Active Learning for Excursion Set Estimation

Jason Hipkins
Mentors: Lukas Heinrich, Irina Espejo, Kyle Cranmer
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 \( f: \mathbb{R}^n \to \mathbb{R} \),

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

...
Naive Approach

1. Regular / Cartesian Grids

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

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

2. Random Samplings

e.g. Uniform Samplings, Poisson Disc, Latin Hypercube, known priors... Works in higher dimensions.

Disadvantages:
- sampling density fixed and not informed by the function $f(x)$
Example in HEP: Interval Estimation during Inference
e.g. exclusion contours
Why do we care: **Testing Beyond Standard Model Theories (BSM)**

- The “black-box” functions are computationally expensive
- e.g. Finding the p-value of a given BSM theory requires an expensive computational pipeline

  Evgen > Simulation > Data Reduction > Event Selection > Inference

- **Key Question:**

  What is a computationally efficient strategy
For finding level set estimates of expensive
black-box functions?
The naive approaches suffer from the **“curse of dimensionality.”** BSM Theories we are interested in can have 5, up to 19 dimensions. **It is infeasible to make exclusion plots in higher dimensions so we need a more efficient approach.**

From Lukas' Talk at ACAT 2019
New Approach: Using Gaussian Processes

A generalization of multivariate normal distribution to *stochastic fields*, such that for any vector of points,

\[ Y(x) = N(\mu(x), k(x, x')) \]

With given dataset D, we can construct levelset estimates via the GP as well as assess the (average) uncertainty of those estimates.

We use the SciKit-Learn and GPyTorch backends to fit these gaussian processes.

Bayesian Optimization: optimize an objective function through *sequential design*. I.e. improving the model by using prior information to choose new, optimal, points to evaluate in an expensive black box function.
The Excursion Package: Active Learning Algorithm

Which point(s) should we evaluate next to improve quality of contours / excursion sets?

Strategy: using properties of normal distributions, derive acquisition functions indicating the quality/helpfulness of points in the parameter space to reaching the objective (low uncertainty about excursion sets)

Can construct global entropy-based ambiguity measure:

\[
H[S] = S_i(x) \log S_i(x)
\]

\[
\langle H[S] \rangle = \int dx' H[S(x')]
\]
Statement

The black box function is $f(\theta) \rightarrow p$-value.

The goal is to find the excursion set $E_t(f) = \{\theta | f(\theta) = t\}$ for a given a threshold $t$ with as few queries as possible.

Method

1. Start with dataset $\mathcal{D} = \{\theta_i, f(\theta_i)\}$
2. Train a Gaussian process $Y|\theta, \mathcal{D}$ with predictive mean $\mu_{Y|\mathcal{D}}(\theta)$ and covariance $\Sigma_{Y|\mathcal{D}}(\theta, \theta')$
3. Evaluate the (cheap) acquisition function $U_t(\theta)$ for all $\theta$ using $Y|\theta, \mathcal{D}$
4. Select new point $\theta^* = \text{argmax} U_t(\theta)$
5. Query the function at $f(\theta^*)$ and update dataset $\mathcal{D} \leftarrow \mathcal{D} \cup (\theta^*, f(\theta^*))$
The Excursion Package: Active Learning Algorithms

- Lukas Heinrich created a packaged called **excursion**
  - Implemented the heuristic using SciKit-Learn package as a backend
- Irina Espejo implemented the same heuristic using GPyTorch as a backend
  - Supports GPU acceleration of expensive computations
- Separate repositories and needed a common API
  - Why use both?

Ask-and-Tell Interface: Active approach to solving Bayesian optimization problems

1. “Ask” an “oracle” for a set a new points that satisfies a cheap objective function. In our case reducing entropy
2. Label these points i.e. compute expensive black box function at these points
3. “Tell” the GP the new data and fit the hyperparameters of the gaussian process to the new data
Refactored Excursion Package

- Abstraction
  - Optimizer doesn’t know about backend when solving excursion problems
- Inspired by scikit-opt, GPyTorch and BoTorch package structures
- Design patterns
  - Strategy (Behavioral)
  - Abstract Factory (Creational)
  - Proxy (Structural)
- Implemented “warm start” by understanding backends
  - Significant savings in training time
  - More testing needed
- Package Repository
  - https://github.com/diana-hep/excursion
Excursion Set Estimation through Active Learning

- Optimizer
  - Sampler
  - Model
  - Acquisition Function
  - Prob. Details
  - Result
  - ask()
  - tell(x, y, fit)
  - fit()
  - update_next()
  - get_result()

- Sampler
  - generate(n, grid)

- Model
  - fit_model(fit_optimizer)
  - update_model(x, y)

- Acquisition Function
  - acquire(model, thresholds, grid)

- Result
  - update_result(model, next_x, acq_vals, grid)

- Builders
  - build_model(model, **kwargs)
  - build_acq_func(acq_func, **kwargs)
  - build_sampler(generator, **kwargs)
  - build_result(prob_details, **kwargs)

- Problem Details
  - true_function
  - thresholds
  - bounding_box
  - ndim
  - rangedef
  - X_meshgrid
  - X_pointsgrid
  - acq_pointsgrid
  - Init_n_points
  - invalid_region(X)

- Learner
  - Problem Details
  - Algorithm Options
  - ask()
  - tell(x, y, fit)
  - evaluate(x)
  - evaluate_and_tell(x, fit)
  - run(n_iterations, plot_result, show_conf_matrix)
  - evaluate_metrics()
Success

- Diagnostics so far indicated it works and is faster.
- Learned:
  - Problems using warm starts
  - Off-the-shelf parallel processing package *joblib* has some machine dependent differences

![Graphs showing performance metrics](image-url)
Next Steps

- Finish Refactor
  - Batched Acquisition Abstraction
  - Multiple Function support
- Finishing merging GPyTorch backend
- Benchmarking on different machines
- Adding BoTorch as a backend option
- Dockerizing pipeline process to use excursion for testing theories
Resources

- [https://indico.cern.ch/event/702612/contributions/2958660/attachments/1649620/2638023/Contours.pdf](https://indico.cern.ch/event/702612/contributions/2958660/attachments/1649620/2638023/Contours.pdf)
- [https://hackmd.io/@irinaespejo/HyiUFSmPL](https://hackmd.io/@irinaespejo/HyiUFSmPL)
- [https://github.com/scikit-optimize/scikit-optimize/issues/68](https://github.com/scikit-optimize/scikit-optimize/issues/68)
- Data Science applications to accelerate High Energy Physics, DQE presentation (05/07/2021), Irina Espejo (NYU)
- [https://indico.cern.ch/event/708041/contributions/3269754/](https://indico.cern.ch/event/708041/contributions/3269754/)
1. Create Excursion Problem Details Object
   a. Returns object with details of problem
2. Create optimizer objects, i.e. model, builder etc, using builders.
3. OR Use preloaded algo options from yaml file
4. Initialize Learner object OR Optimizer directly

Active Learning Loop
1. Call ask() : returns next_x from acquisition (or initialization until empty)
2. Evaluate x with block box or truth function
   a. Learner object can call function handle. learner.evaluate(x)
3. Call tell(x, y) : returns excursion result object
   a. Will fit the GP hyperparams by default and update the acquisition values
4. Go to 1
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.