Maximum likelihood using multidimensional hyperplane sections

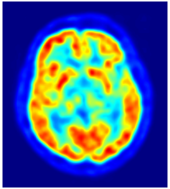
Kacper Topolnicki
kacper.topolnicki@uj.edu.pl

Institute of Physics, Jagiellonian University

11 IX 2026

in collaboration with: Roman Skibiński, Sushil Sharma, Yuriy Volkotrub, Manish Das

- for maximizing likelihood, Bayesian inference
- ideally used for "highly concentrated" distributions
- partially inspired by adaptive histogram by Piotr Libucha
- use "differentiable programming" (Tuesday's talk by Pietro Vischia and Wednesday's talk by Aurélien Coussat)
- applications currently used for (versitile) toy problems



- positron emission tomography:
  - tracer element around tumor emitts e^+, e^- pair
  - this pair annihilates and γ's are observed
- given estimated annihilation points from TOF infer the distribution of the tracer

ELSEVIER

Computer Physics Communications

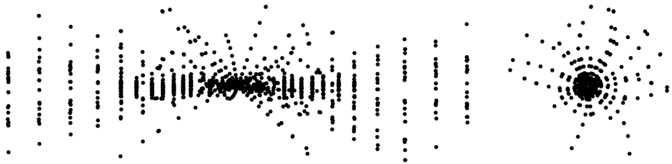
Volume 319, February 2026, 109913

Computational Physics

Exploring maximum likelihood and Bayesian approaches for two-dimensional image restoration: A machine learning perspective

K. Topolnicki, S. Sharma, Yu. Volkotrub, **M. Das**

- example uses a (very) idealized version of 2D tomography



- particles in detector submerged in magnetic field detected as points
- infer the distribution of helix parameters
  - in particular, charged particles that originate away from the beam crossing

Journal of Instrumentation

PAPER

Approximate method for helical particle trajectory reconstruction in high energy physics experiments

K. Topolnicki and T. Bold

Published 26 August 2022 • © 2022 IOP Publishing Ltd and Sissa Medialab

Journal of Instrumentation, Volume 17, August 2022

- alternative to Hough transform

K. Topolnicki, T. Bold

Numerical Complexity of Helix Unraveling Algorithm for Charged Particle Tracking

Acta Phys. Pol. B **55**, 6-A1 (2024) • published online: 2024-05-10

- complexity too big to be practical :-(

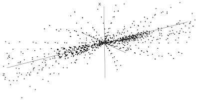
- focus on the helix problem
- very similar description of tomography

let's start with Bayes rule:

$$P(H|DX) = P(H|X)\frac{P(D|HX)}{P(D|X)}$$

P - probability, H - hypothesis, D - data, X - assumptions

- X : eg. helixes are independently drawn from from the normalized probability distribution f, points are independently drawn from a helix
- H: distribution f of helix parameters $(x_c, y_c, r_c, \ldots)$
- D: $\mathbb{D} = \{(x_i, y_i, z_i) : i = 1 \ldots N\}$

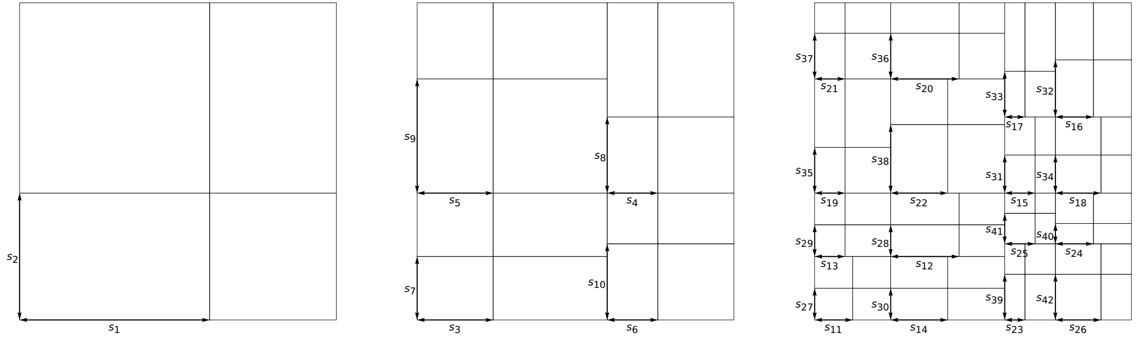


- difficult to obtain distribution of hypothesis
  - ... hamiltonian MC to reconstruct distribution of distributions ...
- forget about the distribution (of distributions): instead find H that maximizes $P(D|HX)$

for simplicity consider the two dimensional case $f = f(x_c, y_c)$:

- assume some reasonable limits for x_c , y_c
- set $f = 0$ outside of these limits
- partition the $x_c - y_c$ domain into non-overlaping regions:

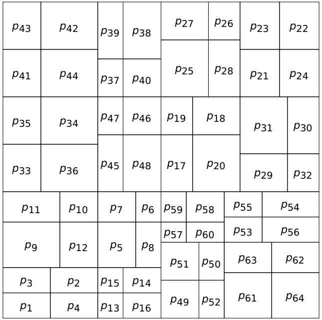
geometric paramters:



$s_1, s_2,$ $\quad$ $s_3, \ldots, s_{10},$ $\quad$ $s_{11}, \ldots, s_{42} : 0 < s_i < 1$

- assume that the probability density is uniform in each region
- overall probabilities for each region consitute additional parameters
  - in principle we could set this to $\frac{1}{64}$ ... but this causes issues

probabilities in regions::



$$p_1, \ldots, p_{64} : p_i > 0, \sum_{i=1}^{64} p_i = 1$$

for unconstrained optimization use $\bar{s}_i$, $\bar{p}_i$ instead of s_i, p_i:

- $s_i = m + \frac{1-2.0m}{2}(1 + \sin(\bar{s}_i))$
  - $\bar{s}_i$ can be any value
  - m is the margin $m \leq s_i \leq 1.0 - m$, typically $m = 0.3$
  - the grid will also be used for numerical integration, m puts lower limit on the resolution
- $p_i = \frac{\bar{p}_i^2}{\sum_i \bar{p}_i^2}$
  - $\bar{p}_i$ can be any value

for simplicity:

$$f_1, f_2, \ldots, f_M := \bar{s}_1, \bar{s}_2, \ldots, \bar{p}_1, \bar{p}_2, \ldots$$

- multidimensional versions available [1]
- under review
- conceptually much simpler then previous two- and multi- dimensional versions
    - first multidimensional version described in Kacper Czykierda's master thesis
    - new version allows tinkering with the partitionig scheme
    - alternative available [2]

[1] https://github.com/kacpertopolnicki/hsec
[2] https://github.com/kacpertopolnicki/hsec_alternative

let's go back to Bayes rule:

$$P(H|DX) = P(H|X)\frac{P(D|HX)}{P(D|X)}$$

$$\downarrow$$

$$p((f_1, f_2, \ldots)|\{(x_i, y_i, z_i) : i = 1 \ldots N\}X)df_1 df_2 \ldots =$$
$$p((f_1, f_2, \ldots)|X)df_1 df_2 \ldots$$
$$\frac{p(\{(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots\}|(f_i : i = 1 \ldots M)X)dx_1 dy_1 dz_1 dx_2 dy_2 dz_2 \ldots}{p(\{(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots\}|X)dx_1 dy_1 dz_1 dx_2 dy_2 dz_2 \ldots}$$

- focus on maximizing $p(\{(x_i, y_i, z_i) : i = 1 \dots N\} | (f_i : i = 1 \dots M)X)$ with respect to f_i
- $\frac{\partial p(\{(x_i, y_i, z_i) : i = 1 \dots N\} | (f_i : i = 1 \dots M)X)}{\partial f_i}$ calculated in *pytorch*
- optimization using the *Adam* optimizer
- very direct approach using "differentiable programming", many algorithms just show that each iteration increases the likelihood

- focus on the $x - y$ histogram
- limit the number of helix parameters: (x_c, y_c) is the position of the helix axis
- in general:

$$p((x,y)|f\,X) = \int dx_c dy_c f(x_c, y_c) p((x,y)|(x_c, y_c))$$

$$p(\{(x_i, y_i) : i = 1 \dots N\}|f\,X) = \Pi_{i=1}^{N} p((x_i, y_i)|f\,X)$$

  - log probabilities used to turn Π into Σ for numerical calculations

- in practice (uniform distribution in each region):

$$p((x,y)|f\,X) = \sum_i f(i) \int_{x_{min}(i)}^{x_{max}(i)} dx_c \int_{y_{min}(i)}^{y_{max}(i)} dy_c p((x,y)|(x_c,y_c))$$

  where $x_{min}(i)$, $x_{max}(i)$, $y_{min}(i)$, $y_{max}(i)$ are the ith region boundries and $f(i)$ is the probability density in region i

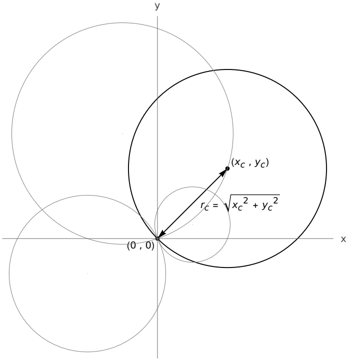
  - all calculated from $f_1, f_2, \ldots, f_M$

- the integral

$$\int_{x_{min}(i)}^{x_{max}(i)} dx_c \int_{y_{min}(i)}^{y_{max}(i)} dy_c p((x,y)|(x_c,y_c)) \approx$$
$$\text{(average value on corners)}(x_{max}(i) - x_{min}(i))(y_{max}(i) - y_{min}(i))$$

  - margins are important

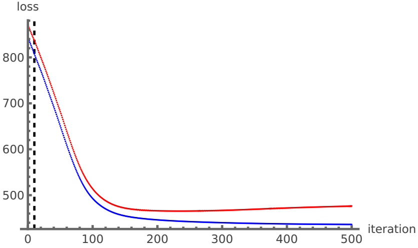
- $p((x,y)|(x_c,y_c))$ is is the most important input: considering two cases assume particle tracks start at $(0, 0, 0)$:

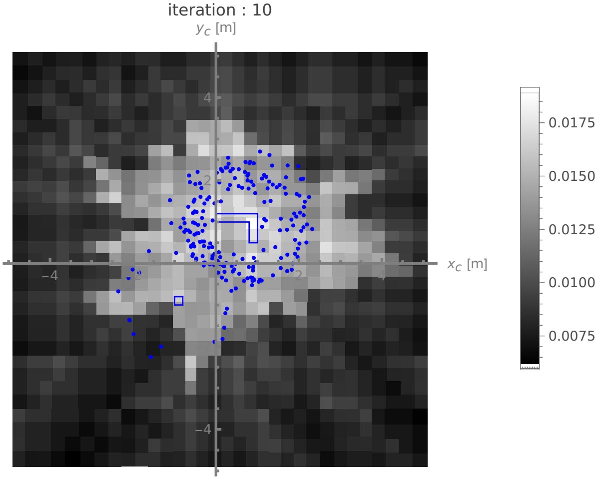


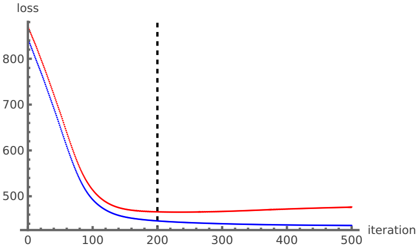
- only look at $x - y$ projection
- γ determines the spread around the helix

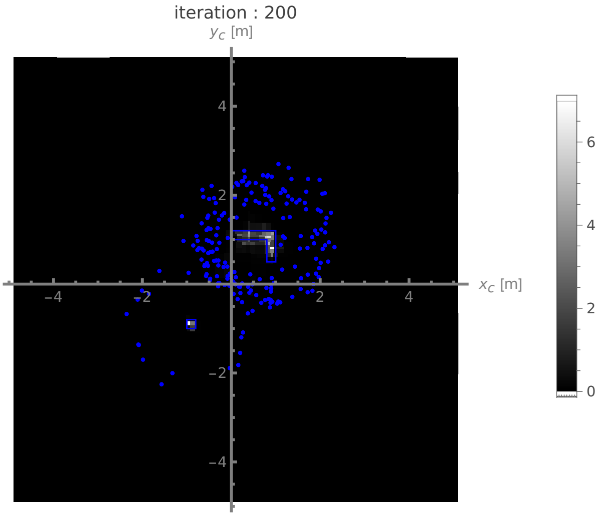
$$p((x,y)|(x_c,y_c)) = \frac{\exp\left(-\frac{1}{2\gamma^2}\left(\sqrt{(x-x_c)^2+(y-y_c)^2}-\sqrt{x_c^2+y_c^2}\right)^2\right)}{\pi\sqrt{2\pi}|\gamma|\sqrt{x_c^2+y_c^2}\left(\operatorname{erf}\left(\frac{\sqrt{x_c^2+y_c^2}}{\sqrt{2}|\gamma|}\right)+1\right)+2\pi\gamma^2 e^{-\frac{x_c^2+y_c^2}{2\gamma^2}}}$$

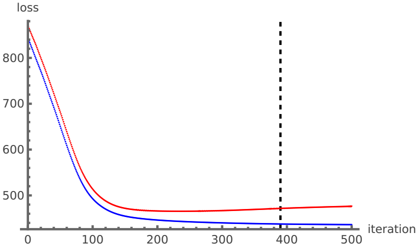
- in the following examples $\gamma = 0.1$, $m = 0.3$, the learning rate for the *Adam* optimizer 0.01, and 200 registered hits (same number of validation points)

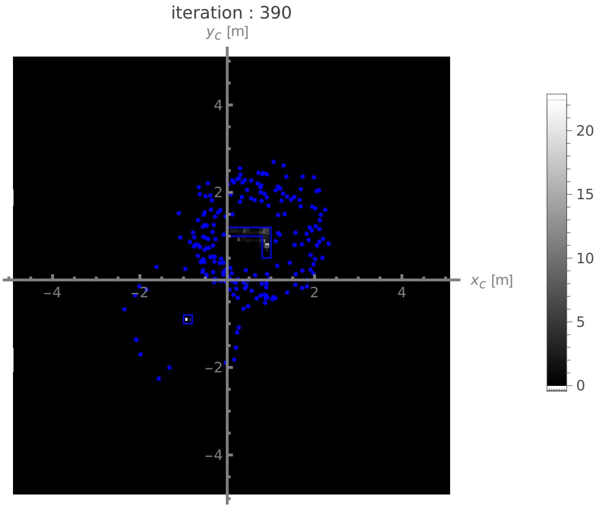

loss
800
700
600
500
0
100
200
300
400
500
iteration


iteration : 10
y_c [m]
4
2
−4
−2
2
4
−4
x_c [m]
0.0175
0.0150
0.0125
0.0100
0.0075

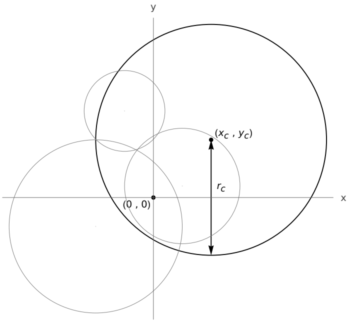

loss
800
700
600
500
0
100
200
300
400
500
iteration


iteration : 200
y_c [m]
4
2
−2
−4
−4
−2
2
4
x_c [m]
6
4
2
0


loss
800
700
600
500
0
100
200
300
400
500
iteration


iteration : 390
y_c [m]
x_c [m]
4
2
−2
−4
−4
−2
2
4
20
15
10
5
0

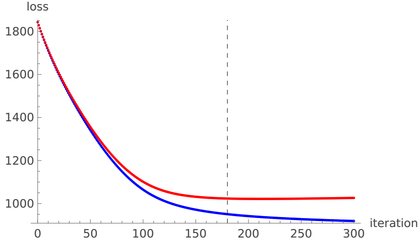
don't assume particle tracks start at $(0, 0, 0)$:

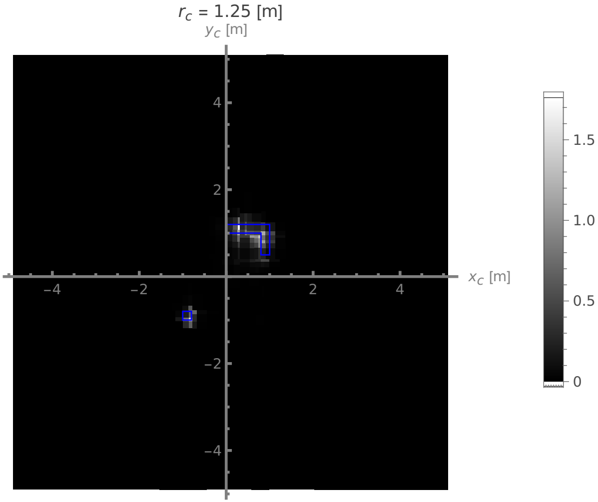


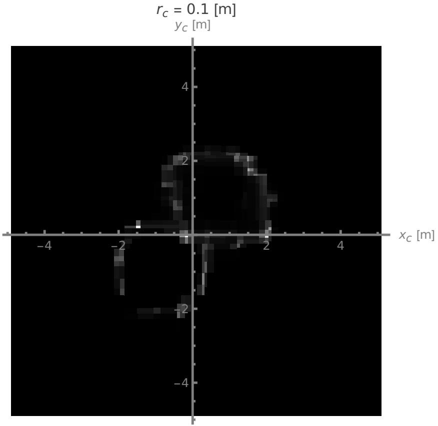
- γ determines the spread around the helix
- additional parameter r_c

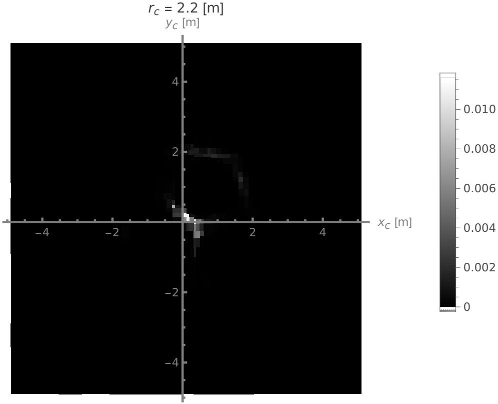
$$p((x,y)|(x_c,y_c,r_c)) = \frac{e^{-\frac{\left(\sqrt{(x-x_c)^2+(y-y_c)^2}-r_c\right)^2}{2\gamma^2}}}{\pi\sqrt{2\pi}\gamma r_c\left(q\left(-\frac{1}{2},0,\frac{r_c^2}{2\gamma^2}\right)+1\right)}$$

- implementing the normalizing factor in *pytorch* was tricky, had help from google llm
- potentially more interesting: look for charged particle tracks that originate away from the collision point
- in the following examples $\gamma = 0.1$, $m = 0.3$, the learning rate for the *Adam* optimizer 0.01, 400 registered hits (same number of validation points), $1 \leq r_c \leq 1.5$


loss
1800
1600
1400
1200
1000
0
50
100
150
200
250
300
iteration


r_c = 1.25 [m]
y_c [m]
4
2
−2
−4
x_c [m]
−4
−2
2
4
1.5
1.0
0.5
0


r_c = 0.1 [m]
y_c [m]
x_c [m]
4
2
−2
−4
−4
−2
2
4
8
6
4
2
0


r_c = 2.2 [m]
y_c [m]
x_c [m]
4
2
−2
−4
−4
−2
2
4
0.010
0.008
0.006
0.004
0.002
0

- finally we have a multidimensoinal version of the algorithm
- easy to reason about and tweak
  - experiment with alternative partitioning schemes
- sample from hypothesis distribution: requires an approach more sophisticated then Metropolis Hastings
- allow for batched output
- apply to real problems and data