Paper Review

LogConcDEAD / fmlogcondens

최 정 인

INDEX

01 LogConcDEAD

02 An Implementation of Shor’s r-Algorithm

03 Source code of **LogConcDEAD**

04 Fast multivariate log-concave density estimation

05 Source code of **fmlogcondens**

01

LogConcDEAD: An R Package for Maximum Likelihood Estimation of a Multivariate Log-Concave Density

Madeleine Cule, Robert Gramacy and Richard Samworth
University of Cambridge

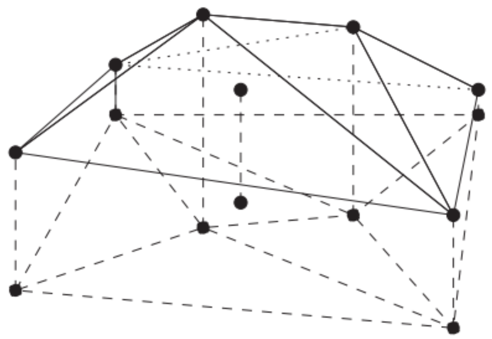
$$\widehat{f}_n = \arg\max_{f \in \mathcal{F}} \sum_{i=1}^{n} w_i \log f(X_i)$$

maximum likelihood estimator $\widehat{f}_n$ of f_0 may be completely specified by its values at the observations $X_1, \ldots, X_n$.

C_n : convex hull of the data.

$$\arg\min_{y \in \mathbb{R}^n} \sigma(y) = -\sum_{i=1}^{n} w_i y_i + \int_{C_n} \exp\{\bar{h}_y(x)\}\, dx$$

$$\bar{h}_y(x) = \inf\{h(x) : h \text{ is concave}, h(X_i) \geq y_i, i = 1, \ldots, n\}.$$



'Tent-like' structure

$$\sigma(y^t) \to \min_{y \in \mathbb{R}^n} \sigma(y)$$

$$y^{(l+1)} = y^{(l)} - h_{l+1} \frac{\partial\sigma(y^{(l)})}{\|\partial\sigma(y^{(l)})\|}$$

At each iteration, the algorithm requires the evaluation $\sigma(y^t)$, and the $\partial\sigma(y^{(l)})$, which determines the direction of the move to the next term y t+1 in the sequence.

Stopping criteria

$$|y_i^{t+1} - y_i^t| \leq \delta |y_i^t| \text{ for } i = 1, \ldots, n$$ ytol

$$|\sigma(y^{t+1}) - \sigma(y^t)| \leq \epsilon |\sigma(y^t)|$$ sigmatol

$$\left| \int_{C_n} \exp\{\bar{h}_{y^t}(x)\}\, dx - 1 \right| \leq \eta$$ integraltol

02

An Implementation of Shor's *r*-Algorithm

FRANZ KAPPEL franz.kappel@kfunigraz.ac.at
ALEXEI V. KUNTSEVICH alex@bedvgm.kfunigraz.ac.at
Institute for Mathematics, University of Graz, Heirichstr., 36, A-8010 Graz, Austria

Received April 7, 1998; Accepted January 6, 1999

2. Shor's r-algorithm

The main idea of the algorithm is to make steps in the direction opposite to a sub-gradient at the current point. However, the steps are to be made in the **transformed space.**

At the initial step, compute a subgradient $g_f(x_0)$ for a given starting point x_0, choose $h_1 > 0$ and find $x_1 = x_0 - h_1 g_f(x_0)$, set $\tilde{g}_1 = g_f(x_0)$ and $B_0 = I$ (the unit $n \times n$ matrix).

2. Shor's r-algorithm

The main idea of the algorithm is to make steps in the direction opposite to a sub-gradient at the current point. However, the steps are to be made in the **transformed space.**

* difference between a subgradient at the current and previous point

1. Calculate $g_f(x_k)$, a subgradient of f at x_k.
2. Calculate $g_k^* = B_k^T g_f(x_k)$, a subgradient of φ_k at the point $y_k = B_k^{-1} x_k$.
3. Calculate $r_k = g_k^* - \tilde{g}_k$, the difference of the two subgradients of φ_k at y_k and $\tilde{y}_k$.
4. Set $\xi_{k+1} = r_k / \|r_k\|$. The normalized vector ξ_{k+1} is the direction of the next space dilation to be performed.
5. Calculate $B_{k+1} = B_k R_\beta(\xi_{k+1})$, where $\beta = 1/\alpha$, $\alpha > 1$ is a fixed constant. The matrix $R_\beta(\xi_{k+1})$ is the inverse of $R_\alpha(\xi_{k+1})$, the matrix of the space dilation in the direction ξ_{k+1} with coefficient α given by

$$R_\alpha(\xi_{k+1})x = x + (\alpha - 1)(x^T \xi_{k+1})\xi_{k+1}, \quad x \in \mathbb{R}^n.$$

6. Calculate $\tilde{g}_{k+1} = B_{k+1}^T g_f(x_k)$, a subgradient of the function $\varphi_{k+1}(y) = f(B_{k+1}y)$ at the point $\tilde{y}_{k+1} = B_{k+1}^{-1} x_k$.
7. Choose a step size h_{k+1}.
8. Set $x_{k+1} = x_k - h_{k+1} B_{k+1} \tilde{g}_{k+1}$.
9. Check the stopping criterion and stop if it is satisfied. Otherwise proceed to the next iteration.

03

Source code of **LogConcDEAD**

```r
out <- .C("logconestw", yvalue = as.double(y[lcdsort]), as.double(x[lcdsort,
]), as.integer(ncol(x)), as.integer(nrow(x)), as.double(w[lcdsort]),
options = as.double(opts), minvalue = double(1), Jtol = as.double(Jtol),
chopts = as.character(chopts), nouter = as.integer(nouter),
PACKAGE = "LogConcDEAD")
```

```c
/* Use the solvoptweights */
*sigmavalue_out=solvoptweights(truepoints,y_in,&sigmaeffw,&subgradeffw,opt_out,&dnull_entry,&null_entry);
renormalise(y_in);
/*That's all!*/
```

```c
double solvoptweights(int n,
               double x[],
               double fun(),
               void grad(),
               double options[],
               double func(),
               void gradc()
              )
{
```

```
/* INITIAL STEPSIZE : */
      d=zero; for (i=0;i<n;i++) { if (d<fabs(x[i])) d=fabs(x[i]); }
      h=h1*sqrt(options[1])*d;           /* smallest possible stepsize */
      /*   if (fabs(options[0])!=one)
           h=h1*max(fabs(options[0]),fabs(h)); *//* user-supplied stepsize */
      /*  else */
        h=h1*max(one/log(ng+1.1),fabs(h));   /* calculated stepsize */
```

$$\hat{h} = \frac{c}{\log_2(\|g_0\|_2 + 1)}$$

```
 /* Gradient in the transformed space (gt) : */
    ngt=zero; ng1=zero; dd=zero;
    for (i=0;i<n;i++)
    {  d=zero; for (j=0;j<n;j++) d+=B[j+i*n]*g[j];
       gt[i]=d;       dd+=d*g1[i];    ngt+=d*d;    ng1+=g1[i]*g1[i];
    }
    ngt=sqrt(ngt); ng1=sqrt(ng1); dd/=ngt*ng1;
    w=wdef;
```

$\tilde{g}_{k+1} = B_{k+1}^T g_f(x_k)$, a subgradient of the function $\varphi_{k+1}(y) = f(B_{k+1}y)$

Fast multivariate log-concave density estimation

Fabian Rathke *, Christoph Schnörr

Image & Pattern Analysis Group (IPA), Heidelberg University, Im Neuenheimer Feld 205, 69120 Heidelberg, Germany

HIGHLIGHTS

- Fast near-optimal solver for multivariate log-concave density estimation.
- Finds sparse parametrization of density function.
- This facilitates significant speed ups (up to 30000x) over state-of-the-art approach.
- Software is readily available as R CRAN package '`fmlogcondens`'.

$$\hat{f}_n = \exp\big(-\hat{\varphi}_n(x)\big)$$

$$C_n = \bigcup_{i=1}^{N_{n,d}} C_{n,i}, \qquad \hat{\varphi}_n(x)\big|_{C_{n,i}} =: \hat{\varphi}_{i,n}(x) = \langle a_i, x\rangle + b_i, \qquad a_i \in \mathbb{R}^d,\ b_i \in \mathbb{R},$$

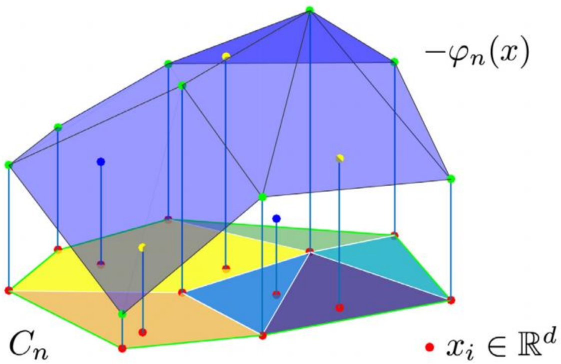
$$\operatorname{epi} \hat{\varphi}_n = \big\{(x, \alpha) \in \mathbb{R}^d \times \mathbb{R} \colon \alpha \geq \hat{\varphi}_n(x)\big\}$$

$$\hat{\varphi}_n = \begin{cases} \max\big\{\hat{\varphi}_{1,n}(x), \ldots, \hat{\varphi}_{N_{n,d},n}(x)\big\}, & x \in C_n, \\ \infty, & x \notin C_n. \end{cases}$$

$$\varphi_n(x_i) \leq y_{\varphi,i}, \qquad i = 1, \ldots, n. \qquad \longleftrightarrow \qquad \bar{h}_y(x) = \inf\{h(x) : h \text{ is concave}, h(X_i) \geq y_i, i = 1, \ldots, n\}.$$

$$\hat{y}_\varphi = \arg\min_{y_\varphi} J(y_\varphi), \qquad J(y_\varphi) = \frac{1}{n}\sum_{i=1}^{n} y_{\varphi,i} + \int_{C_n} \exp\big(-\varphi_n(x)\big)\, \mathrm{d}x$$

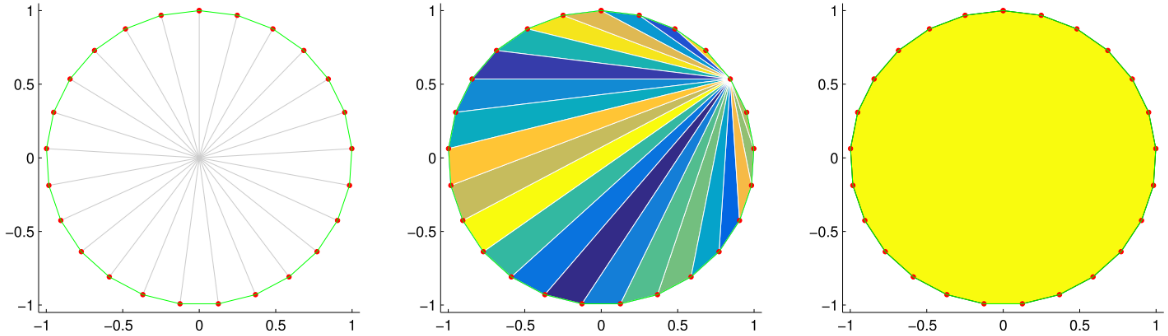
$$\longleftrightarrow \qquad \arg\min_{y \in \mathbb{R}^n} \sigma(y) = -\sum_{i=1}^{n} w_i y_i + \int_{C_n} \exp\{\bar{h}_y(x)\}\, dx$$



$$C_n = \bigcup_{i=1}^{N_{n,d}} C_{n,i},$$

(i) The objective function J is convex but non-smooth. As consequence, the iterative scheme is based on subgradients which are known to converge rather slowly.

(ii) The integral has to be evaluated in every iterative step for each subset Cn. While this can be conveniently done in closed form, it is the increasing number of these subsets for larger dimension d > 2 that slows down the algorithm



(a) n-gon with $n = 25$

(b) Cule et al. (2010), $l(\theta) = -29.4109,\ N_{n,d} = 23$

(c) Our estimate, $l(\theta) = -29.4109,\ N_{n,d} = 1$

(1) We consider the smooth approximation of the non-smooth max-operation. smoothness of the approximation can be controlled by a single parameter γ

(2) we apply a threshold criterion in order to drop ‘inactive’ hyperplanes, since the optimal estimate φn can be expected to be defined by a small subset of them. This measure speeds up the computation without essentially compromising the accuracy of the resulting density estimator.

(3) unlike the approach of Cule et al. (2010), we do not restrict polyhedral subsets Cn,i to simplices.

$$L(\theta) := \frac{1}{n}\sum_{i=1}^{n} \varphi_n(x_i) + \int_{C_n} \exp(-\varphi_n(x))\, dx, \qquad \longleftrightarrow \qquad \arg\min_{y \in \mathbb{R}^n} \sigma(y) = -\sum_{i=1}^{n} w_i y_i + \int_{C_n} \exp\{\bar{h}_y(x)\}\, dx$$

$\hat{\varphi}_n = \varphi_n|_{\theta=\hat{\theta}}$: $\hat{\theta}$ locally minimizes $L(\theta)$.

Our next step is to smoothly approximate the representation of φ_n

$$\mathrm{logexp}_\gamma : \mathbb{R}^d \to \mathbb{R}, \qquad x \mapsto \mathrm{logexp}_\gamma(x) := \gamma\, \mathrm{logexp}\left(\frac{x}{\gamma}\right) = \gamma \log\left(\sum_{i=1}^{d} \exp\left(\frac{x_i}{\gamma}\right)\right)$$

$$\varphi_{n,\gamma}(x) := \begin{cases} \mathrm{logexp}_\gamma\left(\varphi_{1,n}(x), \ldots, \varphi_{N_{n,d},n}(x)\right), & x \in C_n, \\ \infty, & x \notin C_n, \end{cases}$$

$$L_\gamma(\theta) := \frac{1}{n}\sum_{i=1}^{n} \varphi_{n,\gamma}(x_i) + \int_{C_n} \exp(-\varphi_{n,\gamma}(x))\, dx \qquad L_\gamma(\theta) \to L(\theta) \quad \text{for} \quad \gamma \to 0$$

We apply an established, memory-efficient quasi-Newton method known as **L-BFGS**

$$\theta^{(k+1)} = \theta^{(k)} + \lambda_k p^{(k)}, \qquad p^{(k)} = -H^{(k)} \nabla L_\gamma(\theta^{(k)})$$

$$H^{(k+1)} = (V^{(k)})^T H^{(k)} V^{(k)} + \rho^{(k)} s^{(k)} (s^{(k)})^T,$$

$$\rho^{(k)} = \frac{1}{(y^{(k)})^T s^{(k)}}, \qquad V^{(k)} = I - \rho^{(k)} y^{(k)} (s^{(k)})^T,$$

$$s^{(k)} = \theta^{(k+1)} - \theta^{(k)}, \qquad y^{(k)} = \nabla L_\gamma(\theta^{(k+1)}) - \nabla L_\gamma(\theta^{(k)})$$

Algorithm 1: Fast Log-Concave Density Estimation

Input: X, parameters: $\gamma = 10^{-3}$, $\vartheta = 10^{-3}$, $\epsilon = 10^{-3}$, $\delta = 10^{-7}$
Output: Log-concave density estimate $\hat{f}_n$ parametrized by θ (2.1).
Find initial $\theta^{(0)}$ (Section 2.4);
for $k = 1, 2, \ldots$ **do**
  Delete inactive hyperplanes from $\theta^{(k)}$ based on criterion (2.18);
  Compute the gradient $\nabla L_\gamma(\theta^{(k)})$ of the objective (2.7) using numerical integration;
  Find descent direction $p^{(k)}$ from the previous m gradients vectors and step size λ_k (2.11) and update $\theta^{(k+1)}$;
  if *the termination criterion* (2.19) *holds,* **then**
    Denote final parameter vector by $\theta^{(\mathrm{final})}$;
    Quit for-loop;
  end
end
Switch from $\hat{\varphi}_{n,\gamma}$ to $\hat{\varphi}_n$ and perform exact normalization: $\theta^{(\mathrm{final})} \to \hat{\theta}$ (Section 2.7);
return $\hat{\theta}$ (2.22)

05

Source code of **fmlogcondens**