)E(X_s)$$



Spin glass: Trap model (Bouchaud)

- The evolution of the particle system is represented by a Markov process in a random energy landscape
- The process will spend most time into deep valleys of lowest energy where it will be trapped
- The time spent in these valleys is random and aging will appear when the mean time spent in these valleys diverges
- Order parameter: the magnetization and the two point spin correlation between spins at the same site in two different replicas



Rugged energy landscape

- Typical example NP-complete problems:
  - Traveling salesman
  - Graph partitioning
  - Spin glass
- No full optimization is possible (do we need it?)
- Very good minima can be obtained by stochastic optimization
  - Simulated annealing
  - Genetic algorithm

Travelling salesman

- N cities on the 2d space
- Distance between the cities is the Euclidean distance (birds flight)
- The traveller must visit all cities once
- The trajectory is circular so the traveller must return to the starting city
- The optimized quantity is the travelled distance

Simulated annealing

- Cool down the system slowly
- Speed is crucial and many experiments are needed
- No guarantee that we find something meaningful
- Warm up and down if needed, if the system quenched into a local minimum
- One needs a Hamiltonian (or a fitness function) and an elementary move
  - Spin glass: Metropolis
- Traveling salesman
  - Minimal traveling path for visiting a number of cities
  - Elementary move: swap two cities ($T \sim$ alcohol)

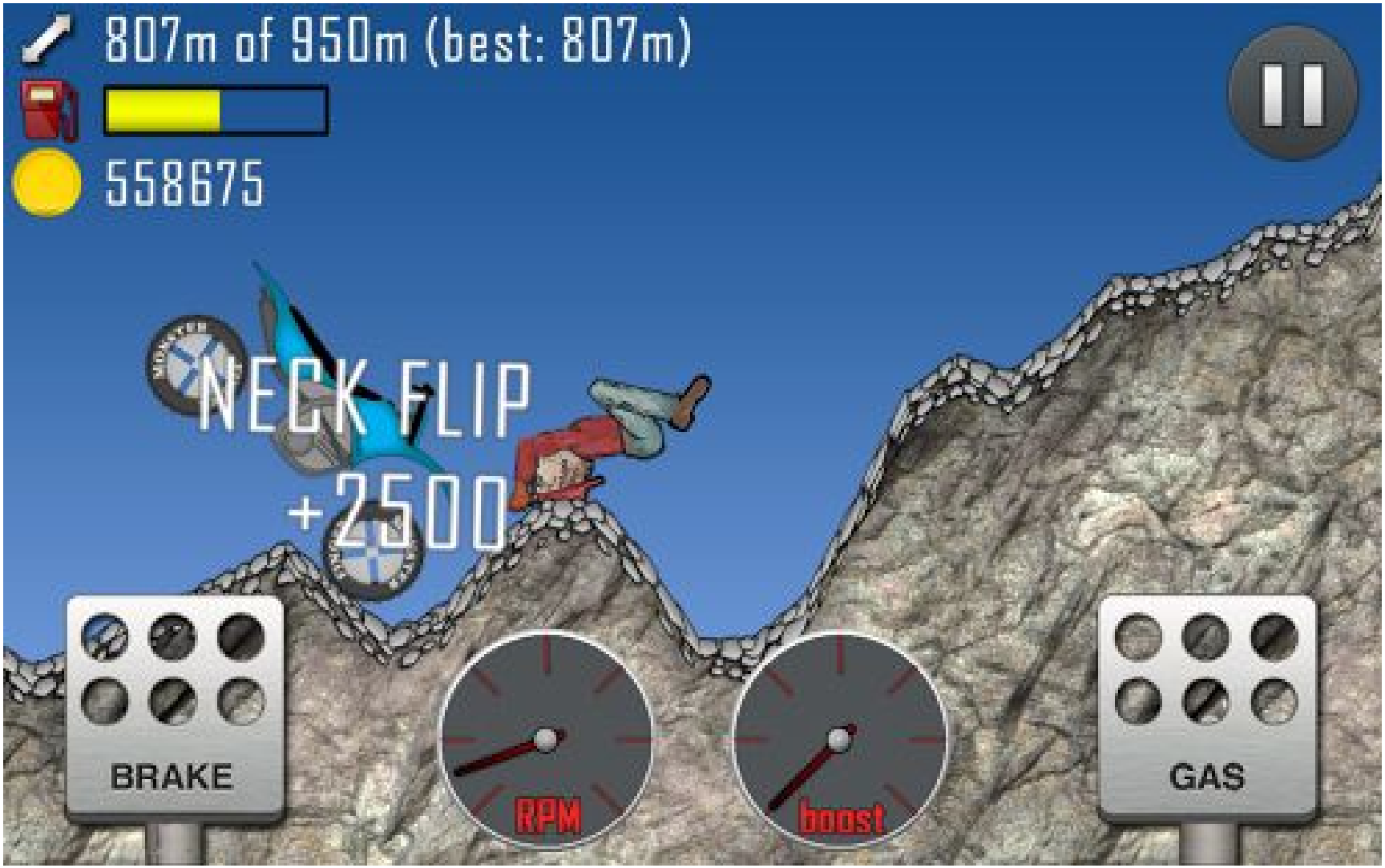
Simulated annealing

- Loss function: e.g. energy E
- Minimize energy like in a physical system
- Vary parameter set w in an ergodic way (all possible values must be reachable)
- Observe detailed balance:

$$p(i \to j) = \begin{cases} 1 & \text{if } E_j < E_i \\ \exp[\beta(E_i - E_j)] & \text{otherwise} \end{cases}$$

- where $\beta \simeq 1/T$
- Slowly decrease T

Hill climb


807m of 950m (best: 807m)
558675
NECK FLIP
+2500
BRAKE
RPM
boost
GAS

Travelling salesman

Genetic algorithm

- Learn from nature
- Let the fittest to survive

Genetic algorithm

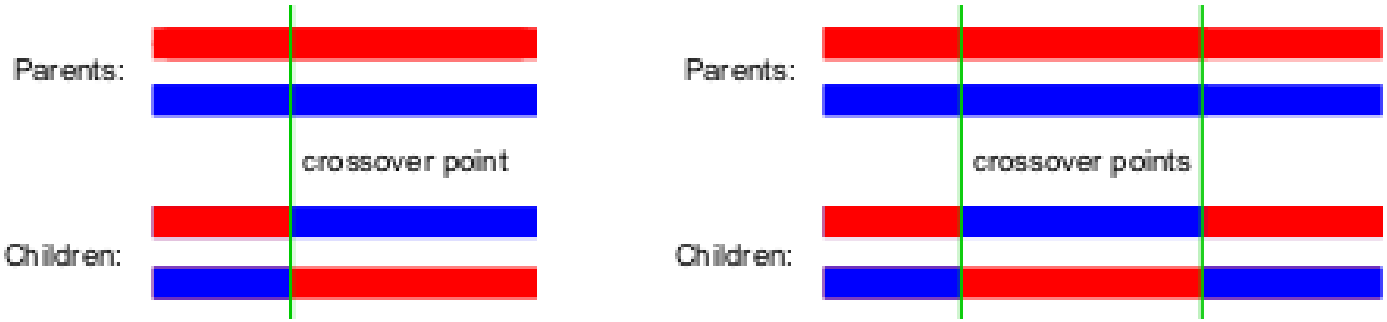
- Learn from nature
- Let the fittest to survive
  - Fitness function, e.g. energy, length, etc.
- Combine different strategies
- State is represented by a vector (genetic code or genotype)
  - Phasespace, city order, neural network parameters, etc.
- Offsprings have two parents with shared genetic code
- Mutations
- Those who are not fit enough die out
  - Keep the number of agents fixed

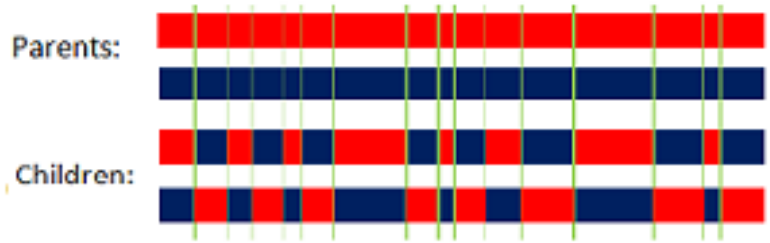
Genetic algorithm terminology

- Chromosome: Carrier of the genetic representation
- Gene: Smallest units in the chromosome with individual meaning
- Parents: Pair of chromosomes, which produce offsprings
- Population: Set of chromosomes from which the parents are selected. Its size should be larger than the length of the chromosome
- Selection principle: The way parents are selected (random, elitistic)
- Crossover: Recombination of the genes of the parents by mixing
- Crossover rate: The rate by which crossover takes place ($\sim$90%)
- Mutation: Random change of genes
- Mutation rate: The rate by which mutation takes place ($\sim$1%)
- Generation: The pool after one sweep.

Genetic algorithm: Reproduction

- Two parents and two children

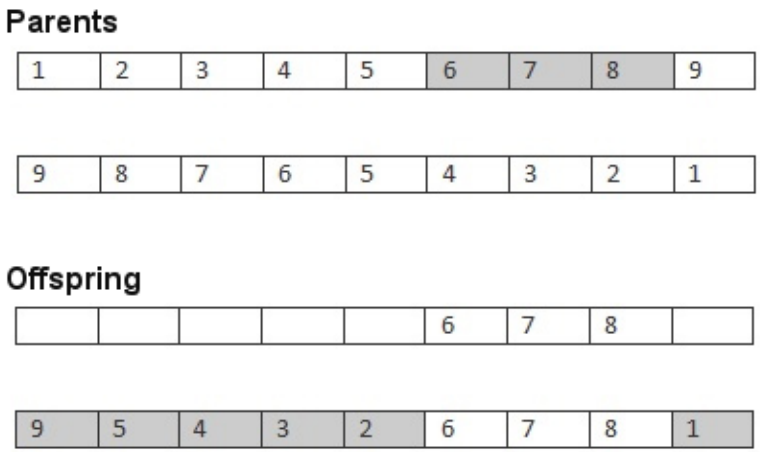




With a probability of 0.5, children have 50% genes from first parent and 50% of genes from second parent even with randomly chosen crossover points.

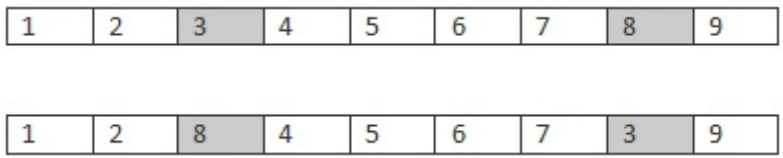
Genetic algorithm schema

1. Start with a randomly generated population
2. Calculate the fitnesses
3. Selection
   - Random
   - Best fitness (keep top 50% and generate new 50%)
   - Roulette (Monte-Carlo) selection
4. Crossover: offsprings must be viable (Sometimes difficult)

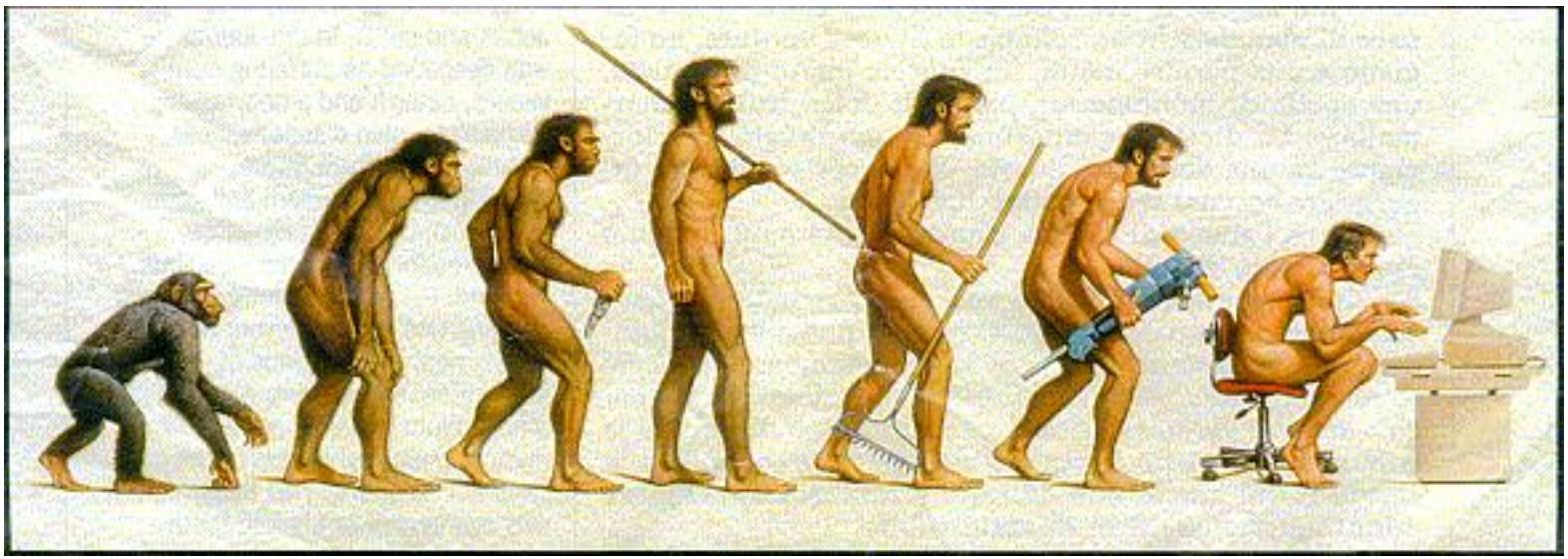


Genetic algorithm schema

1. Start with a randomly generated population
2. Calculate the fitnesses
3. Selection
   - Random
   - Best fitness (keep top 50% and generate new 50%)
   - Roulette (Monte-Carlo) selection
4. Crossover: offsprings must be viable (Sometimes difficult)
   - One-point
   - Two-point
   - Uniform
   - Mutation: small rate



Genetic algorithm example



Practice

Please, solve one of the following exercises to earn the 20 points for practice. You can work in pairs as usual. If you wish, you can solve both problems for a total of 40 points.

- Simulated annealing of the travelling salesman
- Linear regression of the Titanic survivals

Simulated annealing of the travelling salesman

- Place N cities randomly on the unit square. Use $N = 50, 100$ cities for the test.
- Distance between the cities is the Euclidean distance
- The algorithm must propose an order to visit all cities
- The trajectory is circular so the traveller must return to the starting city
- The optimized quantity is thus the sum of the city distances in the proposed order of visit.
- Implement both type of elementary steps:
  - Exchange two cities in the order of visit: e.g. ABCDEFA $\rightarrow$ ABEDCFA
  - Reverse the visit direction of a part of the trajectory: e.g. ABCDEFA $\rightarrow$ AEDCBFA
- Measure the efficiency of the two elementary steps

Linear regression of the Titanic survivals

- Download the Titanic datafile from `https://gist.github.com/michhar/2dfd2de0d4f8727f873422c5d959fff5`
- Keep the following columns: Pclass, Sex, Age, Parch, Fare (Parch: Number of Parents/Children Aboard) for input and the column Survived for output
- Replace text data with different numbers for different strings
- Rescale the numbers in each column to have zero mean and 1 variance
- Do the linear regression. You can use builtin matrix multiplication algorithm, but no builtin regression algorithms
- Discuss the importance of the different parameters

Homework

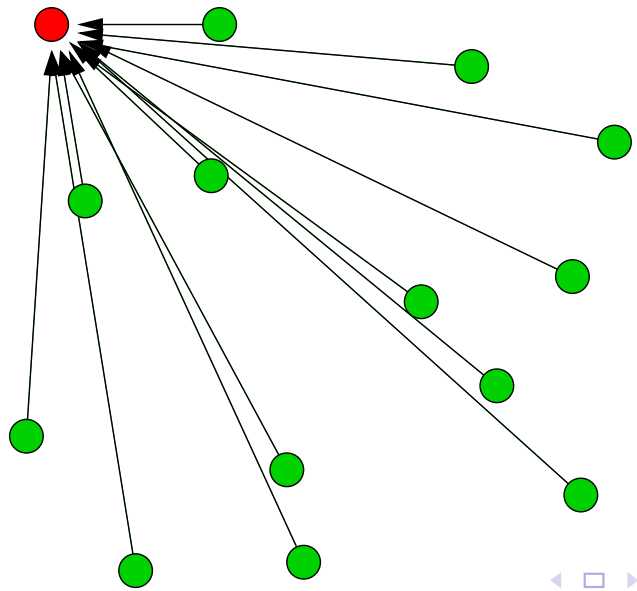
Genetic algorithm

Network optimization example

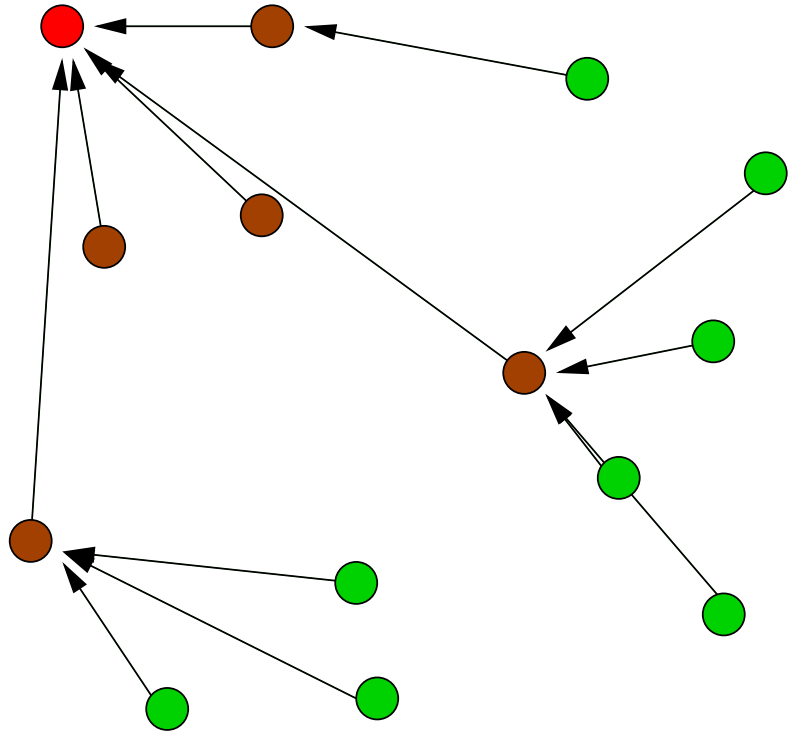
- Optimize transmission energy consumption (Jin et al. 2003)
- Sensors scattered in space
- One data collector
- Sensors can be turned into intermediate collectors (hubs)
- Energy consumption

$$E(k, d) = E_{elec} + d^2 E_{amp},$$

where E_{elec} is the base electric need of a radio station, E_{amp} is the energy need of an amplifier and d is the distance to transmit to.

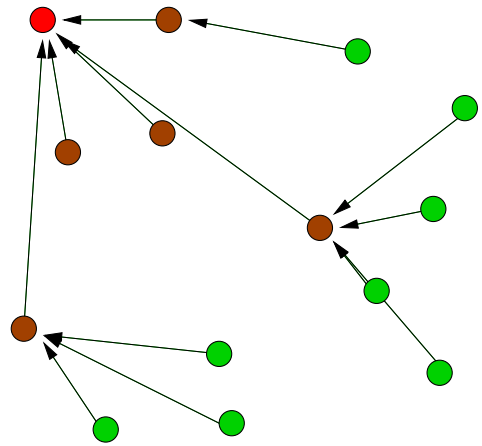


Network optimization example 2.



- Coding: bit
  - 1: direct connection to data collector (it is a hub)
  - 0: connection to nearest hub

Network optimization example 3.



- Coding: bit
  - 1: direct connection to data collector (it is a cluster head)
  - 0: connection to nearest cluster head
- Crossover:

Indv1: 1 1 1 0 0 1 0 1

Indv2: 1 0 1 1 1 1 1 0

Crossover point

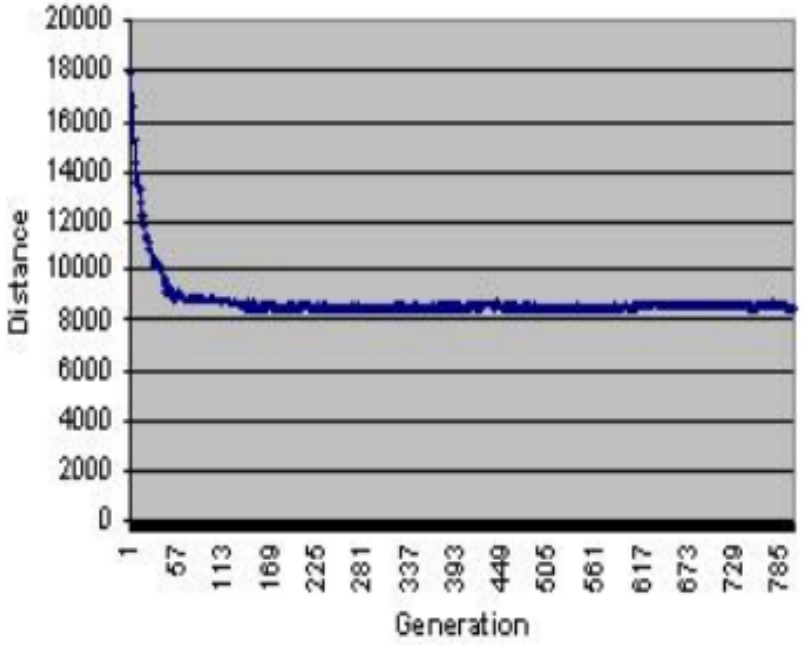
After crossover, two offspring are created as below:

Child1: 1 1 1 0 1 1 1 0

Child2: 1 0 1 1 0 1 0 1

- Mutation: change a bit

Network optimization example results



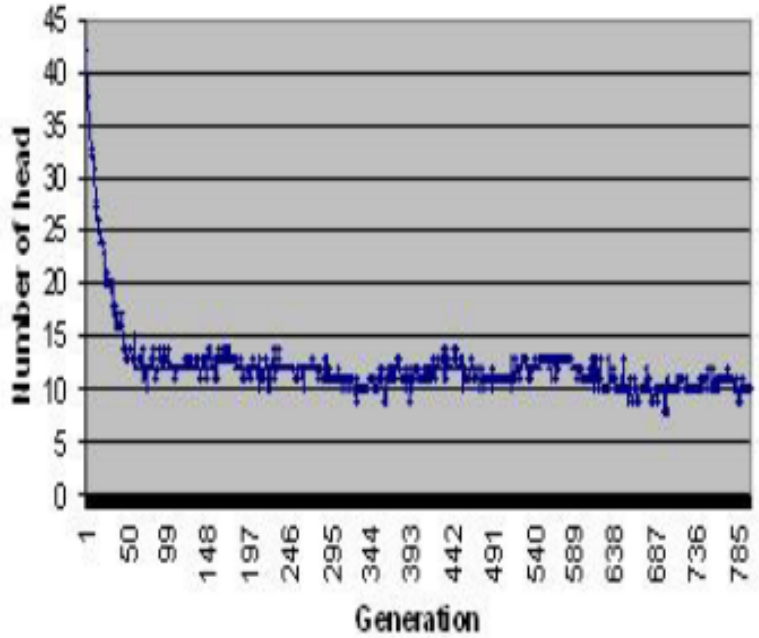


Table 1: Test Results of Three Problem Size

Node Size	Population	Converge after generation	Head (%)	Distance decreased
100	80	105	10.0%	76.85%
200	160	120	10.0%	81.20%
400	300	145	11.2%	82.20%