



Optimisation algorithms in Statistics II, lecture 3

Frank Miller, Department of Statistics; Stockholm University

April 27, 2021

Course schedule

- Topic 1: **Stochastic gradient descent and quasi-Newton algorithms**
Lectures: March 23; Time 10-12, 13-15 (online, Zoom)
- Topic 2: **Particle swarm optimisation and stochastic gradient descent with momentum**
Lectures: April 13; Time 10-12, 13-15
- Topic 3: **Simulated annealing and genetic algorithms**
Lectures: April 27; Time 10-12, 13-15

Course homepage:

<http://gauss.stat.su.se/phd/oasi/optimisation2.html>

Includes reading material, lecture notes, assignments

Today's schedule

- Some remarks connected to L1 and L2
- Simulated annealing
 - Why does simulated annealing work?
 - Generalisation of optimisation problem
 - Convergence result
- Genetic algorithms
 - Idea of algorithm
- Reflections

Some remarks connected to L1 and L2

Quasi-Newton method

- The BFGS (quasi-Newton) method has iteration

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - (\mathbf{M}^{(t)})^{-1} \mathbf{g}'(\mathbf{x}^{(t)})$$

and

$$\mathbf{M}^{(t+1)} = \mathbf{M}^{(t)} - \frac{\mathbf{M}^{(t)} \mathbf{z}^{(t)} (\mathbf{M}^{(t)} \mathbf{z}^{(t)})^T}{\mathbf{z}^{(t)T} \mathbf{M}^{(t)} \mathbf{z}^{(t)}} + \frac{\mathbf{y}^{(t)} \mathbf{y}^{(t)T}}{\mathbf{y}^{(t)T} \mathbf{z}^{(t)}}$$

- For higher dimensional problems, computation of inverse could be computationally intensive, but can be avoided by applying the Sherman-Morrison-Woodbury formula
- Defining $\mathbf{L}^{(t)} = (\mathbf{M}^{(t)})^{-1}$, the matrix update can be done by
$$\mathbf{L}^{(t+1)} = \left(\mathbf{I} - r_k \mathbf{z}^{(t)} (\mathbf{y}^{(t)})^T \right) \mathbf{L}^{(t)} \left(\mathbf{I} - r_k \mathbf{y}^{(t)} (\mathbf{z}^{(t)})^T \right) + r_k \mathbf{z}^{(t)} \mathbf{z}^{(t)T}$$
 with $r_k = 1/(\mathbf{y}^{(t)T} \mathbf{z}^{(t)})$
- Starting value could be $\mathbf{L}^{(0)} = \mathbf{M}^{(0)} = \mathbf{I}$

Sherman-Morrison-Woodbury formula

- Sherman-Morrison formula: Let $A \in \mathbb{R}^{n \times n}$ be an invertible matrix and $u, v \in \mathbb{R}^n$ vectors. Matrix $A + uv^T$ is invertible if and only if $v^T A^{-1} u \neq -1$ and then,

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1 + v^T A^{-1}u}.$$

Note: uv^T is an $\mathbb{R}^{n \times n}$ -matrix with rank 1

- Sherman-Morrison-Woodbury formula: Let $A \in \mathbb{R}^{n \times n}$ be an invertible matrix and $U, V \in \mathbb{R}^{n \times k}$ matrices. Matrix $A + UV^T$ is invertible if and only if $I - V^T A^{-1} U \in \mathbb{R}^{k \times k}$ invertible. Then,

$$(A + UV^T)^{-1} = A^{-1} - A^{-1}U(I - V^T A^{-1}U)^{-1}V^T A^{-1}.$$

Note: UV^T is an $\mathbb{R}^{n \times n}$ -matrix with rank k

- The formulae can speed up iterative programs if n large and k small, since inversion might be costly (example: BFGS)



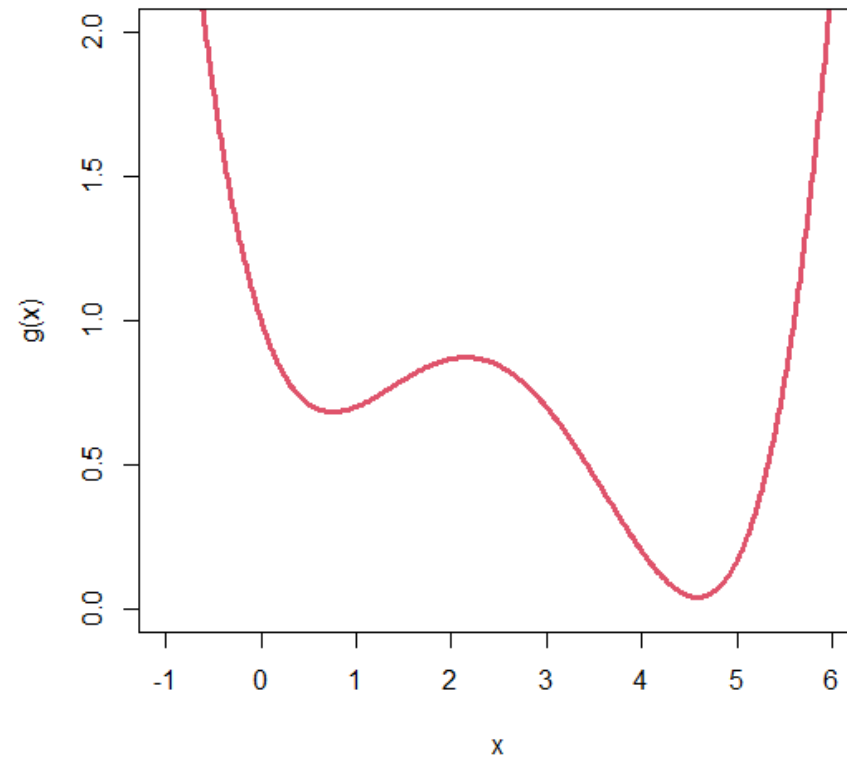
Particle swarm optimisation – choice of hyperparameters using empirical studies

- Example: Problem 3.2 from OASI
 - one global and three other local optima
 - use different swarm sizes (e.g. 10, 20, 50, 100) and different average percentage of informants (e.g. 0.1, 0.2, 0.5, 1), run 100 times each, and report percentage identifications of global maximum
 - here: fixed function given to be optimised
- In general, one might want to compare algorithms for a set of easy and difficult optimisation problems
- For comparability, often “standard optimisation problems” used; see e.g. [Liang et al. \(2013\)](#)
- Can be mathematical functions or statistical optimisation problems

Comparisons of algorithms or hyper-parameter choices using empirical studies

- After choosing some standard optimisation problems, one needs to define a success criterion (example in Clerk, 2016)
- Possibility: count runs of algorithm leading to a solution x_s with $g(x_s) < g(x^*) + \delta$; here x^* true position of global minimum, and δ small (ideally $\delta < g(x_L) - g(x^*)$ for any local minimum x_L)
- If the true success rate for an algorithm is p , we observe a $\text{Bin}(1, p)$ -random variable in each run
- Success rate has sd $\sqrt{\frac{p(1-p)}{n}}$ when doing n runs
- E.g. $p = 0.8, n = 100 \rightarrow \text{sd} = 0.04$.

Simulated annealing



[AlphaOpt \(2017\). Introduction To Optimization: Gradient Free Algorithms \(2/2\)](#)

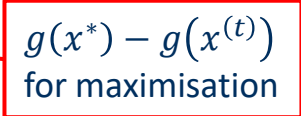
[Simulated Annealing, Nelder-Mead \(0:15-1:35\)](#)

2021-04-27 Optimisation algorithms in Statistics II L3



Stockholm
University

Simulated annealing

- Start value $x^{(0)}$; Stage $j=0,1,2,\dots$ has m_j iterations; set $j=0$
- Given iteration $x^{(t)}$, generate $x^{(t+1)}$ as follows:
 1. Sample a candidate x^* from a proposal distribution $p(\cdot|x^{(t)})$
 2. Compute $h(x^{(t)}, x^*) = \exp\left(\frac{g(x^{(t)}) - g(x^*)}{\tau_j}\right)$ 
 3. Define next iteration $x^{(t+1)}$ according to
$$x^{(t+1)} = \begin{cases} x^*, & \text{with probability } \min\{h(x^{(t)}, x^*), 1\} \\ x^{(t)}, & \text{otherwise} \end{cases}$$
 4. Set $t \leftarrow t+1$ and repeat 1.-3. m_j times
 5. Update $\tau_j = \alpha(\tau_{j-1})$ and $m_j = \beta(m_{j-1})$; set $j \leftarrow j+1$; go to 1

τ_j is temperature; function α should slowly decrease

Function β should be increasing

Simulated annealing

- Initially, also “bad” proposals are accepted
- With decreasing temperature, accept only improvements
- This helps to explore first and avoids convergence to a local minimum too early
- Algorithm has therefore chances to find the global optimum in presence of multiple local optima
- `method="SANN"` of R function `optim` is “a variant of simulated annealing” (documentation of `optim`)
 - Initial temperature seems to be important choice (can be changed e.g. by `control=list(temp=0.01)`; default 10 might be bad)

Markov Chain Monte Carlo (MCMC) – Metropolis algorithm

- Metropolis alg. (symmetric proposal $p(x^{(t)}|x^*) = p(x^*|x^{(t)})$):
- A starting value $x^{(0)}$ is generated from some starting distribution
- Given observation $x^{(t)}$, generate $x^{(t+1)}$ as follows:
 1. Sample candidate x^* from symmetric proposal dist. $p(\cdot|x^{(t)})$
 2. Compute ratio $R(x^{(t)}, x^*) = \frac{f(x^*)}{f(x^{(t)})}$
 3. Sample $x^{(t+1)}$ according to

$$x^{(t+1)} = \begin{cases} x^*, & \text{with probability } \min\{R(x^{(t)}, x^*), 1\} \\ x^{(t)}, & \text{otherwise} \end{cases}$$

4. If more observations needed, set $t \leftarrow t+1$; go to 1



Simulated annealing and Metropolis algorithm

- For fixed temperature τ , simulated annealing algorithm is a Metropolis algorithm (Metropolis et al., 1953)
- Kirkpatrick et al. (1983) proposed the name simulated annealing for using it as optimisation method

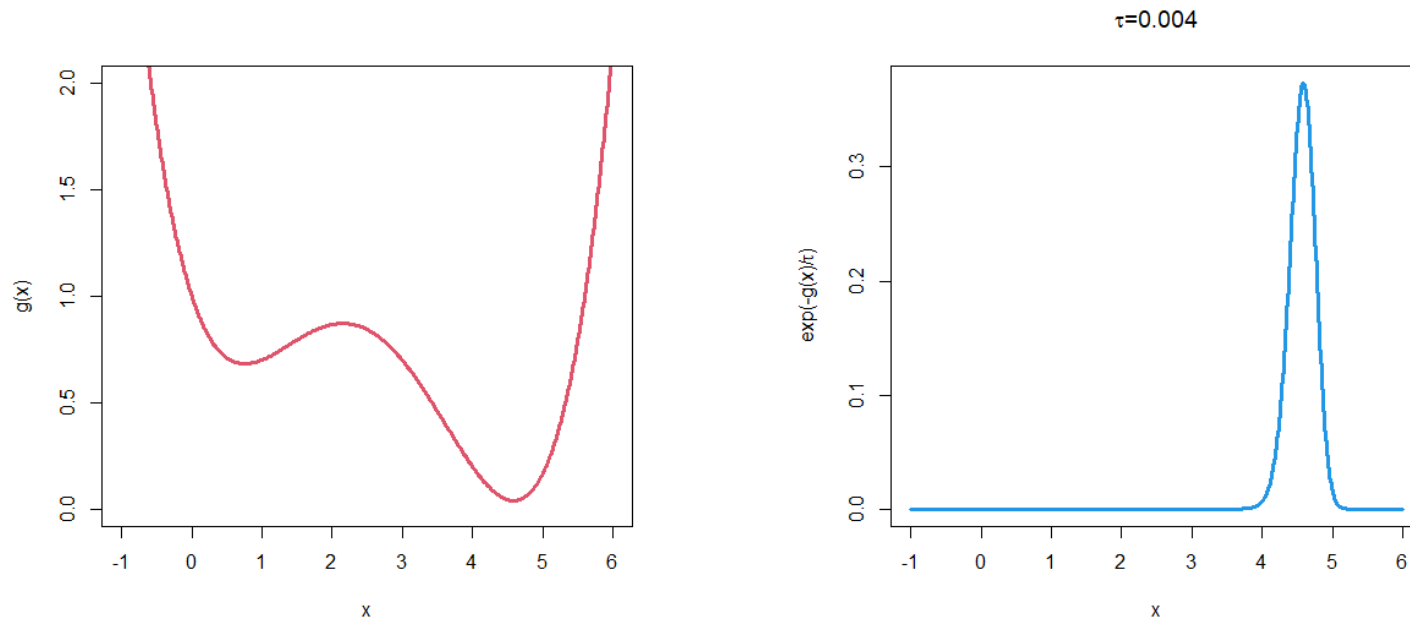
- $$h(x^{(t)}, x^*) = \exp\left(\frac{g(x^{(t)}) - g(x^*)}{\tau_j}\right) = \frac{\exp\left(-\frac{g(x^*)}{\tau_j}\right)}{\exp\left(-\frac{g(x^{(t)})}{\tau_j}\right)} = \frac{f(x^*)}{f(x^{(t)})} = R(x^{(t)}, x^*)$$

- Key ingredient of Metropolis and simulated annealing alg.: Markov chain $x^{(t)}$ **has stationary distribution f** (this means: if $x^{(t)}$ has distribution f , then $x^{(t+1)}$ has same distribution); for a proof see e.g. Koski (2009)
- Consequence: distribution of $x^{(t)}$ converges to f
- Requirement for all: $x^{(t)}$ irreducible and aperiodic chain



Simulated annealing: stationary distribution for fixed temperature τ

- Fixed temperature τ : Markov chain $x^{(t)}$ has stationary distribution with density proportional to $f(x) = \exp\left(-\frac{g(x)}{\tau}\right)$



Simulated annealing: proposal distribution

- Step 1 in simulated annealing iteration rule:
 1. Sample a candidate x^* from a proposal distribution $p(\cdot | x^{(t)})$
- Proposal distribution could be uniform distribution on a **neighbourhood** of $x^{(t)}$; for a unidimensional optimisation problem: `xs <- xt + runif(n=1, min=-1, max=1)`
- Instead of $\text{Unif}[-1,1]$, a distribution on a smaller or larger neighbourhood could be used
- But also, normal distribution $N(0, \sigma^2)$ or other **symmetric** distribution around 0 might be added to $x^{(t)}$ instead
- An option is to narrow the neighbourhood (e.g. to reduce σ^2 in the $N(0, \sigma^2)$ proposal) with decreasing temperature



Optimisation problem

- Generic optimisation problem:
 - x p -dimensional vector, $g: \mathbb{R}^p \rightarrow \mathbb{R}$ function
 - We search x^* with $g(x^*) = \min g(x)$
- Now, we consider also optimisation problems which cannot exactly be formulated according to the generic one
- Especially, function g might be defined on another space than \mathbb{R}^p
- Generalized optimisation problem:
 - x p -dimensional vector, $g: \mathcal{S} \rightarrow \mathbb{R}$ function for some set \mathcal{S}
 - We search x^* with $g(x^*) = \min g(x)$



Example: Multiple linear regression

- Generalized optimisation problem:
 - \mathbf{x} p -dimensional vector, $g: \mathcal{S} \rightarrow \mathbb{R}$ function for some set \mathcal{S}
 - We search \mathbf{x}^* with $g(\mathbf{x}^*) = \min g(\mathbf{x})$
- Multiple linear regression with q predictors
- Desired to choose best model based on criterion like AIC
- There are 2^q possible models
- If q small, AIC of all models can be computed (exhaustive search); for q larger, this is impossible (e.g. $q=50$, 1ms to compute an AIC \rightarrow more than 35 000 years needed!)
- One model can be represented as element of $\mathcal{S} = \{0, 1\}^q$ (1=predictor included in model, 0 otherwise)



Example: Multiple linear regression

- Generalized optimisation problem:
 - x p -dimensional vector, $g: \mathcal{S} \rightarrow \mathbb{R}$ function for some set \mathcal{S}
 - We search x^* with $g(x^*) = \min g(x)$
- Optimisation problem: Which model gives best AIC?
- Model 1: (1, 0, 0, 0, 1, 1, 0, 1, ...)
Model 2: (1, 1, 1, 0, 1, 1, 0, 0, ...)
- Which models are "close" to each other? (Need metric on $\mathcal{S} = \{0, 1\}^q$) What is a neighbourhood of a model?
- Apply simulated annealing with neighbourhood e.g. being all models which differ by one predictor (for proposal dist.)
- Uniform distribution on neighbourhood can be used



Example: Multiple linear regression

- Generalized optimisation problem:
 - \mathbf{x} p -dimensional vector, $g: \mathcal{S} \rightarrow \mathbb{R}$ function for some set \mathcal{S}
 - We search \mathbf{x}^* with $g(\mathbf{x}^*) = \min g(\mathbf{x})$
- Arbitrary starting model generated (e.g. uniform distribution on $\mathcal{S} = \{0, 1\}^q$, `$\mathbf{x}s \leftarrow \text{rbinom}(q, \text{size}=1, \text{prob}=0.5)$`)
- See example in Givens and Hoeting, Section 3.3, with 27 predictors

Convergence of simulated annealing

- Convergence proofs see the generated sequence either as sequence of homogeneous Markov chains (one for each τ) or as one inhomogeneous Markov chain
- For discrete $\mathcal{S} = \{x_1, x_2, x_3, \dots\}$ and g having a finite set M of global minima, simulated annealing converges with probability $1/|M|$ to each of the M global minima (references for proofs in Givens and Hoeting, 2013); main idea:
 - Stationary distribution proportional to: $\exp\left(-\frac{g(x)}{\tau}\right)$ or to $\exp\left(-\frac{g(x)-g_{min}}{\tau}\right)$ with $g_{min} = \min\{g(x)\}$
 - Therefore, if P is distribution according to stationary dist.,

$$P(x_i) = \underbrace{\exp\left(-\frac{g(x_i)-g_{min}}{\tau}\right)}_{\rightarrow 0 \text{ for } x_i \notin M} / \left\{ |M| + \underbrace{\sum_{x_j \notin M} \exp\left(-\frac{g(x_j)-g_{min}}{\tau}\right)}_{\rightarrow 0} \right\} \rightarrow \frac{1}{|M|} \quad (x_i \in M)$$

$\tau \rightarrow 0:$

$\rightarrow 0$ for $x_i \notin M$,
 $\rightarrow 1$ for $x_i \in M$

$\rightarrow 0$

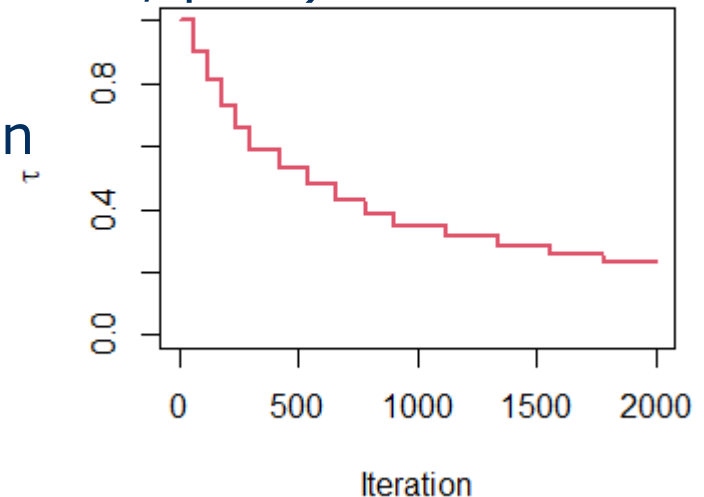


Convergence of simulated annealing

- To achieve convergence to a global minimum (possibly in the presence of local minima) in practise, one needs:
- Run iterations for each fixed temperature long enough such that convergence to the stationary distribution is achieved
- Cool the temperature slowly enough such that the iterations have time to escape from local minima
- Example from Givens and Hoeting (2013; p.73):

- 5 stages with 60 iterations, then
- 5 stages with 120 iterations, then
- 5 stages with 220 iterations
- From one stage to the next, τ is decreased by 10%,

$\mathbf{tau} \leftarrow 0.9 * \mathbf{tau}$; final τ is
 $0.9^{15} = 0.206 * \mathbf{initial} \tau$



Recall: Maximising information of experimental designs

- Regression model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$ (where $\boldsymbol{\varepsilon}$ has iid components)
- \mathbf{X} design matrix (depends on choice of observational points)
- Covariance matrix of Least Squares estimate $\hat{\boldsymbol{\beta}}$ is
$$\text{Cov}(\hat{\boldsymbol{\beta}}) = (\mathbf{X}^T \mathbf{X})^{-1} \cdot \text{const}$$
- Choose design of an experiment such that $\mathbf{X}^T \mathbf{X}$ "large"
- D-optimality: $g(\text{"design"}) = \det(\mathbf{X}^T \mathbf{X})$
- We search design^* with $g(\text{design}^*) = \max g(\text{design})$

Example: Maximising information of experimental designs

- Regression model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$, $\text{Cov}(\hat{\boldsymbol{\beta}}) = (\mathbf{X}^T \mathbf{X})^{-1} \cdot \text{const}$
- We search design^* with $g(\text{design}^*) = \max g(\text{design})$
- Example: cubic regression, $y = \beta_0 + \beta_1 w + \beta_2 w^2 + \beta_3 w^3 + \varepsilon$, w can be chosen in $[-1, 1]$, but practical circumstances require here a distance between design points of 0.05
- Therefore, we allow design points $\{-1, -0.95, -0.9, \dots, 1\}$ and at most one observation can be done at each point
- Each observation has a cost; and we want to minimise the penalized D-optimality $\# \text{observations} * 0.2 - \log(\det(\mathbf{X}^T \mathbf{X}))$

- $$\mathbf{X} = \begin{pmatrix} 1 & w_1 & w_1^2 & w_1^3 \\ 1 & w_2 & w_2^2 & w_2^3 \\ \dots & \dots & \dots & \dots \\ 1 & w_n & w_n^2 & w_n^3 \end{pmatrix}$$

Example: Maximising information of experimental designs

- Example: cubic regression, $y = \beta_0 + \beta_1 w + \beta_2 w^2 + \beta_3 w^3 + \varepsilon$, w can be chosen in $[-1, 1]$, but practical circumstances require here a distance between design points of 0.05
- Therefore, we allow design points $\{-1, -0.95, -0.9, \dots, 1\}$ and at most one observation can be done at each point
- A design can be represented by a vector in $\mathbb{S} = \{0, 1\}^{41}$ where 0 means that no observation is done at a design point and 1 means that one observation is made there
- How can a reasonable neighbourhood on \mathbb{S} look like here?

Simulated annealing

- + Very easy to implement
- + Theoretical property is good: theoretically, we can guarantee convergence to a global optimum even in the presence of local optima
- + Can even handle some non-standard optimisation problems
- In practice, convergence can be “maddeningly slow”
- One needs to play around with cooling schedule to ensure convergence in practice
 - We need to run the algorithm “long enough” at each temperature (to ensure stationary distribution)
 - We need to cool the temperature slowly enough (to allow escaping from local optima)



Genetic algorithms

Genetic algorithms

- Example: Optimisation problem $g: \mathcal{S} = \{0, 1\}^q \rightarrow \mathbb{R}$
- Like for PSO, several candidate solutions are considered in parallel at each iteration
- All candidate solutions at an iteration are called *generation*
- One candidate solution is called *individual* or *organism*
- An individual has its *chromosome*; a coordinate called *gene*
- Generation i (at iteration i):
 - Individual 1: (1, 0, 0, 0, 1, 1, 0, 1, 0, 1)
 - Individual 2: (1, 1, 1, 0, 1, 1, 0, 0, 0, 1)
 - ...
 - Individual n : (1, 0, 1, 0, 0, 1, 1, 0, 1, 1)
- Each individual has a *fitness* which is the objective function

Genetic algorithms

- Idea is to *select* parents (at random, but individuals with better fitness might have higher chance for being selected)
- Then, *genetic operator of crossover* is applied
- Finally, some genes might be *mutated*

- Generation t :
- Individual 1: (1, 0, 0, 0, 1, 1, 0, 1, 0, 1), fitness 10
- Individual 2: (1, 1, 1, 0, 1, 1, 0, 0, 0, 1), fitness 8.7
- Individual 3: (0, 1, 1, 1, 1, 1, 1, 0, 0, 0), fitness 4.5
- Individual 4: (1, 0, 1, 0, 0, 1, 1, 0, 1, 1), fitness 1.9

Genetic algorithms

- Generation t :
- Individual 1: (1, 0, 0, 0, 1, 1, 0, 1, 0, 1), fitness 10
Individual 2: (1, 1, 1, 0, 1, 1, 0, 0, 0, 1), fitness 8.7
Individual 3: (0, 1, 1, 1, 1, 1, 1, 0, 0, 0), fitness 4.5
Individual 4: (1, 0, 1, 0, 0, 1, 1, 0, 1, 1), fitness 1.9
- Selection (probability according to fitness):
- Individual 1: (1, 0, 0, 0, 1, 1, 0, 1, 0, 1), fitness 10
Individual 2: (1, 1, 1, 0, 1, 1, 0, 0, 0, 1), fitness 8.7
Individual 1: (1, 0, 0, 0, 1, 1, 0, 1, 0, 1), fitness 10
Individual 3: (0, 1, 1, 1, 1, 1, 1, 0, 0, 0), fitness 4.5

Genetic algorithms

- Selected parents:
- Individual 1: (1, 0, 0, 0, 1, 1, 0, 1, 0, 1), fitness 10
- Individual 2: (1, 1, 1, 0, 1, 1, 0, 0, 0, 1), fitness 8.7
- Individual 1: (1, 0, 0, 0, 1, 1, 0, 1, 0, 1), fitness 10
- Individual 3: (0, 1, 1, 1, 1, 1, 1, 0, 0, 0), fitness 4.5

- Crossover applied to first two and to last two:
- New indiv. 1: (1, 0, 0, 0, 1, 1, 0, 0, 0, 1)
- New indiv. 2: (1, 1, 1, 0, 1, 1, 0, 1, 0, 1)
- New indiv. 3: (1, 0, 0, 1, 1, 1, 1, 0, 0, 0)
- New indiv. 4: (0, 1, 1, 0, 1, 1, 0, 1, 0, 1)

Genetic algorithms

- New individuals after crossover:
- New indiv. 1: (1, 0, 0, 0, 1, **1**, 0, 0, 0, 1)
- New indiv. 2: (1, 1, 1, 0, 1, 1, 0, 1, 0, 1)
- New indiv. 3: (1, 0, **0**, 1, 1, 1, 1, **0**, 0, 0)
- New indiv. 4: (0, 1, 1, 0, 1, 1, 0, 1, 0, 1)
- Some genes are mutated:
- New indiv. 1: (1, 0, 0, 0, 1, **0**, 0, 0, 0, 1)
- New indiv. 2: (1, 1, 1, 0, 1, 1, 0, 1, 0, 1)
- New indiv. 3: (1, 0, **1**, 1, 1, 1, 1, **1**, 0, 0)
- New indiv. 4: (0, 1, 1, 0, 1, 1, 0, 1, 0, 1)
- This is now generation $t+1$
- Again fitness calculation, selection, crossover, and mutation is done; etc.

Genetic algorithms

- Size of population: Givens and Hoeting write about 10-200 and another rule for binary genes, it should be between 1 and 2 times the chromosome length
- Selection: randomly, e.g. probability proportional to fitness
- Crossover: Pick randomly a position where the chromosome of two parents is splitted (e.g. uniform distribution on all $q-1$ possible split positions)
- Mutation: Change each gene with probability μ , independently; μ should not be too small and not too large; e.g. $\mu=0.01$ in example from Givens and Hoeting (2013)

Example: classification

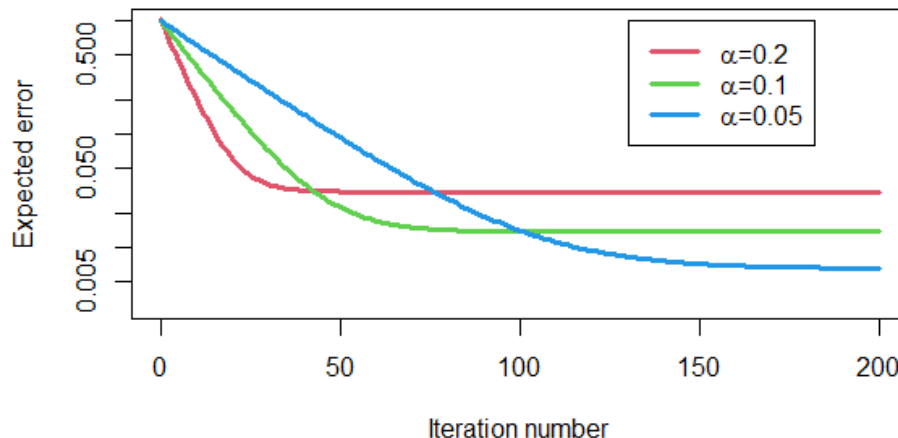
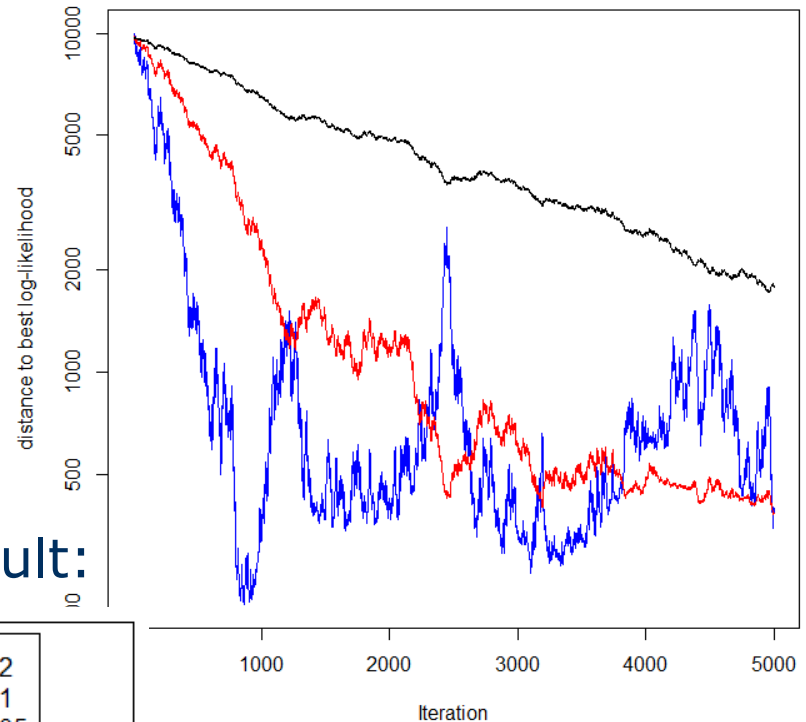
- A dataset $\{x_1, x_2, \dots, x_q\}$ is supposed to be divided into p groups (subsets) such that some criterion is minimized, e.g. the within-group sum of squares
- This is a classification problem from unsupervised learning
- If $p=2$, we can optimize over $\mathcal{S} = \{0, 1\}^q$ and apply simulated annealing or a genetic algorithm
- If $p>2$, we can apply the algorithms as well using $\mathcal{S} = \{1, 2, \dots, p\}^q$, i.e. individuals have chromosomes like (8, 4, 2, 2, 3, 8, 4, 1, 7, 2, ...)

About theoretical results

- SGD:
$$\mathbb{E} \|\mathbf{x}^{(t)} - \mathbf{x}^*\|_2^2 \leq \frac{\alpha s}{m\{1 - \alpha \max(L_i)\}} + (1 - \alpha m\{1 - \alpha \max(L_i)\})^t \|\mathbf{x}^{(0)} - \mathbf{x}^*\|_2^2$$
- PSO: $0 < c < \frac{12(w^2 - 1)}{5w - 7}$

Stochastic gradient descent method – empirical examples from OASI-L2

- Constant step size $\alpha^{(t)} = \alpha$
- Step size
 - $\alpha = 0.0006$ (black)
 - $\alpha = 0.002$ (red)
 - $\alpha = 0.006$ (blue)
- Compare with theoretical result:



Theoretical behaviour of bound in Theorem 2 for expected distance to optimum for $s=0.5$, $m=2$, $\max(L_i)=2$, $\varepsilon_0=1$

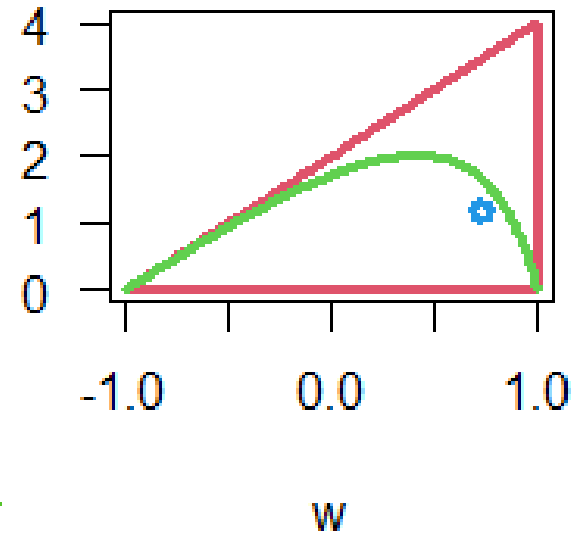


Particle swarm optimisation – stability analyses

- $c = c_1 = c_2$
- $-1 < w < 1$ and
 $0 < c < 2(w + 1)$

- Sequence $(z^{(t+1)})$ is order-2 stable if:
 $-1 < w < 1$ and

$$0 < c < \frac{12(w^2 - 1)}{5w - 7}$$



- Default in R-package `psco` based on Clerc and Kennedy (2002):

$$w = \frac{1}{2 \ln(2)} = 0.721, \quad c = c_1 = c_2 = \frac{1}{2} + \ln(2) = 1.193$$



References

- Clerc M (2016). Chapter 8: Particle swarms. In: *Metaheuristics*. (Siarry P ed.).
- Givens GH, Hoeting JA (2013). *Computational Statistics*, 2nd edition. John Wiley & Sons, Inc., Hoboken, New Jersey.
- Kirkpatrick S, Gelatt CD, Vecchi MP (1983). Optimization by Simulated Annealing. *Science*, 220(4598):671–680.
- Koski T (2009). *Sf2955 Computer intensive methods MCMC – On the Discrete Metropolis-Hastings Algorithm*. KTH, Stockholm.
<https://www.math.kth.se/matstat/gru/sf2955/metropolis3.pdf>
- Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller E (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21, 1087–1092.