Метод имитации закаливания (simulated annealing, SANN)

Содержание

- теоретические основы метода
- пример практической реализации в «R»:
  подбор параметров смеси распределений

Теоретические основы метода

Задача:

$$f(\vec{x}) = f(x_1, \dots, x_d) \to \min_{\vec{x}}.$$

Основная идея метода

Метод имитации закаливания (МИЗ) берёт свой название от технического процесса термообработки металла, при котором его кристаллическая решётка нагревается, а затем медленно охлаждается, позволяя атомам занять состояние с минимальной потенциальной энергией

Идея метода состоит в моделировании этого процесса для нахождения минимума целевой функции, являющейся аналогом энергии

Основная идея метода

Пусть $\vec{x}_k$ — набор параметров (состояние системы) на *k*-й итерации метода

Пусть $Z(\vec{x}_k)$ — множество состояний системы, в которые она может перейти из состояния $\vec{x}_k$ за одну итерацию

Для перехода на следующую итерацию случайным образом выбирается новое значение параметров $\vec{z}_k \in Z(\vec{x}_k)$

Вероятность перехода системы в новое состояние:

$$P(\vec{x}_{k+1} := \vec{z}_k) = \begin{cases} \exp\left(\frac{f(\vec{x}_k) - f(\vec{z}_k)}{t_k}\right), & f(\vec{z}_k) > f(\vec{x}_k) \\ 1, & f(\vec{z}_k) < f(\vec{x}_k) \end{cases}$$, где

t_k — «температура» на *k*-й итерации — некоторая последовательность, удовлетворяющая условиям:

$t_k > 0,\ \lim\limits_k t_k = 0$

Алгоритм работы МИЗ

1. Задать начальное значение $\vec{x}_0$, последовательность t_k, предельное значение $t_{\min}$ и количество итераций внутреннего цикла *N*, $k := 0$
2. До тех пор, пока $t_k > t_{\min}$ выполнять
   - для *i* от 1 до N выполнить
     - сгенерировать новые значения параметров $\vec{z}_k \in Z(\vec{x}_k)$
     - с вероятностью $P(\vec{x}_{k+1} := \vec{z}_k)$ принять новые значения параметров, с обратной вероятностью — оставить прежние
   - $k := k + 1$

Внутренний цикл служит для достижения термодинамического равновесия, в котором эмпирические частоты переходов из состояния $\vec{x}_k$ в состояние $\vec{z}_k$ равны вероятности $P(\vec{x}_{k+1} := \vec{z}_k)$

Пример практической реализации в «R»

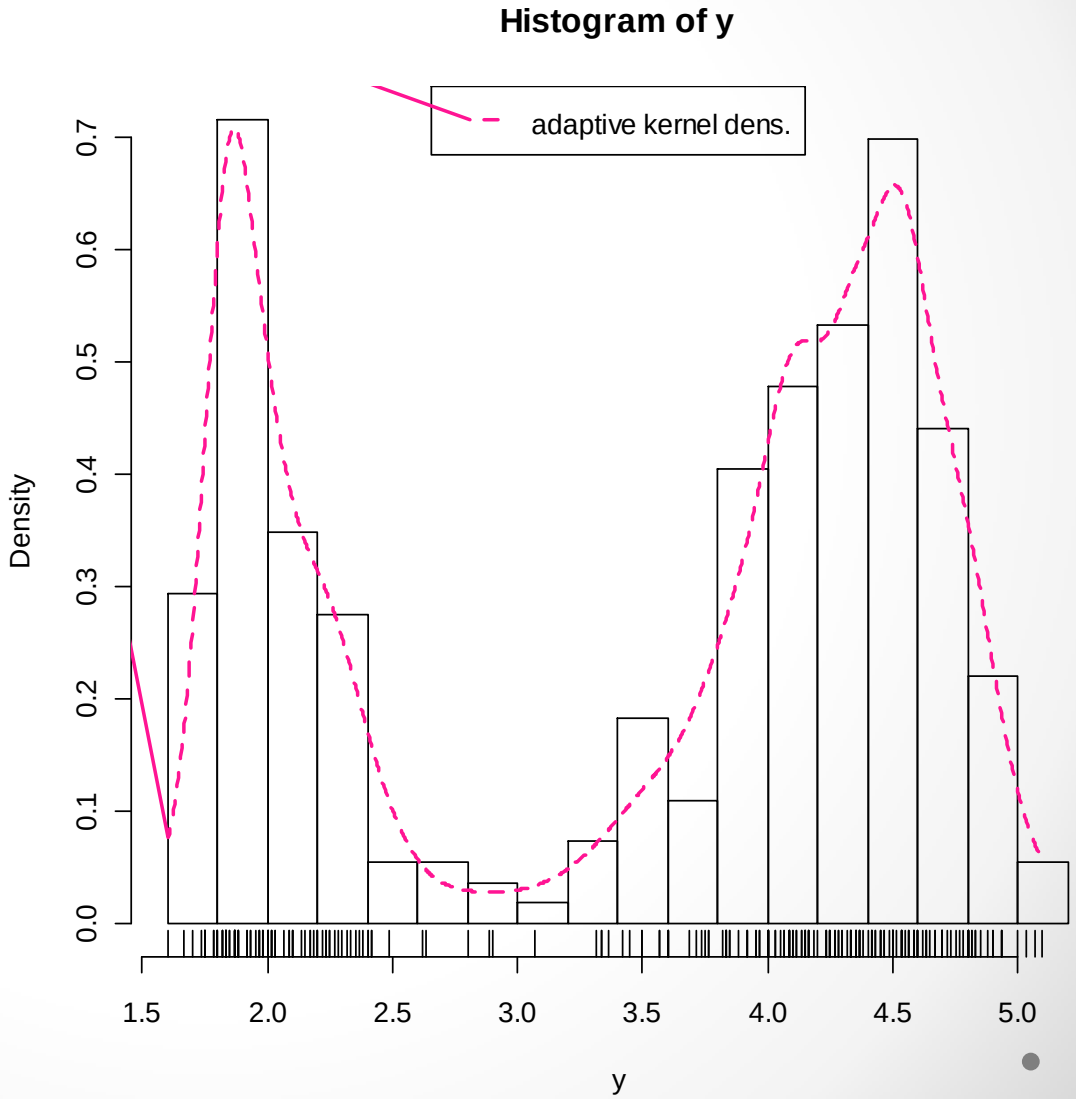
Задача:

подбор оптимальных параметров смеси распределений

Исходные данные

Рассмотрим распределение длительностей извержений гейзера «старый служака»

```
library(np)
y <- faithful$eruptions
```



Распределение Стьюдента с четырьмя параметрами

В распределении ***y*** чётко видны два колоколообразных участка
Рассмотрим задачу моделирования этого распределения с помощью смеси двух четырёхпараметрических распределений Стьюдента

Смесь распределений

Пусть $f_1(x), \dots, f_n(x)$ — функции плотности неких распределений, тогда функция

$f(x) = \sum_{i=1}^{n} w_i f_i(x)$

называется функцией плотности смеси этих распределений, где величины $w_1, \dots, w_n$ — веса распределений в смеси, $\forall i \in \{1; \dots; n\}\ w_i \geq 0,\ \sum_{i=1}^{n} w_i = 1$

Начальные значения параметров

Набор параметров, подлежащих максимизации, состоит из девяти элементов: четыре пары параметров местоположения, масштаба, формы и асимметрии и один параметр веса

```
# начальные значения параметров
n <- 2   # количество распределений
mu0 <- rep(mean(y),times=n); sigma0 <- rep(sd(y),times=n)
nu0 <- rep(10,times=n); xi0 <- rep(1,times=n)
w0 <- rep(1/n,times=n-1)
par <- c(mu0,sigma0,nu0,xi0,w0)

# множество Z(x_k) зададим границами
# изменений значений параметров
mean.cng <- 0.2*mean(y); std.cng <- 0.2*sd(y)
nu.cng <- 0.2; xi.cng <- 0.1; prob.cng <- 0.2
```

Для каждой конкретной задачи рекомендуется задавать свои границы изменения параметров

Внешний цикл алгоритма

Внешний цикл служит для изменения «температуры»: от начального уровня до минимального

```
temp <- 10
min.temp <- 10^-3
N <- 125*n

while(temp > min.temp) {

  ### внутренний цикл ###

  temp <- 0.95*temp
}
```

Внутренний цикл: выбор $\vec{z}_k$

Точку $\vec{z}_k$ мы получаем, случайным образом выбрав один из параметров смеси и изменив его значение в заданных границах

Запишем функцию *change.par*: $\vec{x}_k \to \vec{z}_k$

```
change.par <- function(par,number,eps=10^-3) {
  if (number<=n) {
    # изменение параметра положения
    par[number] <- par[number] + runif(n=1,-mean.cng,mean.cng)
  } else { if (number<=2*n) {
    # изменение параметра масштаба
    par[number] <- par[number] + runif(n=1,-std.cng,std.cng)
    # ограничение σ > 0
    if (par[number]<=0) par[number] <- eps
  } else { if (number<=3*n) {
    # изменение параметра формы
    par[number] <- par[number] + runif(n=1,-nu.cng,nu.cng)
    # ограничение ν > 2
    if (par[number]<=2) par[number] <- 2+eps
```

продолжение на следующем слайде

начало на предыдущем слайде

```
  } else { if (number<=4*n) {
    # изменение параметра асимметрии
    par[number] <- par[number] + runif(n=1,-xi.cng,xi.cng)
    # ограничение ξ > 0
    if (par[number]<=0) par[number] <- eps
  } else {
    # изменение веса одного из распределений
    delta.w <- runif(n=1,-prob.cng,prob.cng)
    par[number] <- par[number] + delta.w
    # ограничение w ≥ 0
    if (par[number]<0) par[number] <- 0
    # пересчёт весов, чтобы ∑w_i = 1
    par[(4*n+1):(5*n-1)] <- par[(4*n+1):(5*n-1)] / (1+delta.w)
  }}}}
  return(par)
}
```

Внутренний цикл: принимать ли $\vec{z}_k$

Запишем логическую функцию, известную также как «критерий Метрополиса», определяющую принимать ли новые значения параметров

```
accept.new.par <- function(cur.energy,new.energy,temp) {
  if (new.energy < cur.energy) return(TRUE)
  if (temp == 0) return(FALSE)
  prob <- exp(-(new.energy-cur.energy)/temp)
  return( runif(n=1,0,1) < prob )
}
```

В качестве «энергии» в нашей задаче выступает отрицательное значение функции правдоподобия

Внутренний цикл: плотность смеси и функция правдоподобия

Плотность смеси есть средневзвешенное её компонент:

```
library(fGarch)
mix.dens <- function(x,par) {
  # количество распределений
  n <- (length(par)+1)/5
  # веса распределений
  wgt <- par[(4*n+1):(5*n-1)]
  wgt[n] <- 1-sum(wgt[1:(n-1)])
  # res – вектор значений плотности
  res <- numeric(length(x))
  for (i in 1:n) res <- res + wgt[i]*dsstd(x,mean=par[i],
                        sd=par[n+i],nu=par[2*n+i],xi=par[3*n+i])
  return(res)
}
```

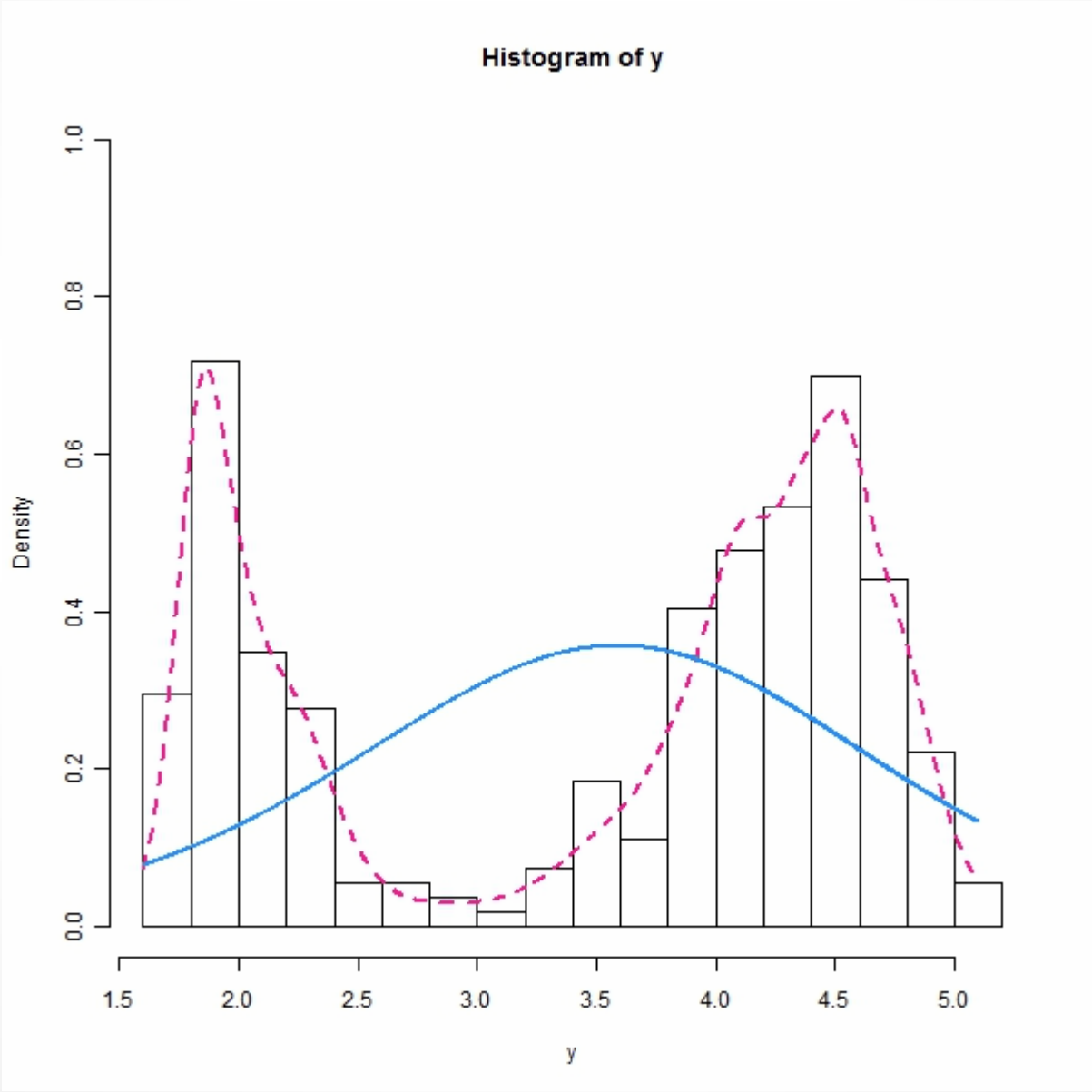
Отрицательная функция правдоподобия:

```
neg.llh <- function(par) -sum(log(mix.dens(y,par)))
```

Домашнее задание

Записать внешний и внутренний циклы алгоритма SANN, используя описанные ранее функции

Подбор смеси: первое прохождение внутреннего цикла



Подбор смеси: прохождение внешнего цикла

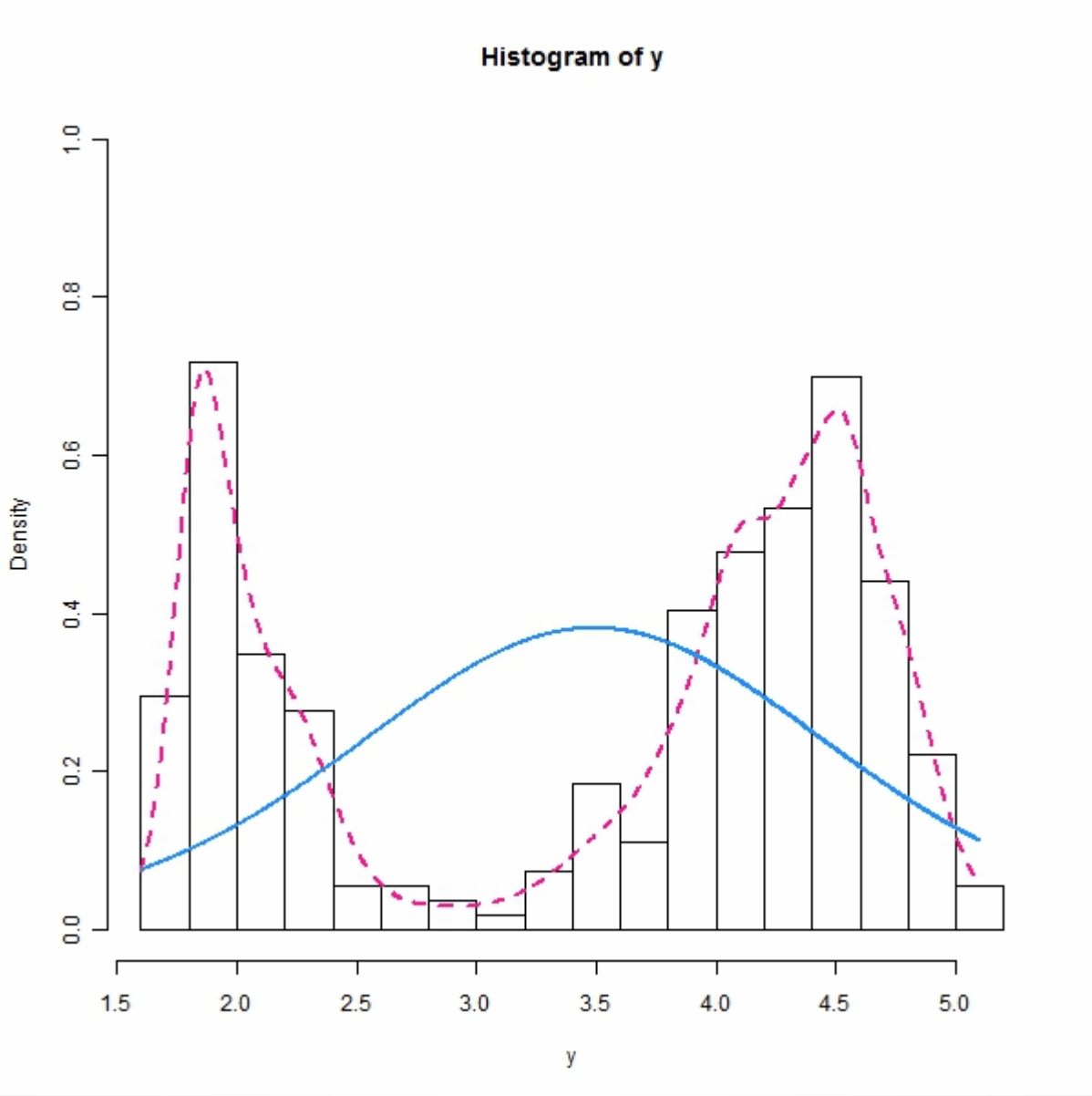


График плотности смеси