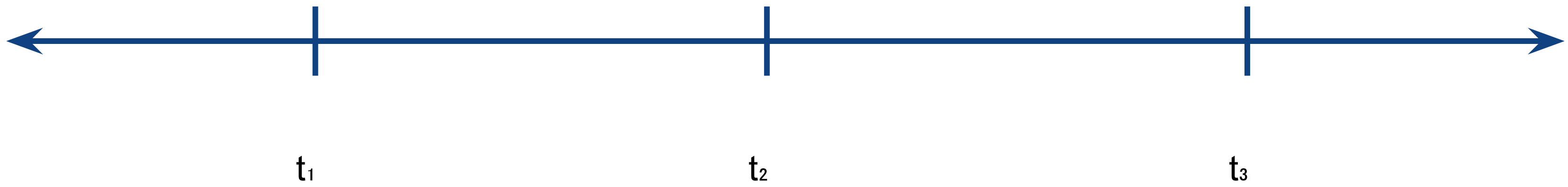
# Active Learning for Excursion Set Estimation

Kyle Cranmer, Lukas Heinrich, Gilles Louppe

# Common problem in Science: Finding Excursion Sets of Functions




$$\mathcal{L}(t_j) = \{x | t_j < f(x) \leq t_{j+1}\}$$

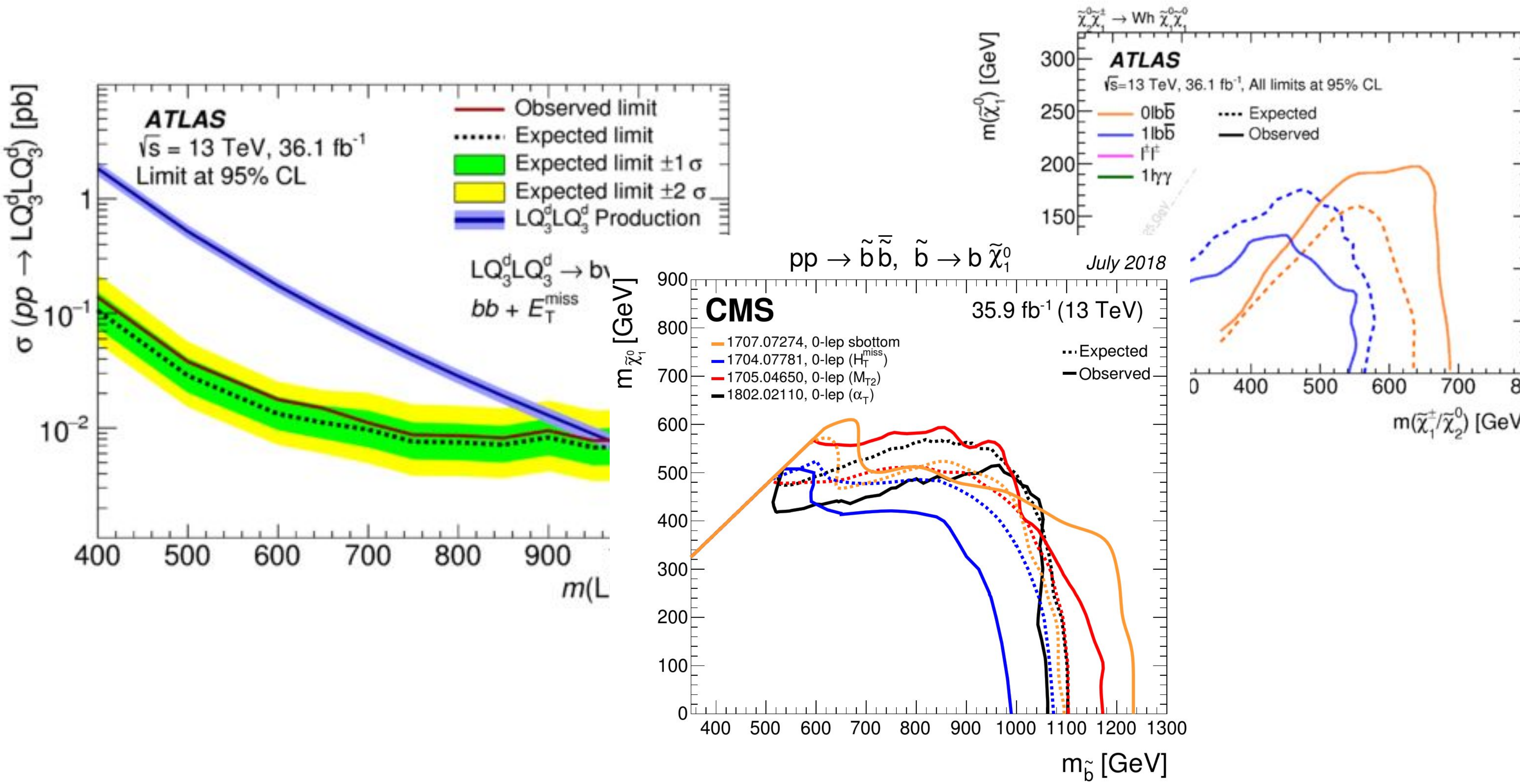
i.e. the sets $L_i$ of points for which the function values is within the interval $[t_i, t_{i+1}]$

Equivalently: the iso-hypersurfaces $f(x)=t_i$ with of multivariate functions $f: R^n \rightarrow R$,

Examples:
1D: intersections
2D: iso-contours:
3D: iso-surfaces,
...

# Example in HEP: Interval Estimation during Inference

## e.g. likelihood contours, exclusion contours, etc..



## Core Problem:

**The functions are often very expensive to evaluate.**

E.g. in case of BSM searches for a single p-value of a given BSM theory, one needs full (and expensive) chain of

Evgen > Simulation > Data Reduction > Event Selection > Inference

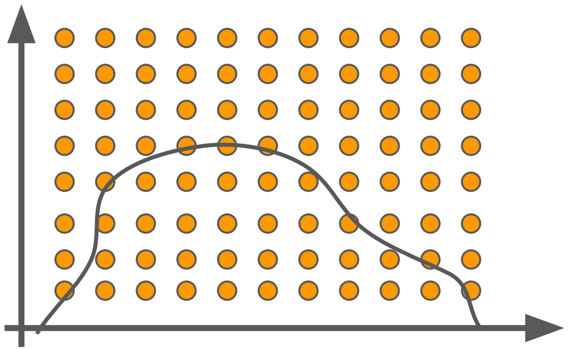
To get a $p(x_1,x_2,\ldots) = 0.05$ contour this function might have to be evaluated many times.

**Key Question:**

**What is a computationally efficicient strategy to find a levelset estimates of expensive black-box functions?**

# Current Approaches

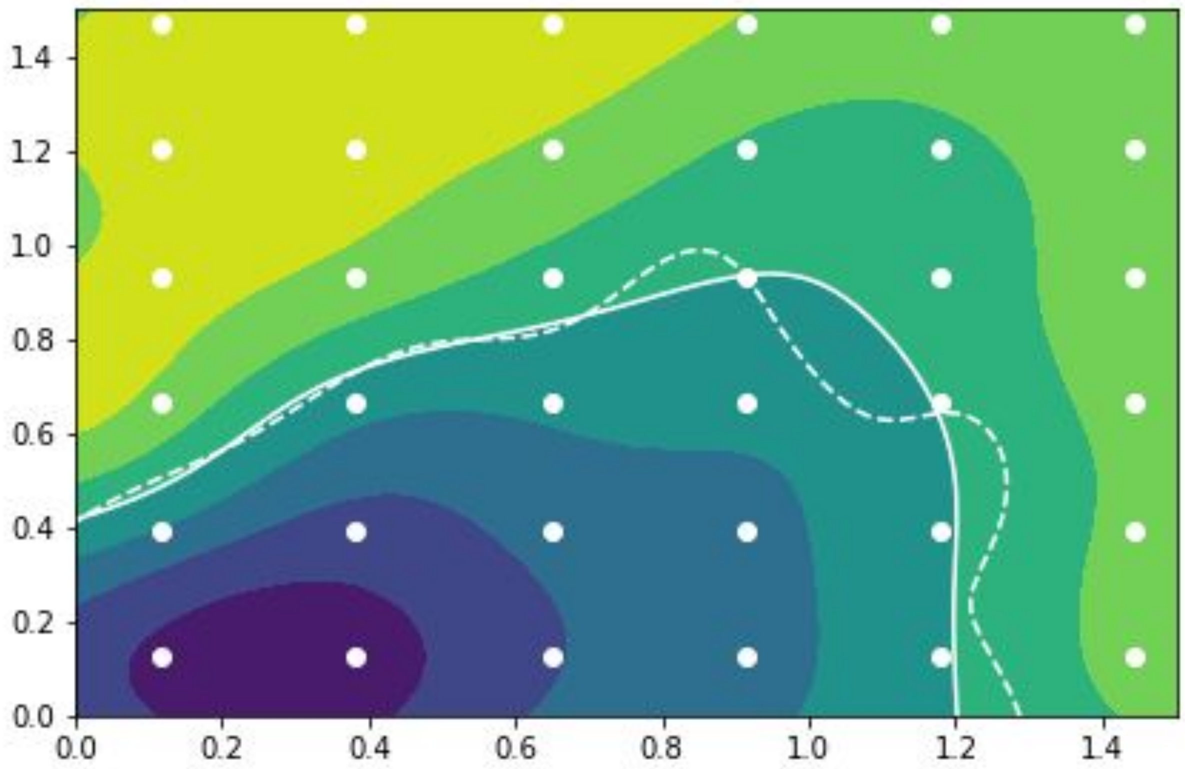
1. Regular / Cartesian Grids



**Evaluate Function on a dense grid O(100) points in 2D. Using a interpolation algorithm find iso-contour.**

**Disadvantages:**

- curse of dimensionality limits parameter space
- arbitrariness of choosing the grid (e.g. placement)



1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

# Current Approaches

## 2. Random Samplings

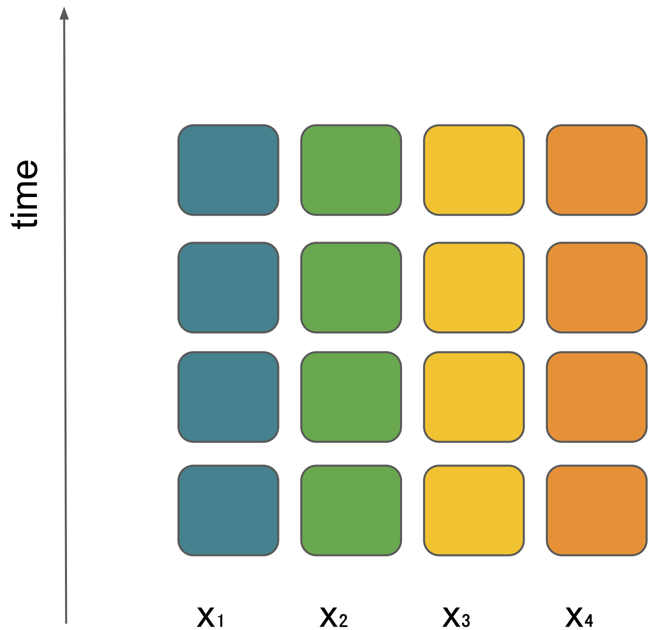
**Various schemes: Uniform Samplings, Poisson Disc, Latin Hypercube, known priors...**

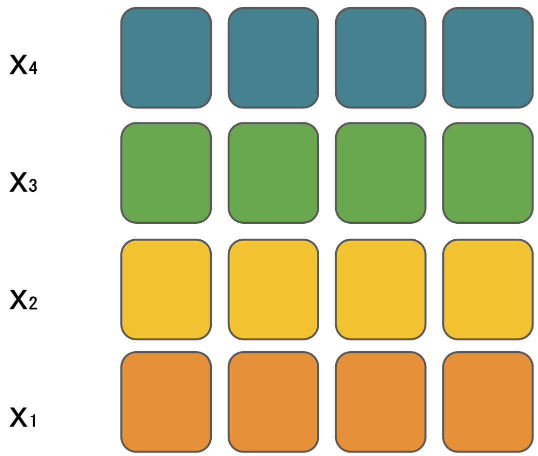
**Works in higher dimensions.**

**But Sampling density fixed and not informed by the function f(x)**

# Redeeming quality of many functions we're interested in:

## Embarrassingly parallel computation.

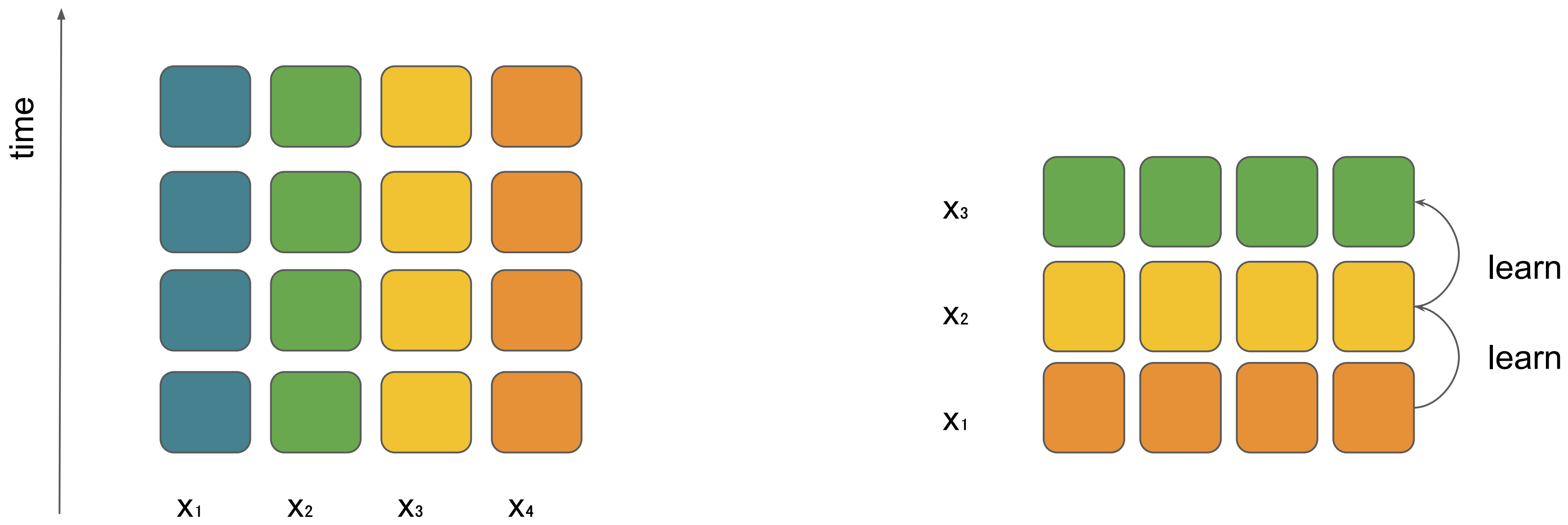


## Redeeming quality of many functions we're interested in:

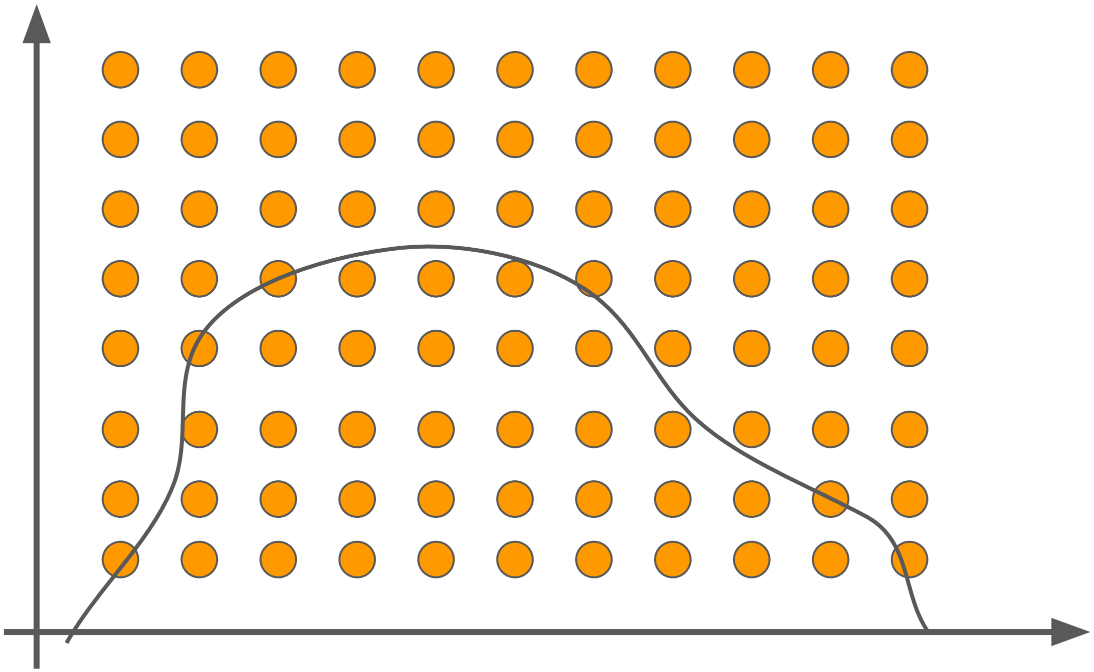
## Embarrassingly parallel computation.

**Idea:** by turning to an iterative approach, use information about f(x) from evaluations $f(x_i)$ to sample parameter space more efficiently.



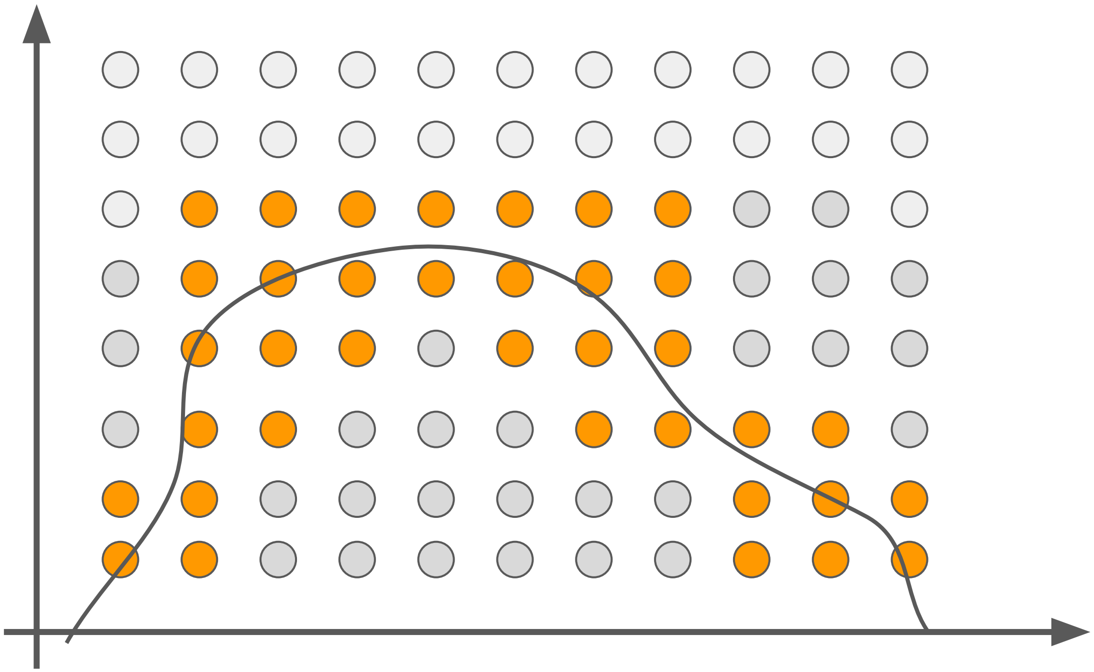
## What we want:

## An iterative algorithm that suggests points to evaluate that help the most in finding the contour.



## What we want:

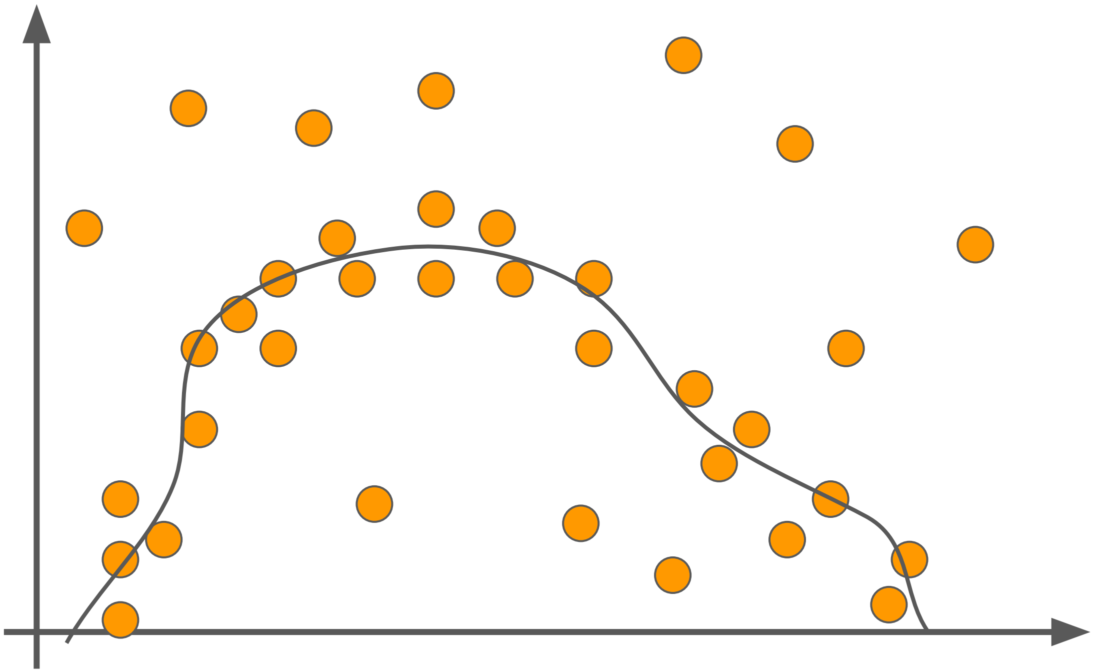
## An iterative algorithm that suggests points to evaluate that help the most in finding the contour.



## Intuitively speaking points closer to the hypersurface carry more information than those far away from it.

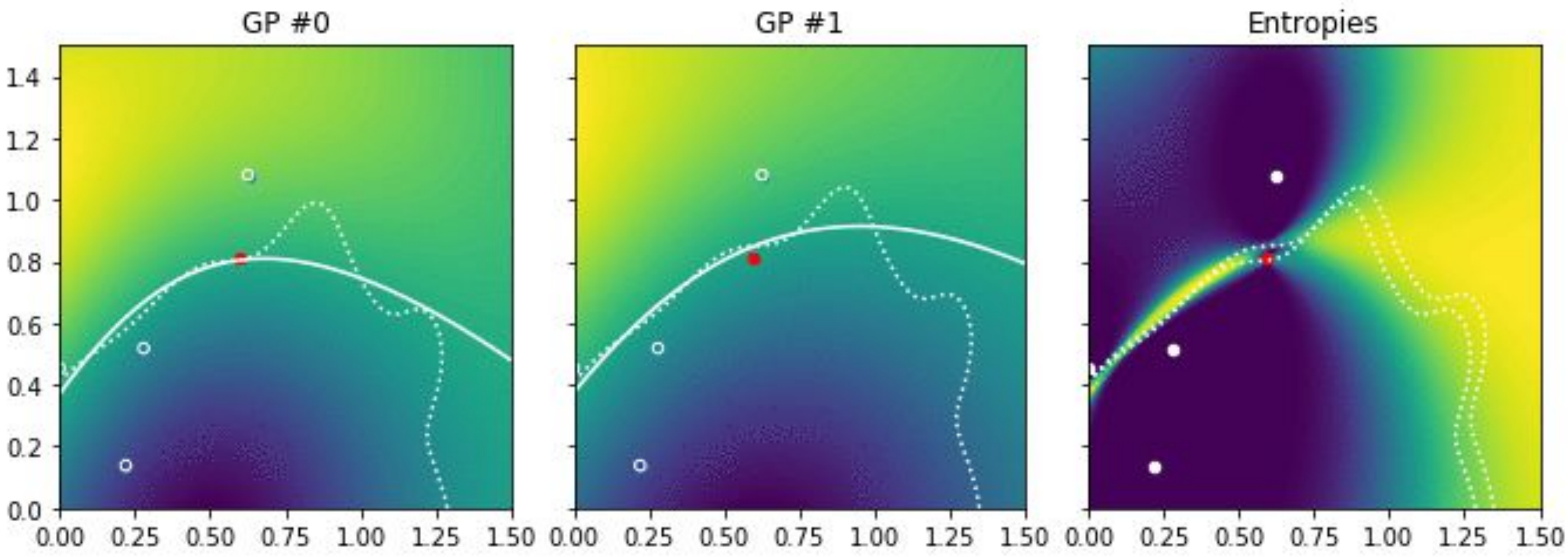
## What we want:

## An iterative algorithm that suggests points to evaluate that help the most in finding the contour.



## Intuitively speaking points closer to the hypersurface carry more information than those far away from it. Want strategy to sample that region densely.

# Where this is going:

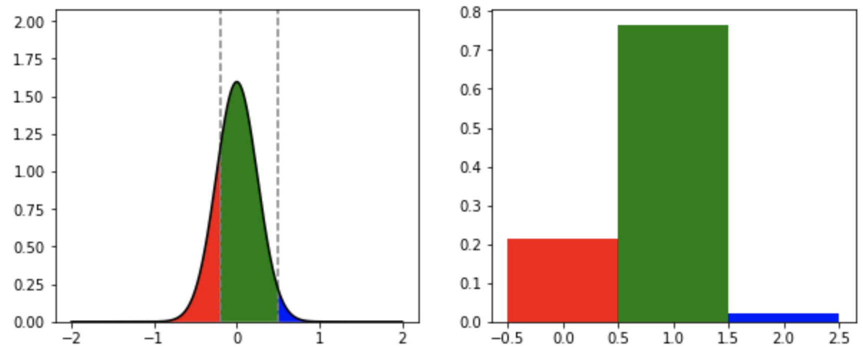


**Approach:**

**Do not need to learn the full function f(x). Only interesting property is its classification w.r.t. set boundaries**

**Given a prob. distribution of the function value p(f(x)) compute the entropy of discrete classification distribution S(c|x):**

$$H[S(x)] = \sum_i S(c_i|x) \log S(c_i|x)$$




**Can construct global entropy-based ambiguity measure:**

$$\langle S \rangle = \int H[S(x)]\, dx$$

**Gaussian Processes**: Provide us with probability densities p(x)

How to choose next best point? Pick one with best expected improvement in global ⟨S⟩

$$EI(x') = \int dx\, H[S] - E_{y(x')}[S(x)|Y(x')]$$

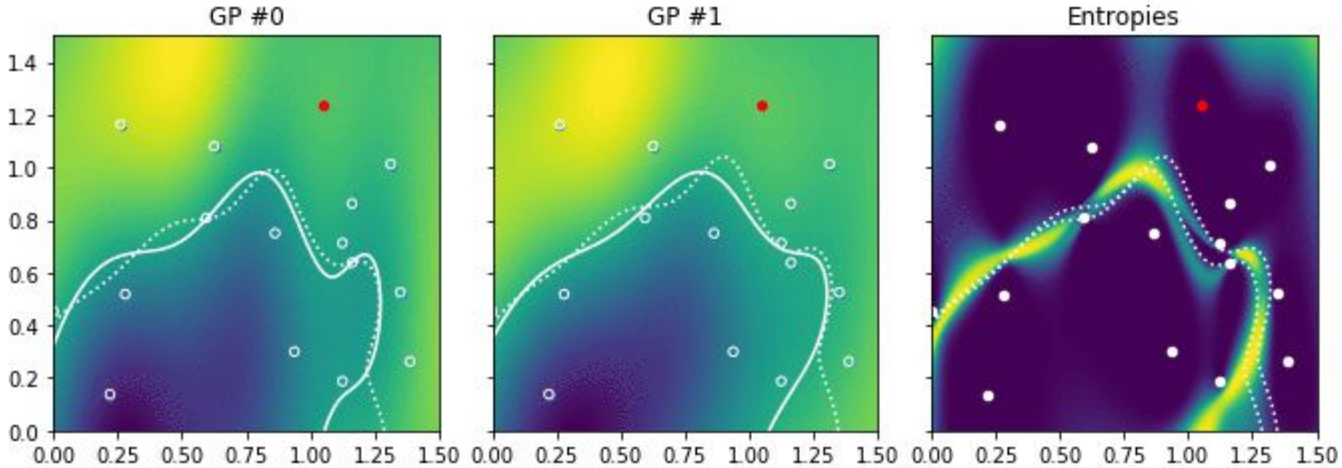
**Next point:** x = argmax EI(x)

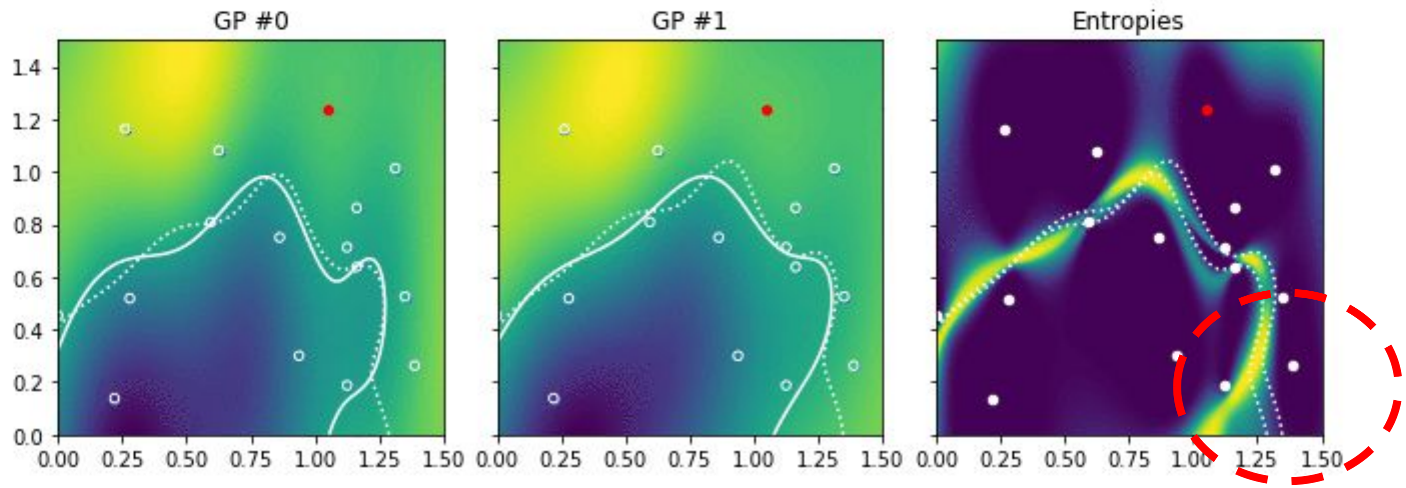
Problem: **involves integral over function domain → slow!**

Trick: use mutual information between S(x) and Y(x')

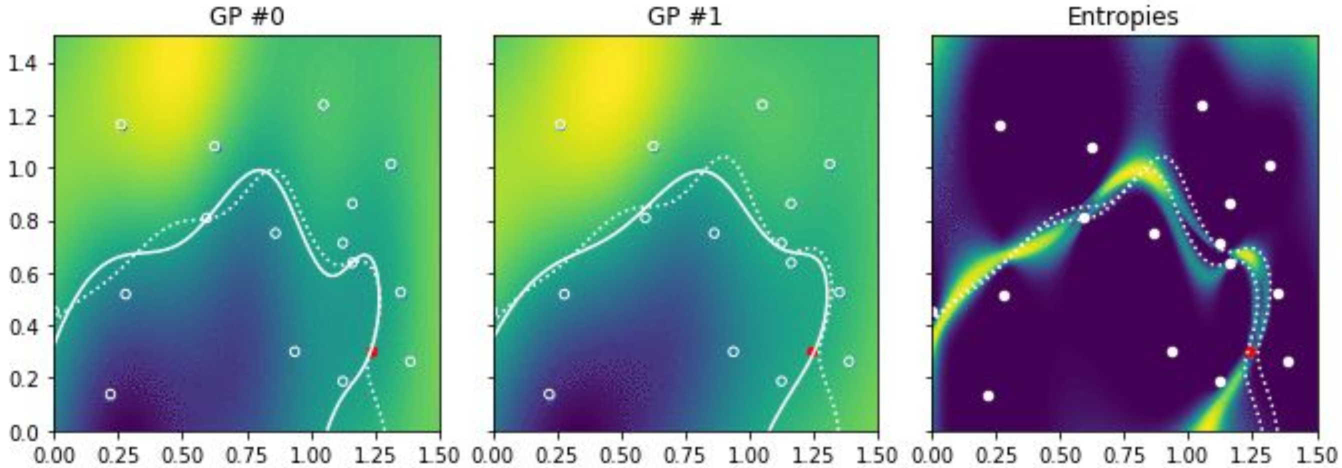
$$EI = H[S] - E_{y(x')}[S(x)|Y(x')] = H[Y] - E_s[Y|S]$$

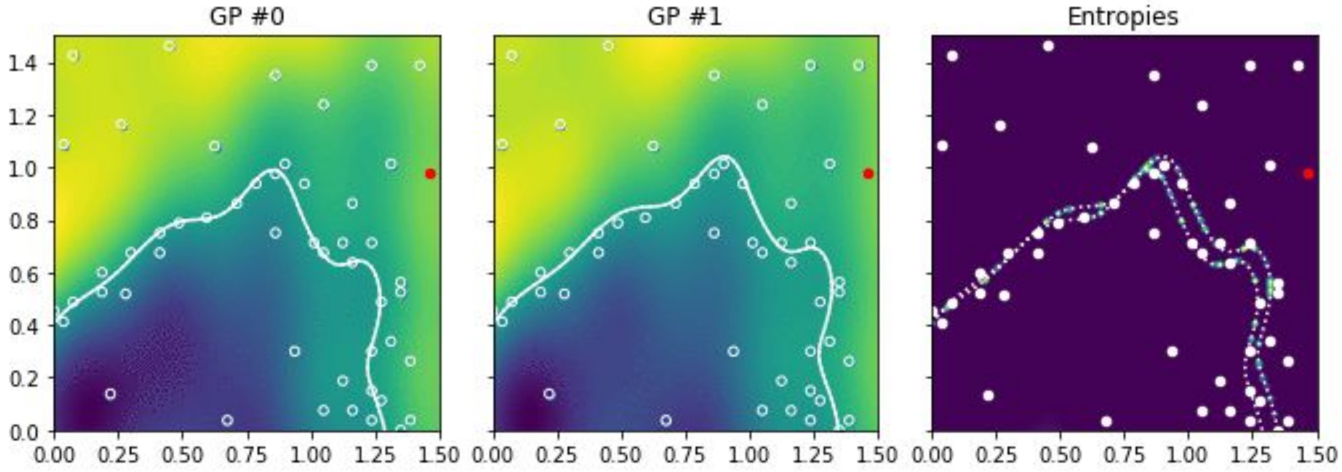
More tractable: estimate of EI can be computed analytically.


GP #0
GP #1
Entropies
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0.00
0.25
0.50
0.75
1.00
1.25
1.50



GP #0
GP #1
Entropies
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.00 0.25 0.50 0.75 1.00 1.25 1.50

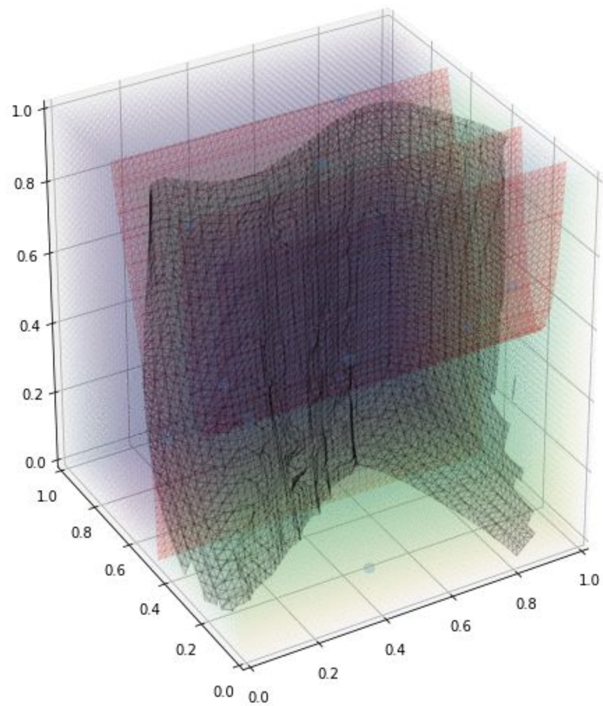

GP #0
GP #1
Entropies
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0.00 0.25 0.50 0.75 1.00 1.25 1.50

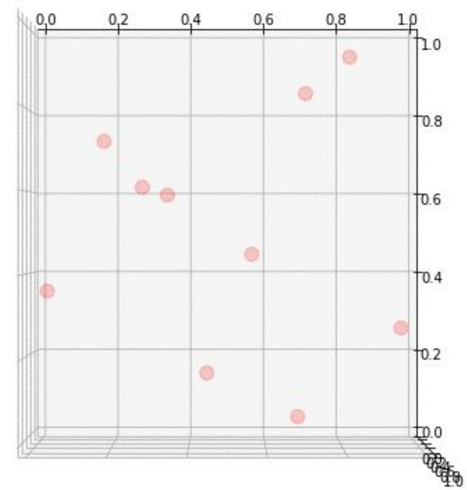


GP #0
GP #1
Entropies
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.00 0.25 0.50 0.75 1.00 1.25 1.50
0.00 0.25 0.50 0.75 1.00 1.25 1.50
0.00 0.25 0.50 0.75 1.00 1.25 1.50

# 3D Example:

## Use-case: Dark Higgs cross-section iso-surface.



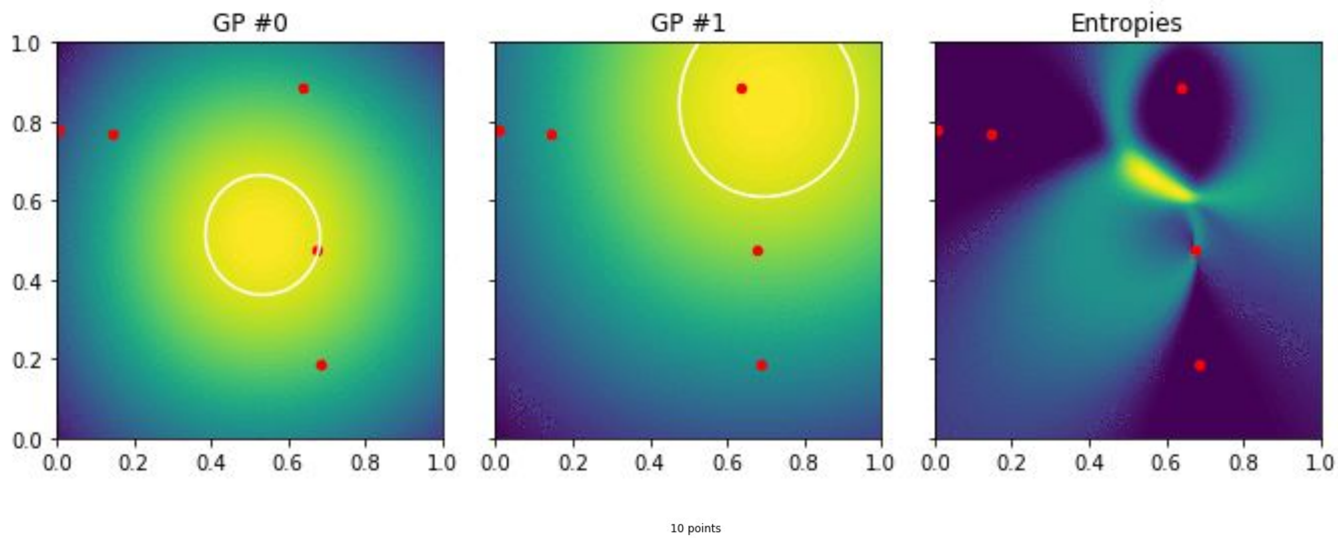

**Can clearly see interplay between exploration and exploitation. Sparse sampling of general volume, dense sampling close to surface**

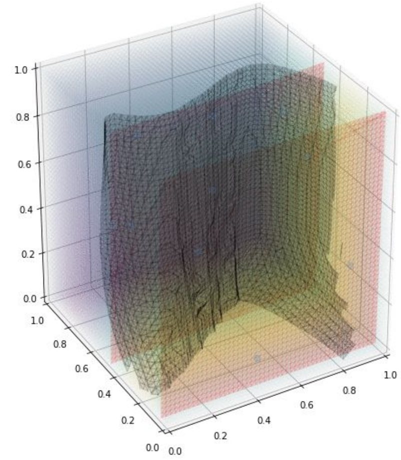
## Extensions: naturally follow-up questions:

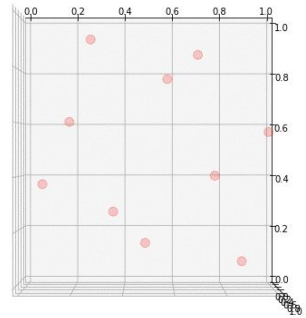
- Estimation of multiple levelsets of the same function (e.g. 1σ, 2σ contours)
  - Automatic just add more thresholds / extend discrete S(x)
- Estimation of (multiple levelsets) of multiple functions (e.g. expected and observed p-value)
  - Use average EI from two GPs

- **Batch Acquisition:** perhaps would not want to to point-by-point but in larger batches (parallel evaluation)
  - greedily build up batch - for each point sample evaluations from last known GP → iterate
  - For large batches re-sample batch values to avoid “lock-in” into a given evaluation history

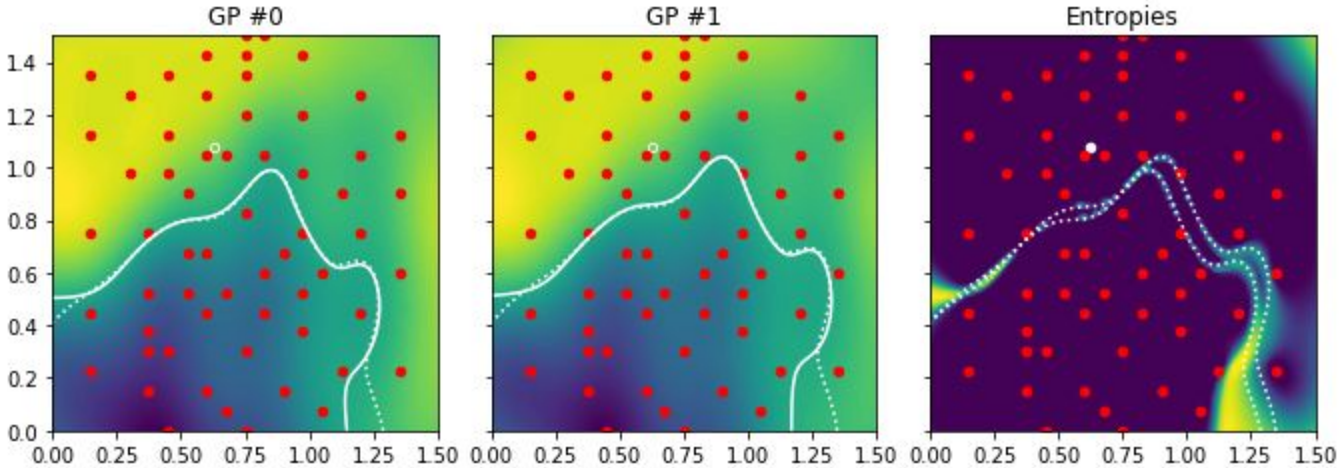
# Stable batches in which points do not interfere with each other.





“One-shot” limit: if all we can afford is a single iteration: would expect well-spaced sampling across full domain:



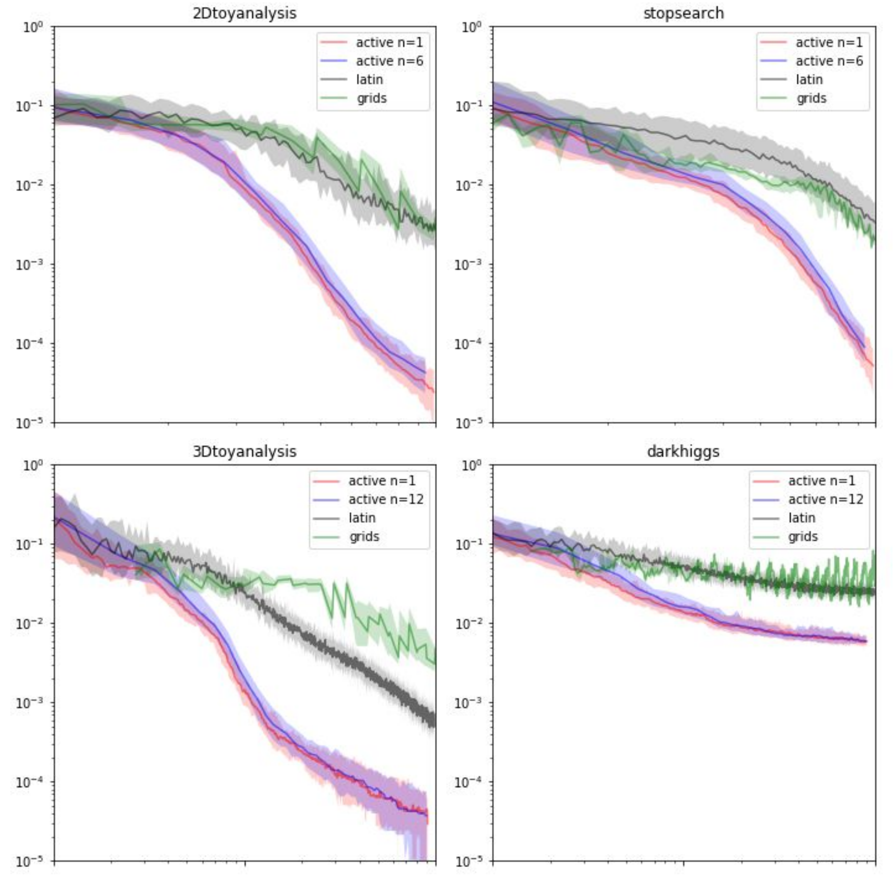

Possible scenario: broad base scan using random sampling, refine using few iterations.

Performance: compared to

1. Latin Hypercube
2. Regular Grids

Large factors of savings in total number of function evaluations.

- batched acquisition performs well compared to step-by-step acquisition



- **Python package**
  - **simple ask/tell interface**

README.md

# `excursion` — Efficient Excursion Set Estimation

DOI 10.5281/zenodo.1634427 | launch binder | build passing

This package implements a Bayesian Optimization procedure based on Gaussian Processes to efficiently determine excursion sets (or equivalently iso-surfaces) of one or many expensive black-box functions.

## Installation and Example

Install via `pip install excursion==0.0.1a0`.

- **Next steps:**
  - **GP scaling difficult for high-dimensional spaces**
  - **work on scaling speed up using e.g GPyTorch (GPU-accelerated Gaussian Processes)**