

# Computational statistics, lecture 6

Frank Miller, Department of Computer and Information Science,  
Linköping University  
February 24, 2026

# Today

(Literature: Givens and Hoeting, 3.3, 4.1-4.2.2; Gentle, 6.3, 6.6, 14.3)

- EM algorithm
- Stochastic optimization
  - Simulated annealing
  - Stochastic Gradient Ascent
- Combinatorial optimization
- Optimization problem:
  - $\mathbf{x}$   $p$ -dimensional vector,  $g: \mathbb{R}^p \rightarrow \mathbb{R}$  function
  - We search  $\mathbf{x}^*$  with  $g(\mathbf{x}^*) = \max g(\mathbf{x})$

$$g: \mathcal{S} \rightarrow \mathbb{R}$$

# EM algorithm

- EM = “Expectation-Maximization”
- Main application of this algorithm is in cases where not all data is observed
- E: Expectation will be taken over all (unobserved) data which lead to the observed data
- Algorithm is iterative:  
each iteration has an E step, followed by an M step

# EM algorithm

- Let  $X$  be observed data,  $Y$  complete data,  $\theta$  unknown parameter-vector,  $L(\theta|x)$  likelihood to be maximized
- Iteration  $t$  ( $t = 0, 1, \dots$ ):  $\theta^{(t)}$
- Let  $Q(\theta|x; \theta^{(t)}) = E\{\log L(\theta|Y)|x; \theta^{(t)}\}$  be expectation of **log likelihood for complete data** conditional on observed data  $X = x$
- EM algorithm:
  1. Initialize parameter-vector with a guess  $\theta^{(0)}$ ,  $t = 0$
  - 2. E step:** Compute  $Q(\theta|x; \theta^{(t)})$
  - 3. M step:** Maximize  $Q(\theta|x; \theta^{(t)})$  with respect to  $\theta$  -> result is  $\theta^{(t+1)}$
  4. If not stopping criterion (e.g.  $(\theta^{(t+1)} - \theta^{(t)})^T (\theta^{(t+1)} - \theta^{(t)}) < \epsilon$ ) met, set  $t \leftarrow t+1$ , and go back to E step

# EM algorithm: Example

- Effect of a drug to be measured and  $n$  patients (randomly chosen out of a population of patients) treated with the drug
- $X_i, i = 1, \dots, n$ , observed for each patient after drug-treatment
- Known that population consists of two groups:
  - One group responds well to the drug (i.e., larger  $X_i$ )
  - Another group responds only barely (smaller  $X_i$ )
- It is not known which patient belongs to which group

**Observed:**  $X_i$ ,

**Unobserved:**  $Z_i = \begin{cases} 1, & \text{if patient } i \text{ belongs to responder group} \\ 0, & \text{otherwise} \end{cases}$

**Complete data:**  $Y_i = (X_i, Z_i)$

# EM algorithm: Example

- In this example, we assume that  $X_i$  has normal mixture density  $f$  for  $c = 2$  groups (responder, non-responder)

- Generally, a normal mixture (also called GMM, Gaussian mixture model) has density  $f$  being sum of  $c$  weighted densities:

$$f(x) = \sum_{i=1}^c p_i \varphi(x; \mu_i; \sigma_i),$$

where  $p_i$  are weight or mixing coefficients ( $p_i > 0$ ;  $p_1 + \dots + p_c = 1$ ), and  $\varphi(x; \mu; \sigma)$  being density of  $N(\mu, \sigma^2)$

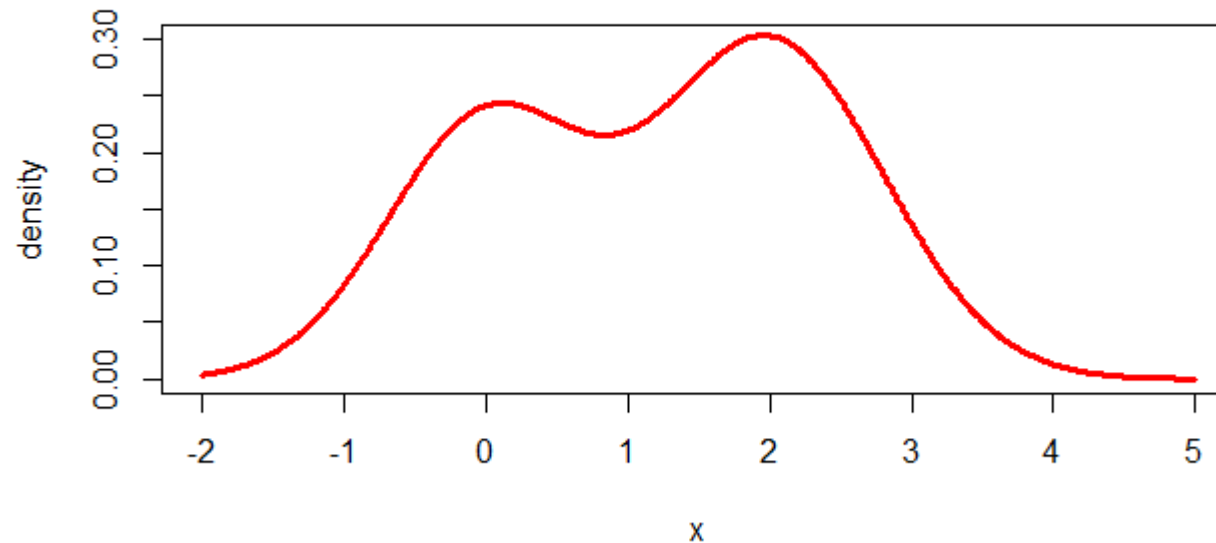
- Here for  $c = 2$  groups ( $p = p_1$ ,  $p_2 = 1 - p$ ):

$$f(x) = p\varphi(x; \mu_1; \sigma_1) + (1 - p)\varphi(x; \mu_2; \sigma_2)$$

- 5 parameters to estimate from data:  $p$ ;  $\mu_1$ ;  $\sigma_1$ ;  $\mu_2$ ;  $\sigma_2$

# EM algorithm: Example

- $f(x) = p\varphi(x; \mu_1; \sigma_1) + (1 - p)\varphi(x; \mu_2; \sigma_2)$
- Parameters:  $p; \mu_1; \sigma_1; \mu_2; \sigma_2$



- Example here:  $p = 0.4; \mu_1 = 0; \sigma_1 = 0.7; \mu_2 = 2; \sigma_2 = 0.8$

# EM algorithm for normal mixtures

- The estimated probability that observation  $j$  belongs to group  $i$  (of  $c$  groups) is

$$\hat{\pi}_{ij} = \frac{\hat{p}_i \varphi(x_j; \hat{\mu}_i; \hat{\sigma}_i)}{\sum_{k=1}^c \hat{p}_k \varphi(x_j; \hat{\mu}_k; \hat{\sigma}_k)},$$

where

$\varphi(\cdot; \mu; \sigma)$  is density of normaldist. with mean  $\mu$  and sd  $\sigma$

- Model parameters maximizing  $Q = E\{\log L(\boldsymbol{\theta}|Y)|\mathbf{x}; \boldsymbol{\theta}^{(t)}\}$  are:

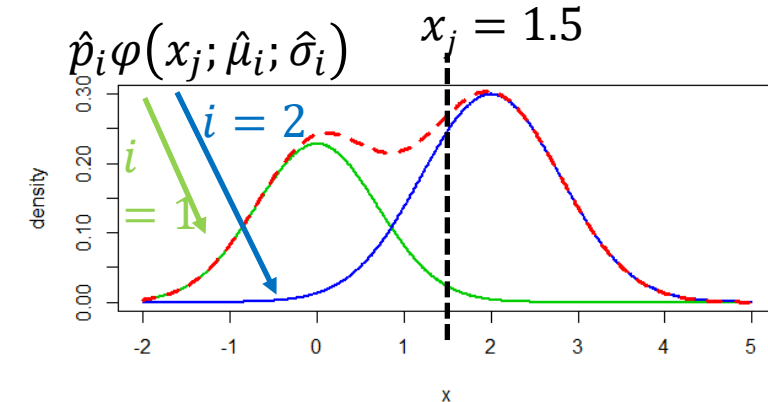
$$\boldsymbol{\theta} = (p_1, \dots, p_c, \mu_1, \dots, \mu_c, \sigma_1, \dots, \sigma_c)$$

$$\boldsymbol{\theta}^{(t)} = (\hat{p}_1, \dots, \hat{p}_c, \hat{\mu}_1, \dots, \hat{\mu}_c, \hat{\sigma}_1, \dots, \hat{\sigma}_c)$$

Multivariate case similar, except:

$$\hat{\boldsymbol{\mu}}_i = \frac{1}{\hat{p}_i n} \sum_{j=1}^n \hat{\pi}_{ij} \cdot \mathbf{x}_j,$$

$$\hat{\boldsymbol{\Sigma}}_i = \frac{1}{\hat{p}_i n} \sum_{j=1}^n \hat{\pi}_{ij} \cdot (\mathbf{x}_j - \hat{\boldsymbol{\mu}}_i)(\mathbf{x}_j - \hat{\boldsymbol{\mu}}_i)^T$$



**E step**

**M step**

$$\hat{p}_i = \frac{1}{n} \sum_{j=1}^n \hat{\pi}_{ij},$$

$$\hat{\mu}_i = \frac{1}{\hat{p}_i n} \sum_{j=1}^n \hat{\pi}_{ij} \cdot x_j,$$

$$\hat{\sigma}_i^2 = \frac{1}{\hat{p}_i n} \sum_{j=1}^n \hat{\pi}_{ij} \cdot (x_j - \hat{\mu}_i)^2$$

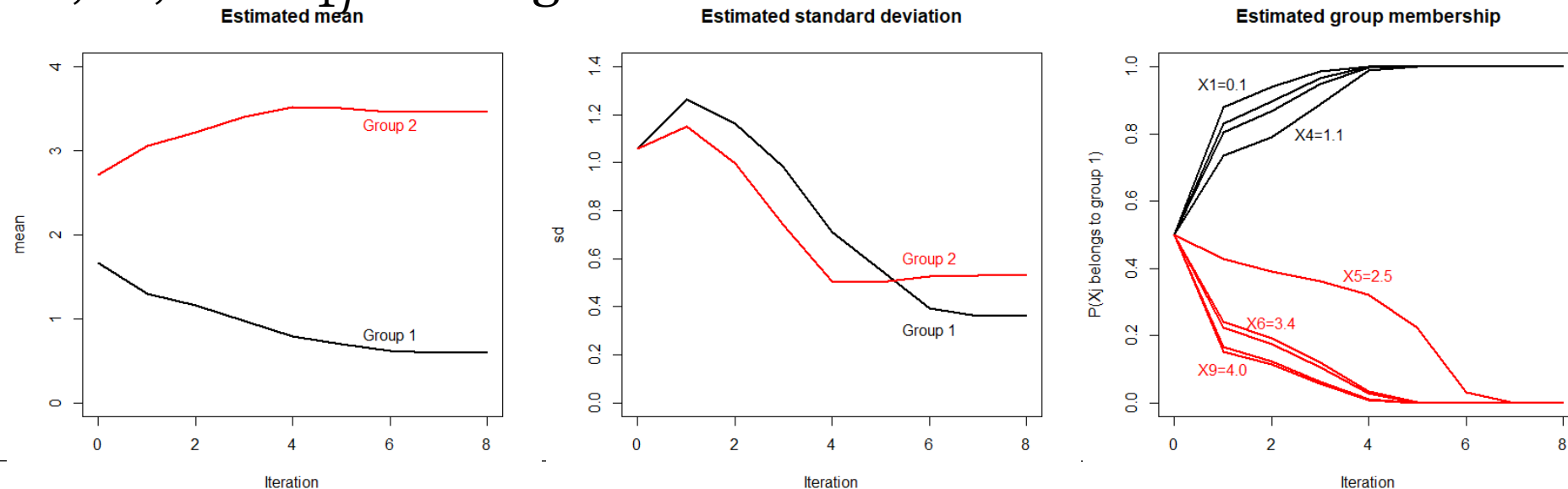
$$Q = \sum_{i=1}^c \sum_{j=1}^n \hat{\pi}_{ij} \{ \log(\hat{p}_i) + \log \varphi(x_j; \hat{\mu}_i; \hat{\sigma}_i) \}$$

- See Section (10.1 and) 10.2 of [Lindholm, Wahlström, Lindsten, Schön \(2022\)](#)



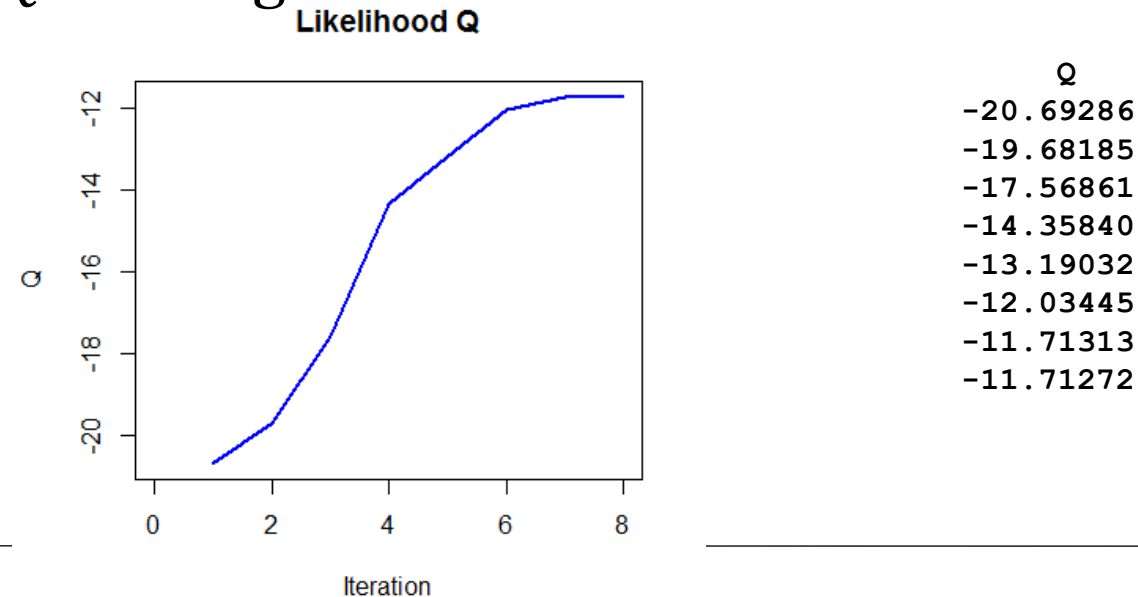
# EM algorithm for normal mixtures

- Example for illustration:  $n = 9$  observations obtained. Ordered data: 0.1, 0.5, 0.7, 1.1, 2.5, 3.4, 3.5, 3.9, 4.0
- EM algorithm terminates after 8 iterations with:  
 $(p_1^{(8)}, \mu_1^{(8)}, \mu_2^{(8)}, \sigma_1^{(8)}, \sigma_2^{(8)}) = (0.444, 0.600, 3.460, 0.361, 0.532)$
- Mean, sd, and  $\hat{\pi}_{1j}$  converge as follows:



# EM algorithm for normal mixtures

- Example for illustration:  $n = 9$  observations obtained. Ordered data: 0.1, 0.5, 0.7, 1.1, 2.5, 3.4, 3.5, 3.9, 4.0
- EM algorithm terminates after 8 iterations with:  
 $(p_1^{(8)}, \mu_1^{(8)}, \mu_2^{(8)}, \sigma_1^{(8)}, \sigma_2^{(8)}) = (0.444, 0.600, 3.460, 0.361, 0.532)$
- Over the iterations,  $Q$  converges as follows:



```

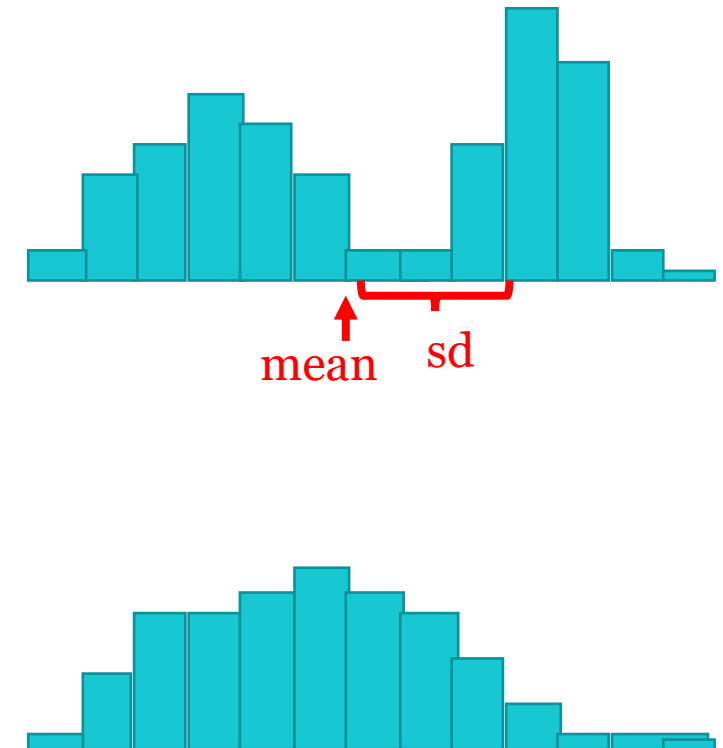
emalg <- function(dat, eps=0.000001){
  n      <- length(dat)
  pi      <- rep(NA, n)      # initialize vector for prob. to belong to group 1
  # define reasonable starting values for parameters; here based on summary statistics for total dataset
  p      <- 0.5              # starting value for mixing parameter
  sigma1 <- sd(dat)*2/3      # starting value for standard deviation in group taken as 2/3 of total sd
  sigma2 <- sigma1
  mu1     <- mean(dat)-sigma1/2 # starting values for means, taken a bit lower and a bit higher than overall mean
  mu2     <- mean(dat)+sigma1/2
  pv      <- c(p, mu1, mu2, sigma1, sigma2) # parameter vector
  cc      <- eps + 100        # initialize convergence criterion to avoid stopping the while-loop directly
  while (cc>eps){
    pv1 <- pv                # save previous parameter vector
    ### E step ###
    for (j in 1:n){
      pi1 <- p*dnorm(dat[j], mean=mu1, sd=sigma1)
      pi2 <- (1-p)*dnorm(dat[j], mean=mu2, sd=sigma2)
      pi[j] <- pi1/(pi1+pi2)
    }
    ### M step ###
    p      <- mean(pi)
    mu1     <- sum(pi*dat)/(p*n)
    mu2     <- sum((1-pi)*dat)/((1-p)*n)
    sigma1 <- sqrt(sum(pi*(dat-mu1)*(dat-mu1)/(p*n)))
    sigma2 <- sqrt(sum((1-pi)*(dat-mu2)*(dat-mu2)/((1-p)*n)))
    #####
    pv      <- c(p, mu1, mu2, sigma1, sigma2)
    cc      <- t(pv-pv1)%*(pv-pv1) # a convergence criterion, maybe not the best one
  }
  pv
}

data <- c(0.1, 0.5, 0.7, 1.1, 2.5, 3.4, 3.5, 3.9, 4.0) # example data and run of the algorithm
emalg(data)

```

# Choice of starting values in example before

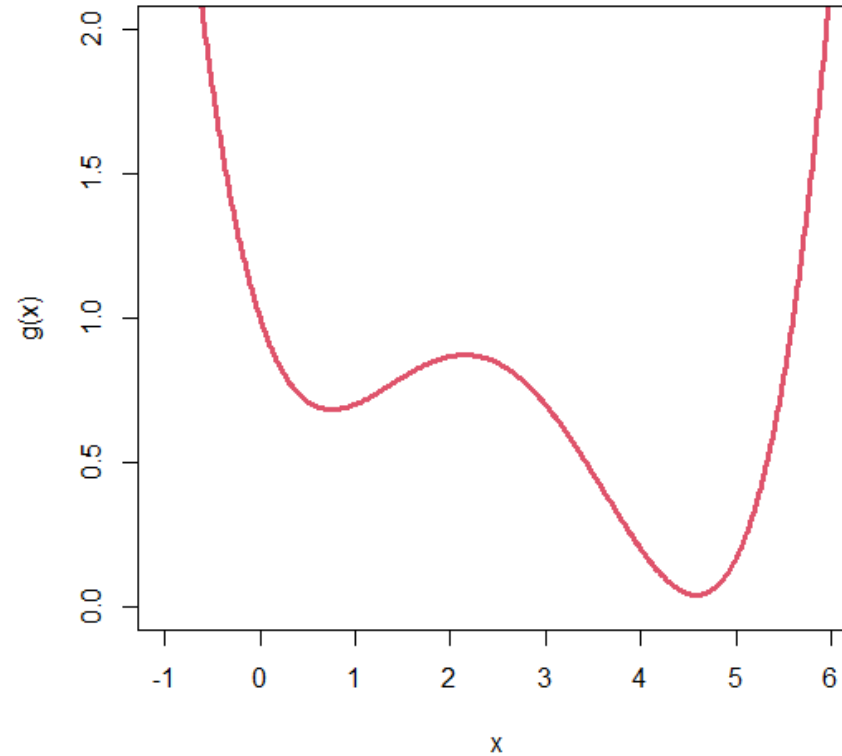
- We want to create automatically starting values which are meaningful for the data
- My heuristic rule to choose them in the R-code before:
  - Take total data and compute overall mean and sd
  - Overall sd is usually larger than sd's for groups
  - Therefore, I took  $2/3^*$  overall sd as starting values for the sd in both groups
  - For group means, starting values with 1 sd difference chosen



# Choosing starting values, connection to K-means

- We can look at the data and guess the components in the mixture, their mean and variance
- We can use a heuristic rule to determine starting values (like in the example before)
- We can try a grid of starting parameter values
- Specifically, for the EM algorithm for normal mixtures, we can first run a classification algorithm and use its result as start for the EM algorithm
- Note (cp. course Machine Learning & Sec. 10.2 of [Lindholm et al., 2022](#)):  
The K-means algorithm can be seen as special case of the EM algorithm for normal mixtures when the variances tend to 0

# Simulated annealing



[AlphaOpt \(2017\). Introduction To Optimization: Gradient Free Algorithms \(2/2\) – Simulated Annealing, Nelder-Mead \(0:15-1:35\)](#)

# Simulated annealing

- Start value  $x^{(0)}$ ; stage  $j = 0, 1, 2, \dots$  has  $m_j$  iterations; initial temperature  $\tau_0$ ; 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  $h(x^{(t)}, x^*) = \exp\left(\frac{g(x^*) - g(x^{(t)})}{\tau_j}\right)$

$g(x^{(t)}) - g(x^*)$   
for  
minimisation

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  $\mathbf{t} \leftarrow \mathbf{t} + 1$  and repeat 1.-3.  $m_j$  times

5. Update  $\tau_j = \alpha(\tau_{j-1})$  and  $m_j = \beta(m_{j-1})$ ; set  $\mathbf{j} \leftarrow \mathbf{j} + 1$ ; go to 1

$\tau_j$  is temperature; function  $\alpha$  should slowly decrease it; function  $\beta$  should be increasing

# MCMC – Metropolis algorithm

From Lecture 4

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

Special case when  
 $g$  is symmetric:  
 $g(x^*|x^{(t)}) = g(x^{(t)}|x^*)$

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

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



# Simulated annealing

- Initially, also “bad” proposals are accepted
- With decreasing temperature, accept only improvements
- This helps to explore first and avoids convergence to a local maximum too early
- Algorithm which 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 can be important choice (can be changed e.g. by **control=list(temp=0.01)**; default 10 might be bad)

# 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 **neighborhood** 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 neighborhood could be used
- But also, normal distribution  $N(0, \sigma^2)$  or other symmetric distribution around 0 might be added to  $x^{(t)}$  instead
- For multidimensional cases, one could use iid components, a uniform distribution on a ball around  $\mathbf{x}^{(t)}$ , or a multivariate normal distribution with mean  $\mathbf{x}^{(t)}$

# Combinatorial optimization

- Generic optimization problem:
  - $\mathbf{x}$   $p$ -dimensional vector,  $g: \mathbb{R}^p \rightarrow \mathbb{R}$  function
  - We search  $\mathbf{x}^*$  with  $g(\mathbf{x}^*) = \max g(\mathbf{x})$
- Now, we consider also optimization 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 optimization problem:
  - $\mathbf{x}$   $p$ -dimensional vector,  $g: \mathbb{S} \rightarrow \mathbb{R}$  function for some set  $\mathbb{S}$
  - We search  $\mathbf{x}^*$  with  $g(\mathbf{x}^*) = \max g(\mathbf{x})$

# Example: Multiple linear regression

- Generalized optimization problem:
  - $\mathbf{x}$   $p$ -dimensional vector,  $g: \mathcal{S} \rightarrow \mathbb{R}$  function for some set  $\mathcal{S}$
  - We search  $\mathbf{x}^*$  with  $g(\mathbf{x}^*) = \max g(\mathbf{x})$
- 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

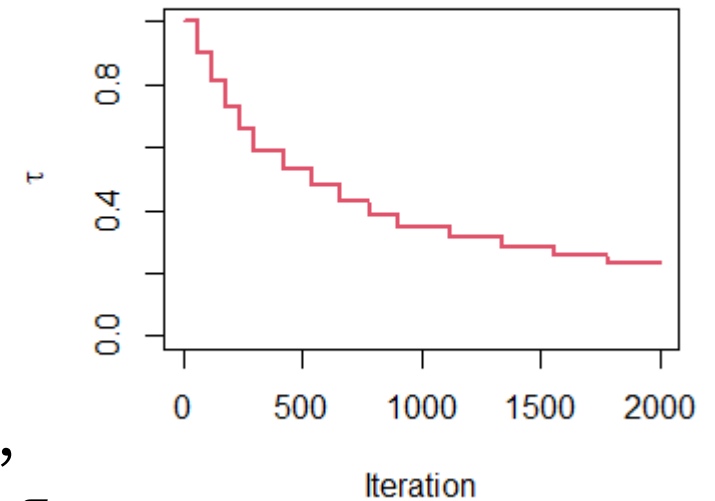
- Generalized optimization problem:
  - $\mathbf{x}$   $p$ -dimensional vector,  $g: \mathcal{S} \rightarrow \mathbb{R}$  function for some set  $\mathcal{S}$
  - We search  $\mathbf{x}^*$  with  $g(\mathbf{x}^*) = \max g(\mathbf{x})$
- Optimization problem: Which model gives best AIC?
- Model 1: (1, 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 neighborhood of a model?
- Apply simulated annealing e.g. with neighborhood being all models which differ by one predictor (for proposal dist.)
- Uniform distribution on neighborhood can be used

# Example: Multiple linear regression

- Generalized optimization problem:
  - $\mathbf{x}$   $p$ -dimensional vector,  $g: \mathcal{S} \rightarrow \mathbb{R}$  function for some set  $\mathcal{S}$
  - We search  $\mathbf{x}^*$  with  $g(\mathbf{x}^*) = \max g(\mathbf{x})$
- Arbitrary starting model generated (e.g. uniform distribution on  $\mathcal{S} = \{0, 1\}^q$ ,  
`xs <- rbinom(q, size=1, prob=0.5)`)
- See example in Givens and Hoeting (2013), Section 3.3, with 27 predictors

# Convergence of simulated annealing

- To achieve convergence to a global optimum (possibly in presence of local optima) in theory, one needs:
  - Run iterations for each fixed temperature long enough such that convergence to stationary distribution achieved
  - Cool temperature slowly enough such that iterations have time to escape from local optima
- 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,  $\tau$  is decreased by 10%,  
`tau <- 0.9*tau`; final  $\tau$  is  $0.9^{15} = 0.206$ \*initial  $\tau$



# Simulated annealing: + and -

- +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 optimization problems
- In practice, convergence can be “maddeningly slow”
- One needs to play around with cooling schedule to ensure convergence in practice



# Steepest ascent (steepest descent)

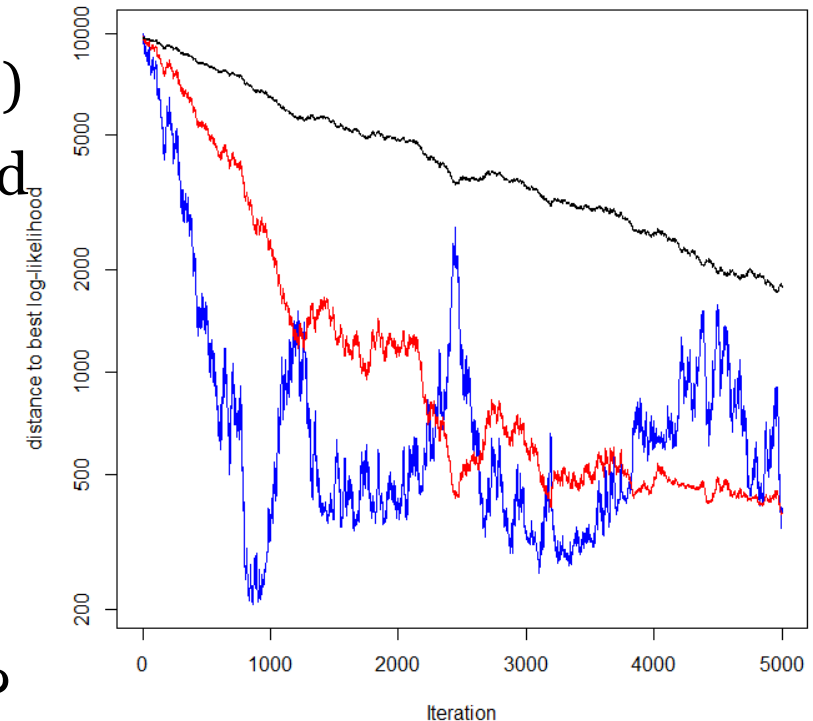
- Iteration:  $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \alpha^{(t)} \mathbf{g}'(\mathbf{x}^{(t)})$
- Optimization problem (finite sum case):
  - $\mathbf{x}$   $p$ -dimensional vector,  $g_i: \mathbb{R}^p \rightarrow \mathbb{R}$  functions
  - We search  $\mathbf{x}^*$  with  $g(\mathbf{x}^*) = \max g(\mathbf{x})$ , where  $g = \sum_{i=1}^n g_i$
- If  $n$  large: Takes time to evaluate gradient  $\mathbf{g}' = \sum_{i=1}^n \mathbf{g}'_i$

# Stochastic steepest ascent

- Iteration:
  - Choose  $i \in \{1, \dots, n\}$  randomly
  - $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \alpha^{(t)} \mathbf{g}'_i(\mathbf{x}^{(t)})$
- $\alpha^{(t)}$  is a predefined sequence, either
  - constant step size  $\alpha^{(t)} = \alpha$  or
  - decreasing step size e.g.  $\alpha^{(t)} = \alpha/t$
- Convergence (to a local optimum) can be shown if step size fullfills  $\sum_{t=1}^{\infty} \alpha^{(t)} = \infty$  and  $\sum_{t=1}^{\infty} (\alpha^{(t)})^2 < \infty$  (example:  $\alpha^{(t)} = \alpha/t$ )

# Stochastic steepest ascent: step size, runtime

- Example: Two-parameter MLE comp. ( $n=1\,000\,000$ )
- We monitor distance of current to max. log likelihood
- **Here constant step size**  $\alpha^{(t)} = \alpha$
- Choice of step size is critical
- $\alpha = 0.0006$ ,  $\alpha = 0.002$ ,  $\alpha = 0.006$
- Which step size is best,
  - if you have time for 5000 iterations?
  - if you have time for 500, or for 50000 iterations?
- Stochastic steepest ascent: 50000 iterations took 7 s
- Steepest ascent with alpha-halving: 112 iterations took 52 s
- 3320 iterations of stochastic steepest ascent same time as 1 iteration of steepest ascent



# Stochastic steepest ascent: mini-batches

- Instead of sampling a single  $i$ , a batch of size  $m$  can be sampled in each iteration
- Iteration:
  - Choose  $\{i_1, \dots, i_m\} \subseteq \{1, \dots, n\}$  randomly
  - $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \alpha^{(t)} \sum_{j=1}^m \mathbf{g}'_{i_j}(\mathbf{x}^{(t)})$
- Decreases risk of large random oscillations
- Especially interesting when algorithm performed on a parallel computer