UiO : Matematisk institutt
Det matematisk-naturvitenskapelige fakultet

STK-4051/9051 Computational Statistics Spring 2024
Ex2

Instructor: Odd Kolbjørnsen, oddkol@math.uio.no

Exercise 4 (Variable selection and neighborhood)
Assume we have a linear regression model

$$Y = \beta_0 + \sum_{j:j\in\mathcal{M}} \beta_j x_j + \varepsilon$$

where $M \subset \{1, ..., p\}$. Our aim is to find the best subset M based on some data $\{(y_i, \boldsymbol{x}_i), i = 1, ..., n\}$ and some performance criterion (e.g. AIC).

For many of the optimization methods discussed, a neighborhood of a current solution is needed. We will look at different choices of neighborhoods in this exercise.

(a). Introduce the p-vector $\boldsymbol{\theta}$ where $\theta_j = 1$ if $j \in M$ and 0 otherwise. Argue that there is a one-to-one correspondence between M and $\boldsymbol{\theta}$.

(a). Assume we have M. Then we put $\theta_j = 1$ for those $j \in M$ and zero for the rest, defining $\boldsymbol{\theta}$.

Assume we have $\boldsymbol{\theta}$. Then we include into all M those j's for which $\theta_j = 1$, defining M.

(*b*). We will now consider four different neighborhoods:

$$\mathcal{N}_1(\boldsymbol{\theta}) = \{\boldsymbol{\theta}^*; \exists k \text{ such that } \theta_j^* = \theta_j \text{ for all } j \neq k \text{ and } \theta_k^* \neq \theta_k\}$$
$$\mathcal{N}_2(\boldsymbol{\theta}) = \{\boldsymbol{\theta}^*; \exists k, k' \text{ such that } \theta_j^* = \theta_j \text{ for all } j \neq k, k', \theta_k^* \neq \theta_k \text{ and } \theta_{k'}^* \neq \theta_{k'}\}$$
$$\mathcal{N}_3(\boldsymbol{\theta}) = \{\boldsymbol{\theta}^*; \exists k, k' \text{ such that } \theta_j^* = \theta_j \text{ for all } j \neq k, k', \theta_{k'}^* = \theta_k \text{ and } \theta_k^* = \theta_{k'}\}$$
$$\mathcal{N}_4(\boldsymbol{\theta}) = \mathcal{N}_1(\boldsymbol{\theta}) \cup \mathcal{N}_3(\boldsymbol{\theta})$$

In each case, answer the following questions:

- What are the sizes of the neighborhoods?
- Do all solutions in Θ communicate?
- If the solutions communicate, what is the maximum of the number of moves needed to move between two arbitrary solutions?

Max cardinality, mo
complex , consider
changes to be mad
on any theta totally

Max cardinality,

(*b*). We have:

- The sizes of the neighborhoods are p, $p(p-1)/2$, $p(p-1)/2$ and $p + p(p-1)/2$, respectively.
- The first one communicate, the second do not and the third one does not allow the number of "active" components to change and therefore do not communicate. The last one communicate since the first one does.
- For $\mathcal{N}_1$, the maximum number of moves is p. For $\mathcal{N}_4$, in order to move from $(00\cdots0)$ to $(11\cdots1)$ we need p moves (all in $\mathcal{N}_1$), giving also p necessary moves in this case.

(c). Consider a case where $p = 3$ and we want to maximize $f(\boldsymbol{\theta})$ where

$\boldsymbol{\theta}$	$f(\boldsymbol{\theta})$
000	0.5
001	0.2
010	0.2
100	0.1
011	2.0
101	2.1
110	0.5
111	1.5

$$\mathcal{N}_1(\boldsymbol{\theta}) = \{\boldsymbol{\theta}^*; \exists k \text{ such that } \theta_j^* = \theta_j \text{ for all } j \neq k \text{ and } \theta_k^* \neq \theta_k\}$$
$$\mathcal{N}_2(\boldsymbol{\theta}) = \{\boldsymbol{\theta}^*; \exists k, k' \text{ such that } \theta_j^* = \theta_j \text{ for all } j \neq k, k', \theta_k^* \neq \theta_k \text{ and } \theta_{k'}^* \neq \theta_{k'}\}$$
$$\mathcal{N}_3(\boldsymbol{\theta}) = \{\boldsymbol{\theta}^*; \exists k, k' \text{ such that } \theta_j^* = \theta_j \text{ for all } j \neq k, k', \theta_{k'}^* = \theta_k \text{ and } \theta_k^* = \theta_{k'}\}$$
$$\mathcal{N}_4(\boldsymbol{\theta}) = \mathcal{N}_1(\boldsymbol{\theta}) \cup \mathcal{N}_3(\boldsymbol{\theta})$$

Which of the methods covered in chapter 3 in the book (steepest ascent, simulated annealing, genetic algorithms, tabu algorithms) will be able to find the global optimum based on the first three different neighborhoods?

Discuss the pros and cons for the different neighborhoods.

(c). Steepest ascent: If we start on 000, all solutions within $\mathcal{N}_1$ have lower values, so 000 is a local mode that we are not able to escape from. However, for $\mathcal{N}_2$ we are able to move out of this mode.

Simulated annealing: If we just use a neighbourhood that communicate, all possibilities are available.

Genetic algorithms: As long as we include mutations that make all solutions communicate, the algorithms is able to find the solution.

Tabu algorithms: For $\mathcal{N}_1$ and as long as the memory is smaller than the size of the neighborhoud, this is ok.

3.1. Implement a random starts local search algorithm for minimizing the AIC for the baseball salary regression problem. Model your algorithm after Example 3.3.

a. Change the move strategy from steepest descent to immediate adoption of the first randomly selected downhill neighbor.

b. Change the algorithm to employ 2-neighborhoods, and compare the results with those of previous runs.

AIC is used for model selection where we also account for different numbers of parameters, want AIC to be small

$$AIC(M) = -2\log L(\theta|\boldsymbol{x}, M) + 2p$$

Term giving penalty for «bad match» to data

Term giving penalty for number of parameters in *M*

Intuition: if you have a model with many parameters, you will fit data better, but you are in danger of overfitting
Thus we need to penalize model complexity

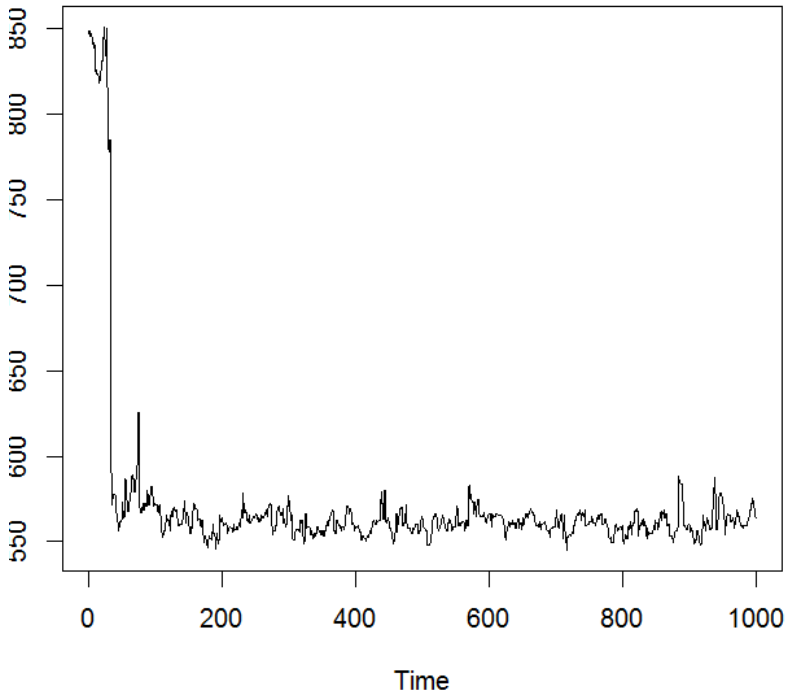
3.3. Implement simulated annealing for minimizing the AIC for the baseball salary regression problem. Model your algorithm on Example 3.4.

a. Compare the effects of different cooling schedules (different temperatures and different durations at each temperature).

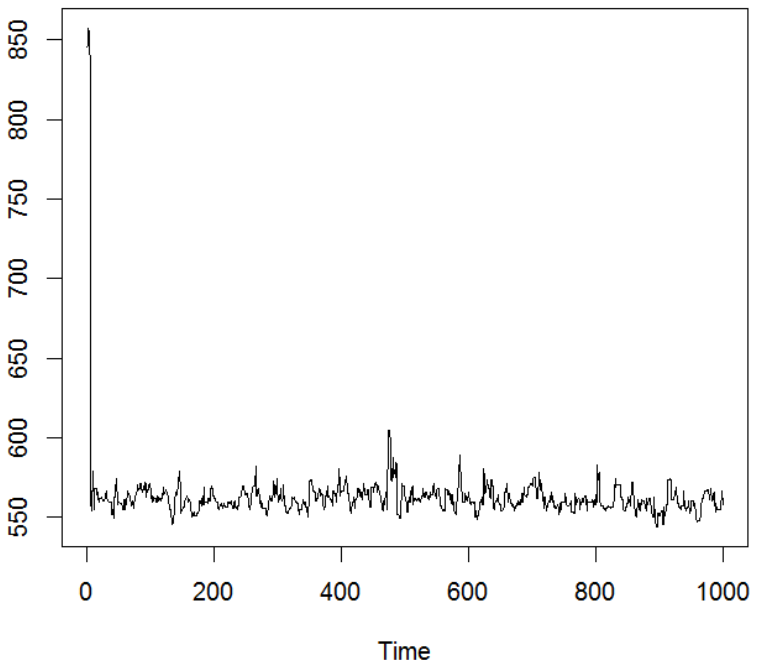
b. Compare the effect of a proposal distribution that is discrete uniform over 2-neighborhoods versus one that is discrete uniform over 3-neighborhoods.

Ex 3.3 100/log(i+1) mi=1

2-nbh

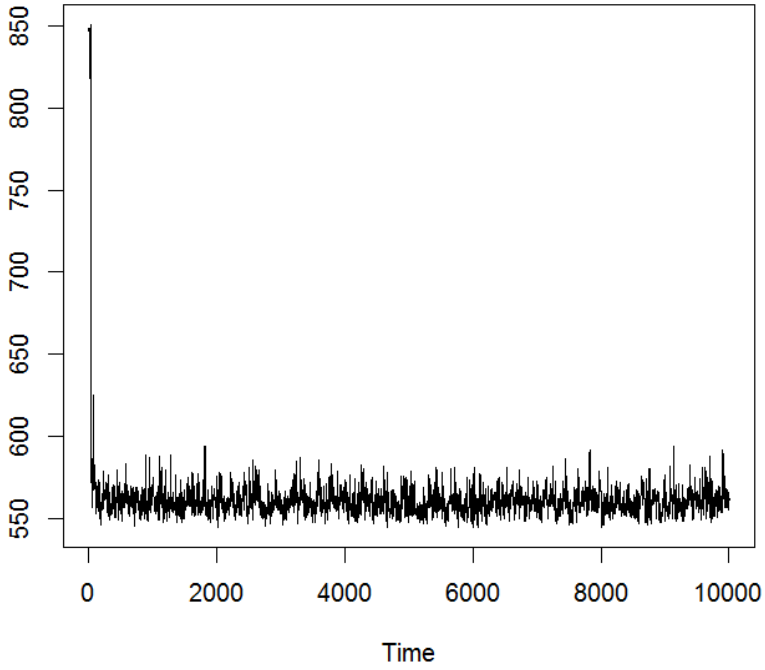


3-nbh

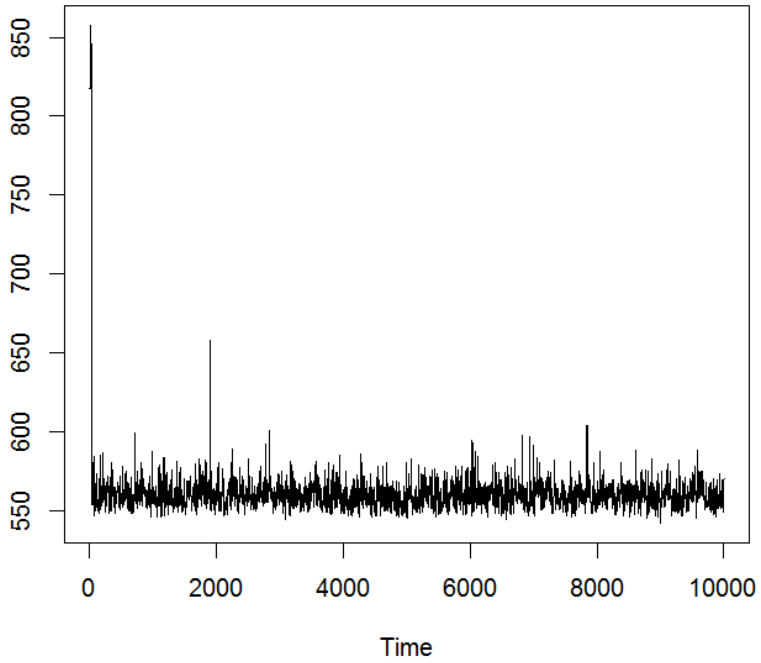


Ex 3.3 100/log(i+1) mi=1

2-nbh

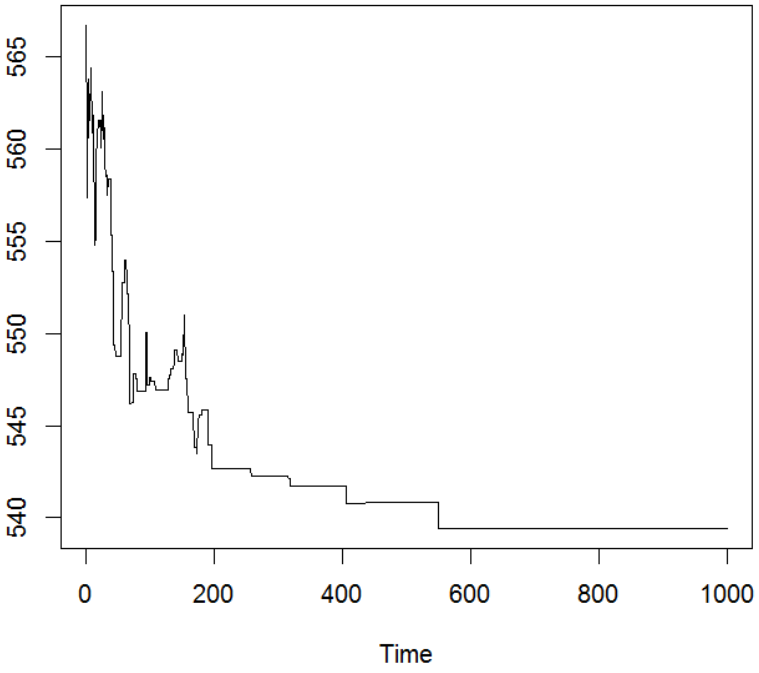


3-nbh

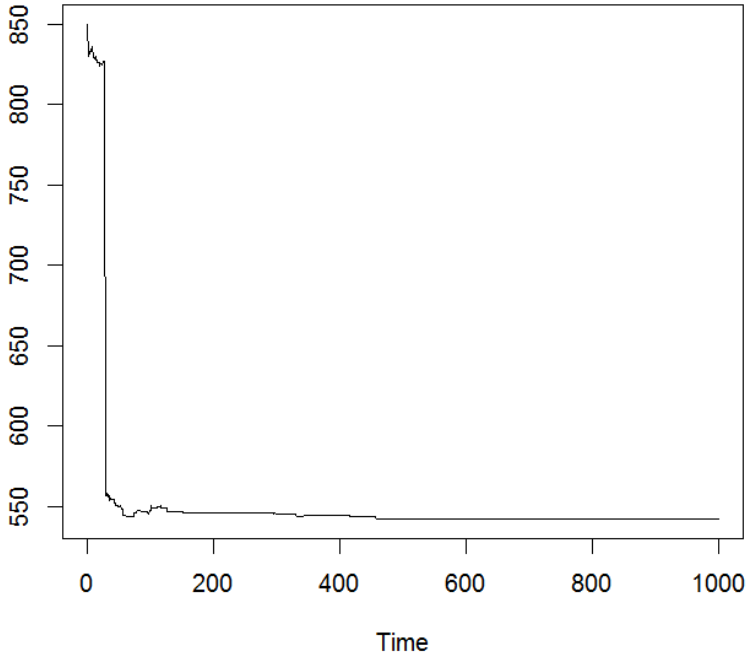


Ex 3.3 100/(i+1) mi=1

2-nbh

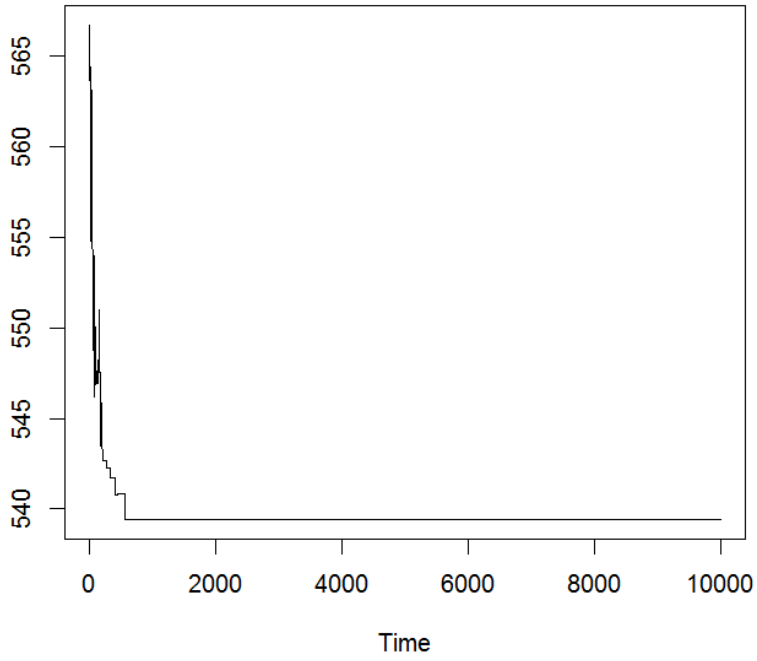


3-nbh

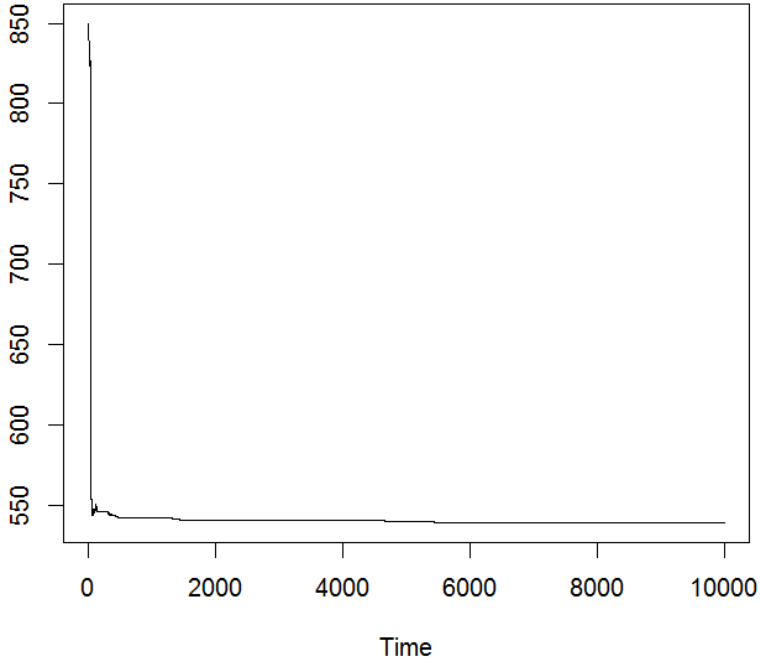


Ex 3.3 100/(i+1); mi=1

2-nbh



3-nbh



Ex 3.3 100/(i+1); mi=100

3-nbh