BAYESIAN ON-THE-FLY EXPERIMENT DESIGN

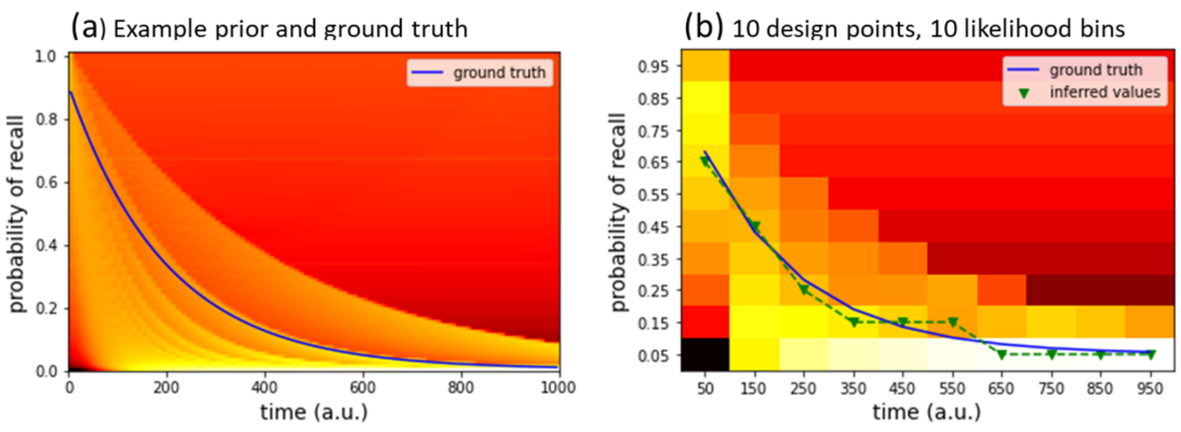
Rocco Di Tella[1], Andrés Rieznik[2][3], Ariel Futoransky[3]

[1] UBA, FCEN ; [2] Laboratorio de Neurociencias I UTDT ; [3]BitTrap Research Team

Bayes Plurinacional

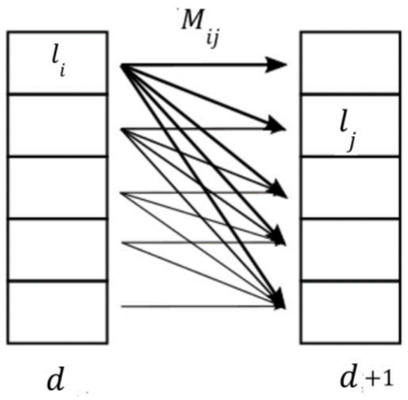
DISCRETIZACIÓN DEL ESPACIO

En este trabajo presentamos un modelo no paramétrico con actualización bayesiana en tiempo real. La principal aproximación que hacemos es tomar una grilla de i x k y considerar sólo los modelos que pueden ser representados como funciones crecientes (o decrecientes) que pasen por los centros de estos bins.
Esta aproximación se puede hacer tan fina como uno quiera, o como permitan los tiempos de cómputo de la aplicación.



UPDATE DEL PRIOR

La principal bondad de usar esta aproximación discreta no paramétrica, es poder hacer las cuentas muy rápido. Tenemos un algoritmo que permite hacer un update en una sóla pasada por la grilla, en lugar de tener que calcular el posterior de cada uno de los (combinatoriamente muchos) modelos posibles.



$$l_i^*(d') \equiv l_i(d'|y) = \frac{v_i\, l_i(d')}{p(y|d')}$$

$$M_{ij}^*(d') = \frac{l_i^*(d')}{l_i(d')}\, M_{ij}(d')$$

$$l_j^*(d'+1) = \sum_{i=1}^{K} M_{ij}^*(d')$$

ELECCIÓN DEL DISEÑO ÓPTIMO

Esta cuenta del paso anterior se puede hacer muy rápido, con lo cual es factible hacerla para todas las mediciones posibles, y así calcular cuál es el punto de máxima esperanza para la información ganada. Esperanza que se computa bajo la probabilidad marginal de una medición en la ubicación d óptima.

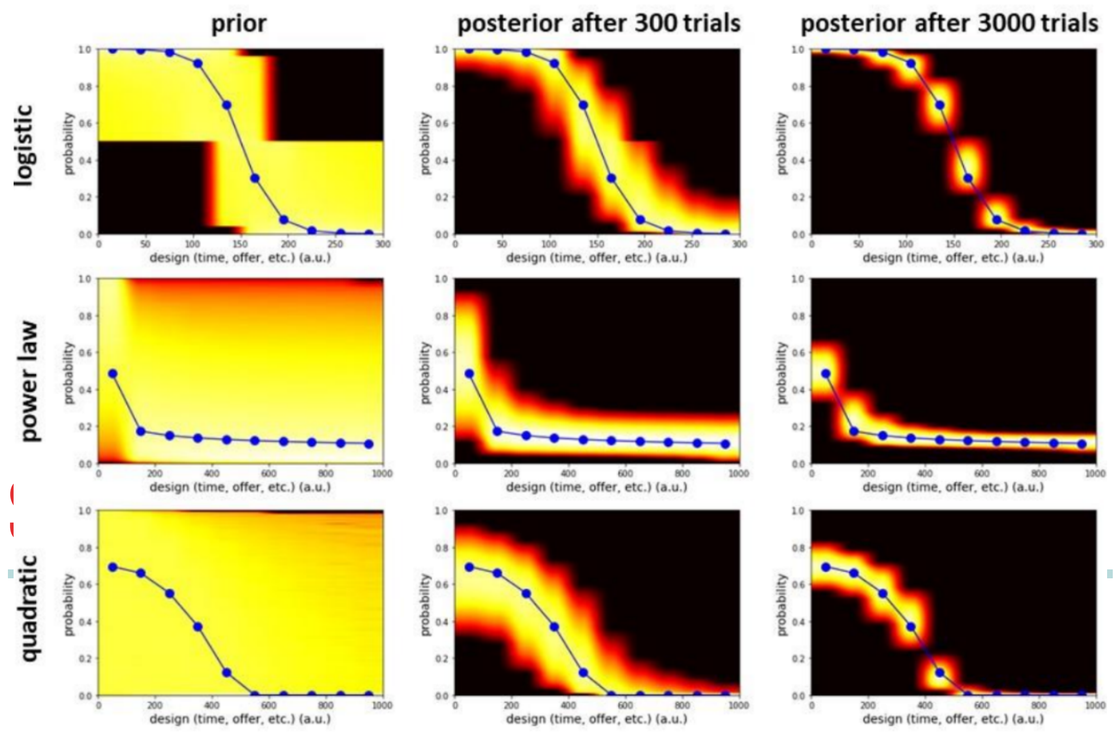
$$I(d) = \sum_{y} p(y|d) \sum_{m=1}^{N} p(m|y,d)\, log_2\left(\frac{p(m|y,d)}{p_m}\right)$$

Pseudo code showing how to implement the open-source python package in an adaptive design experiment:

- from ADO_Experiment import * # import our package
  import numpy as np
- *n* = 10 # number of design points
  k = 100 # number of likelihood bins
  exp = Experiment() # create the object exp from the class Experiment
  exp.generate(*n*,*k*) # set the number of design and likelihood points and, among other data, generate a prior matrix considering the set of all monotonically decreasing functions enabled by the discretizations and considering all of them equally probable.
- Load *p*.npy # if you wish, you can load your own p matrix representing your priors
  exp.set_prior(*p*) # set the new prior
- For i from 1 to number of trials,
  d' = exp.ADOchoose() # select the design that maximizes the expected information gain.
  # Perform a measure using the design *d'* and inform its result:
  exp.update(*d'*,result)
  # The posterior can be accessed using *exp.p*

BENCHMARKS

Observamos que bajo distintas suposiciones, nuestro método da un mejor uso de la información donde podemos hacer inferencias con ~250 datos de la misma precisión que ~450 datos en un régimen uniformemente aleatorio.



DÓNDE SE PAGA LA VELOCIDAD

En el paper, tenemos una demostración analítica de qué suposiciones tenemos que tomar para que nuestro algoritmo sea full-bayesiano y no una aproximación heurística.