Distributed Hyperparameter Search (HPS) with DeepHyper

Romain Egele, Prasanna Balaprakash, Misha Salim, Sam Foreman

Simulation, Data and Learning Workshop (October 6th 2021)

The DeepHyper Project

"Automated development of machine learning algorithms to support scientific applications"

Prasanna Balaprakash

Romain Egele

DeepHyper

Open-Source

https://deephyper.readthedocs.io/

The DeepHyper Community


Misha Salim


Stefan Wild


Venkatram Vishwanath


Romit Maulik


Bethany Lusch


Kyle Gerard Felker


Taylor Childers


Tom Uram


Elise Jennings


Matthieu Dorier


Sandeep Madireddy


Sam Foreman

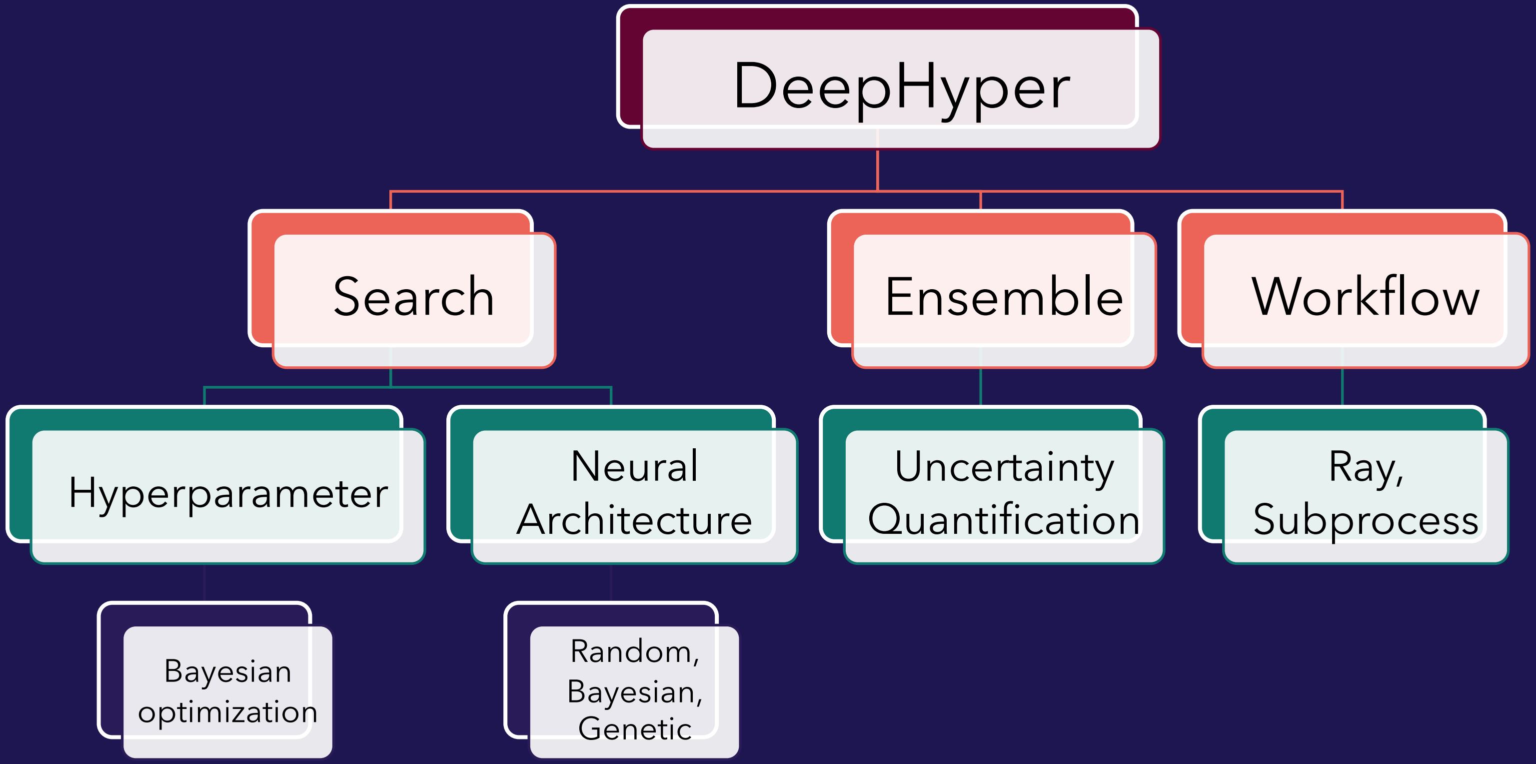

Shengli Jiang


Mansi Sakarvadia


Jaehoon Koo


Tanwi Mallick

DeepHyper Overview



[DeepHyper documentation: http://deephyper.readthedocs.io](http://deephyper.readthedocs.io)

Installed on ALCF systems

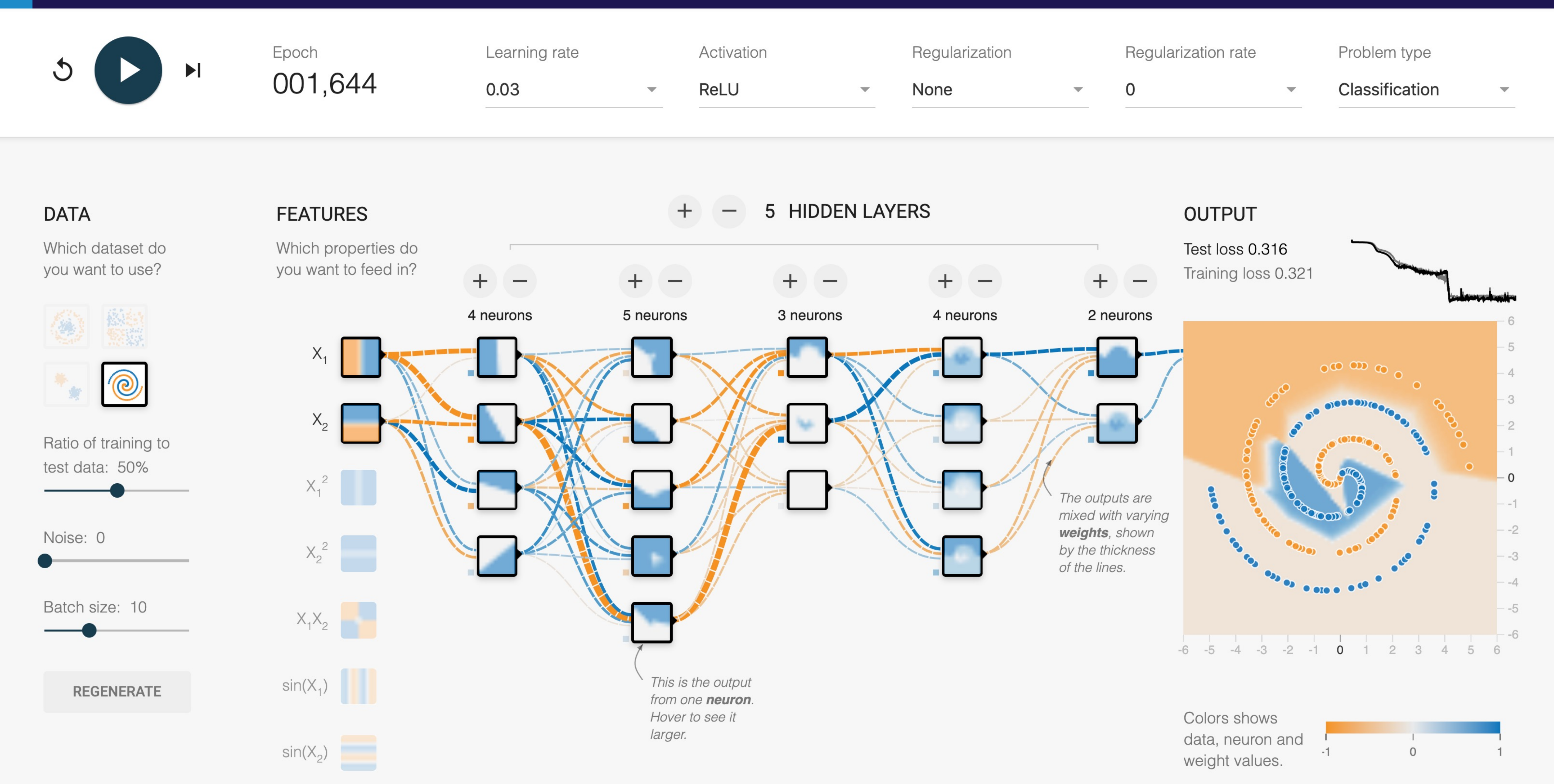
- **Theta**

```
$ module load conda/2021-09-22
```

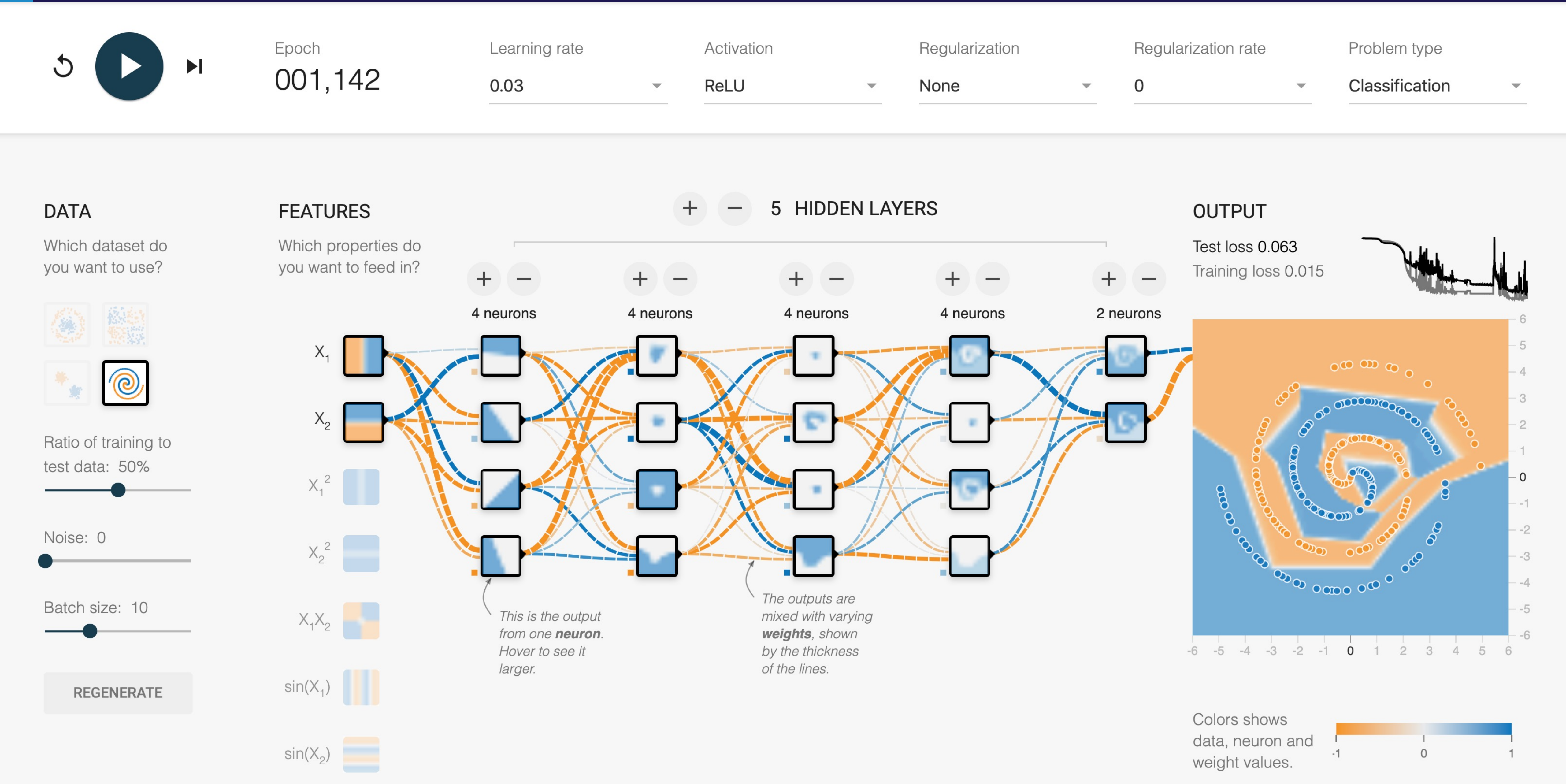
- **ThetaGPU**

```
$ module load conda/2021-09-22
```

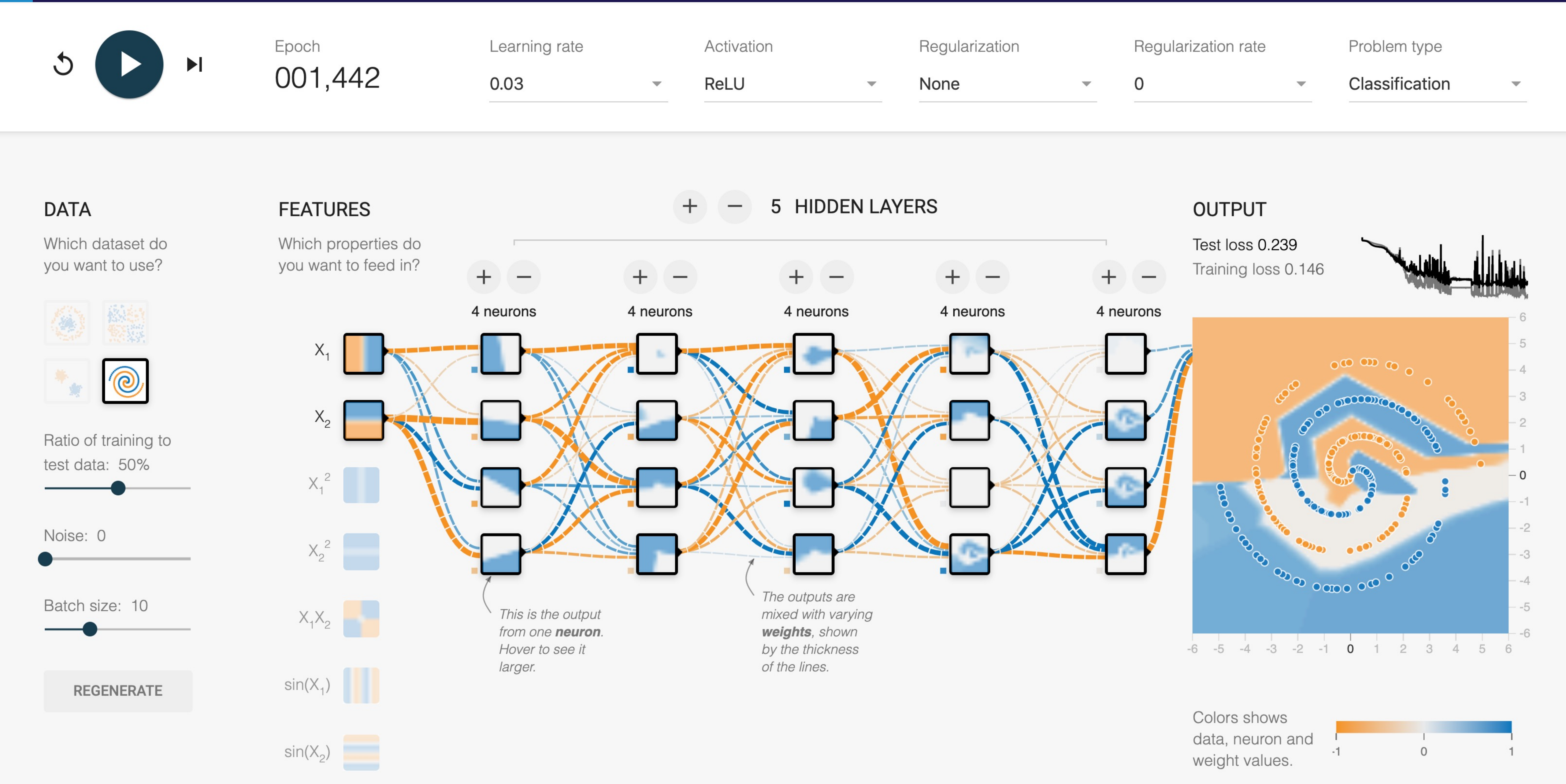
Warning: After loading the module, don't forget to run `$ conda activate base`



https://playground.tensorflow.org/



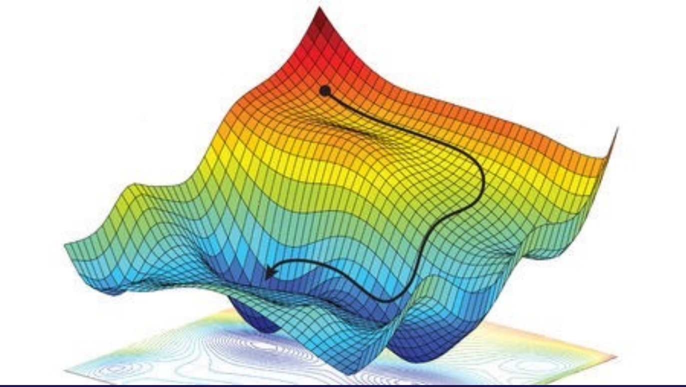
https://playground.tensorflow.org/



https://playground.tensorflow.org/

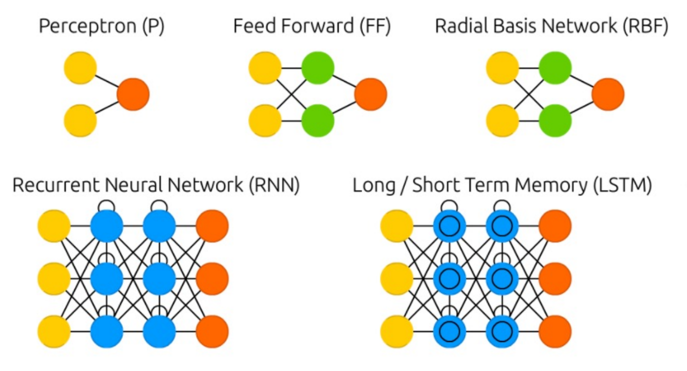
Hyperparameters of Neural Networks

Algorithm Hyperparameters



Optimizer: SGD, RMSprop, Adam…
Learning rate
Mini-batch size
Learning rate scheduler
Adaptative batch size
…

Model Hyperparameters



Number of layers
Type of the layer: Fully Connected, Convolution, Recursive…
Activation function
Dropout rate
Skip connection
…

Hyperparameters Search Problem

Lower-level problem: *Training data "T"*

$$\min_{w} \ \mathrm{err}_T(h; T; w)$$

Upper-level problem: *Validation data "V"*

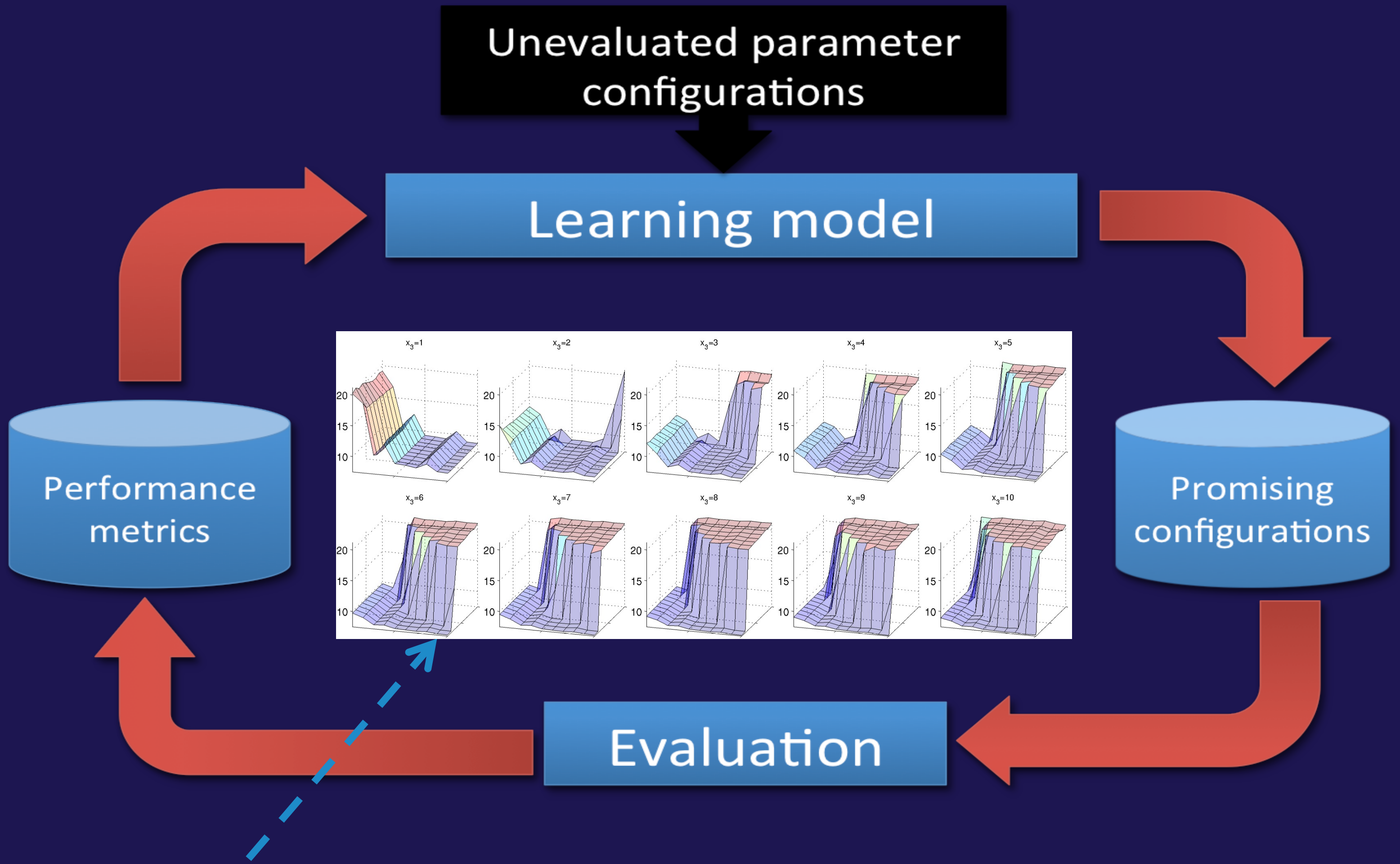
$$\min_{h} \mathrm{err}_V(h; V; w^*)$$

Machine-Learning Based Search

Two Phases

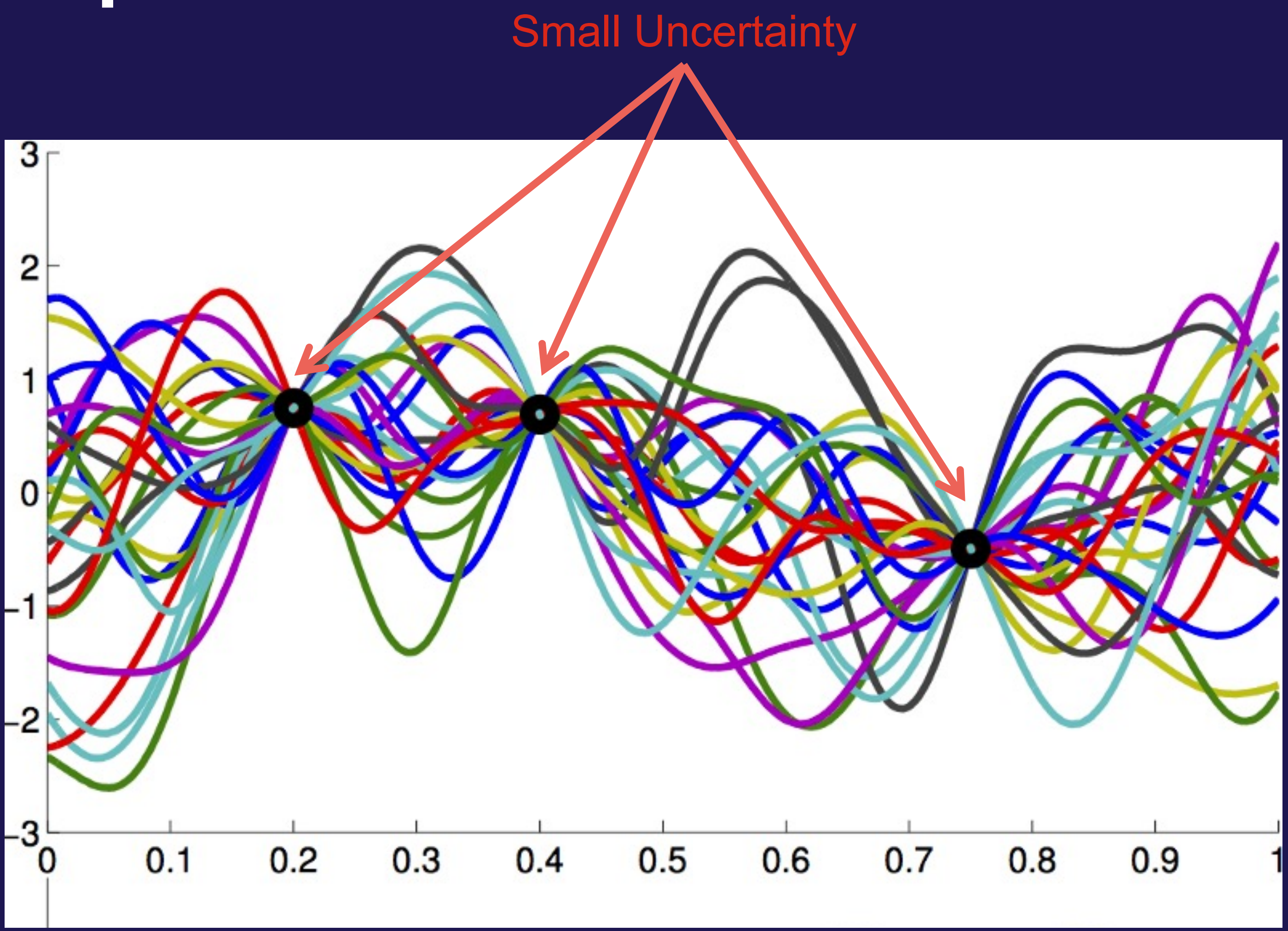
1. Initialization
   - Random sampling of hyperparameter configurations

2. Iterative
   - Fit the model to collected (configuration, error)
   - Sample using the model



Example Surrogate Model Fitted to Sampled Performance (iterative refinement improves the learning model)

Bayesian Optimization

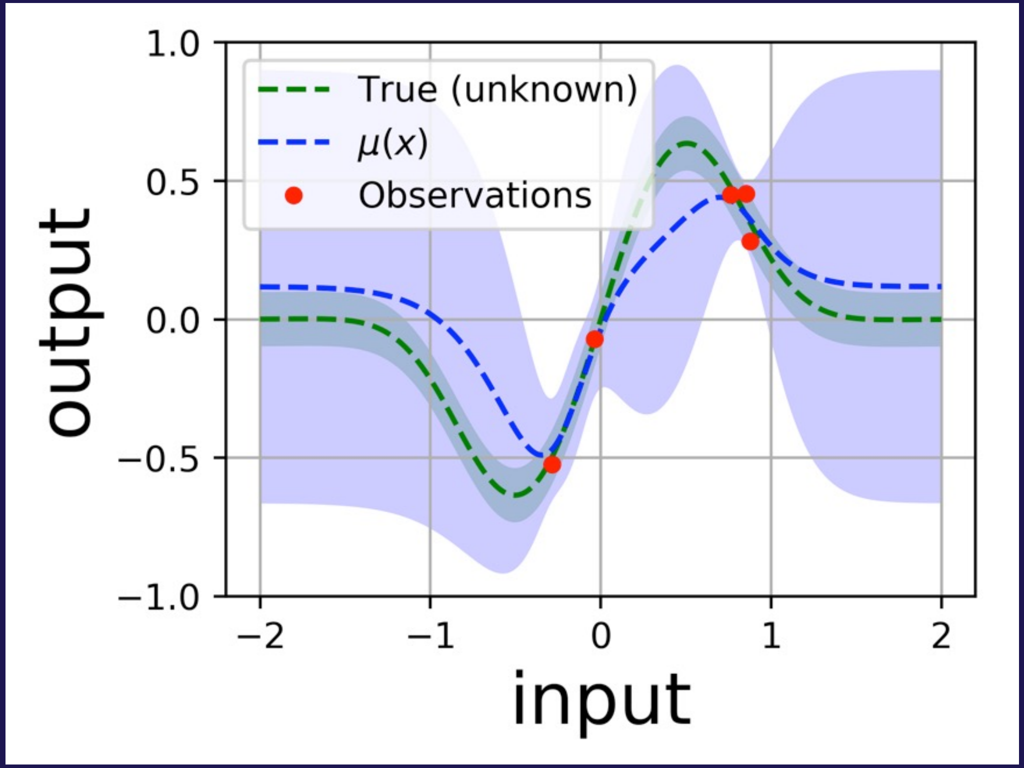


- Usual Gaussian process regression cannot handle discrete space natively
- Appropriate methods: random forest, extra tree regressor

Acquisition Function

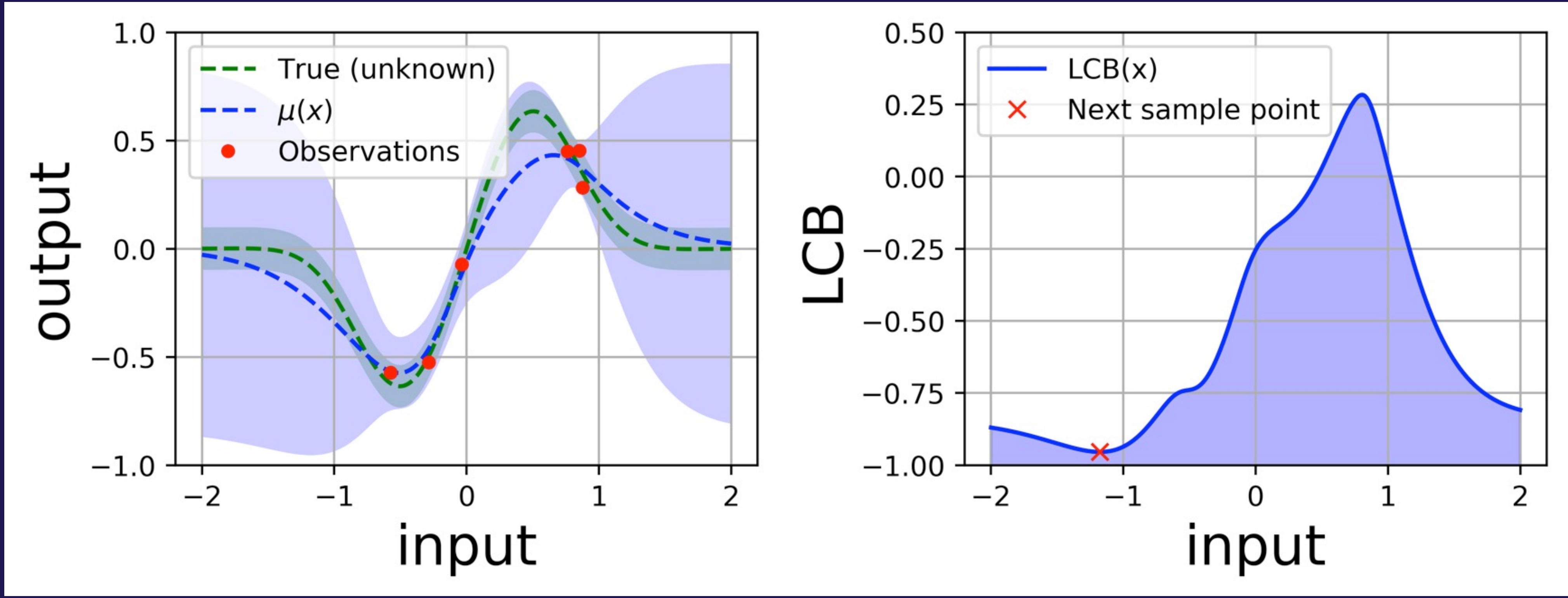
= 1.96
(exploration/exploitation)

$$\mathrm{LCB}(h;\kappa) = \mu(h) - \kappa \cdot \sigma(h)$$



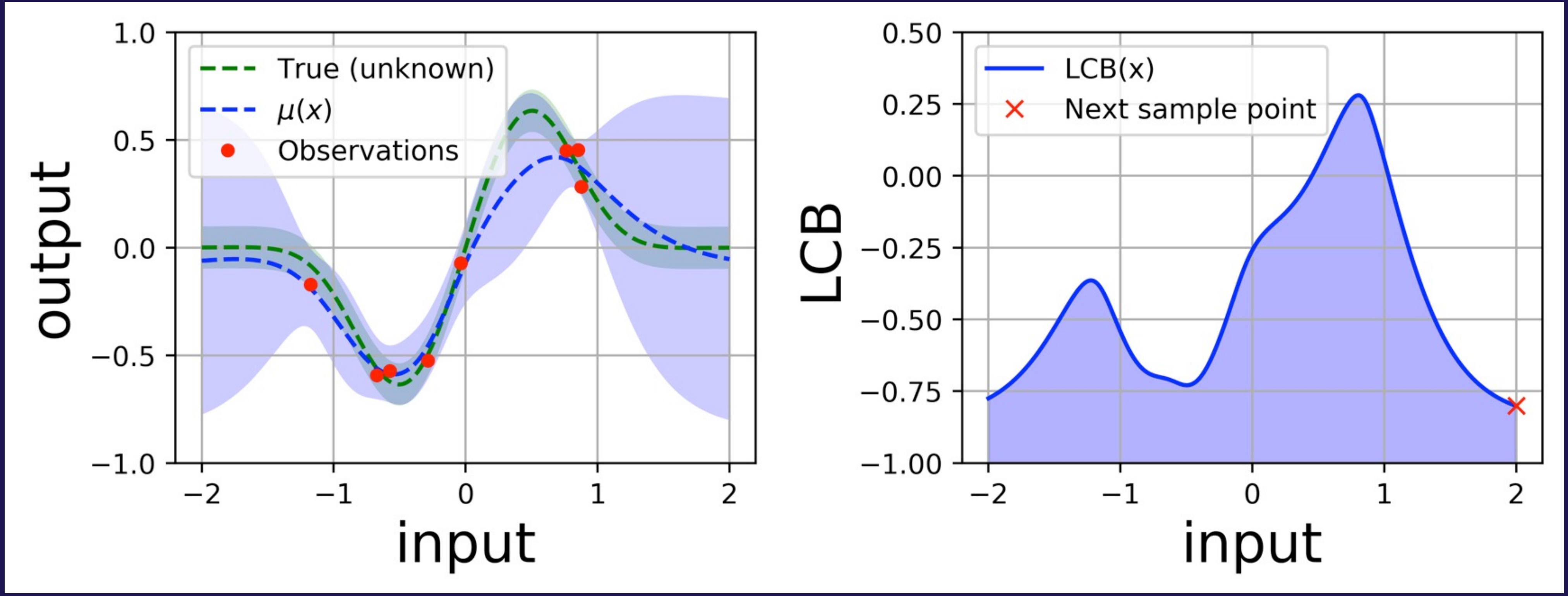
Acquisition Function

$$\mathrm{LCB}(h;\kappa) = \mu(h) - \kappa \cdot \sigma(h)$$



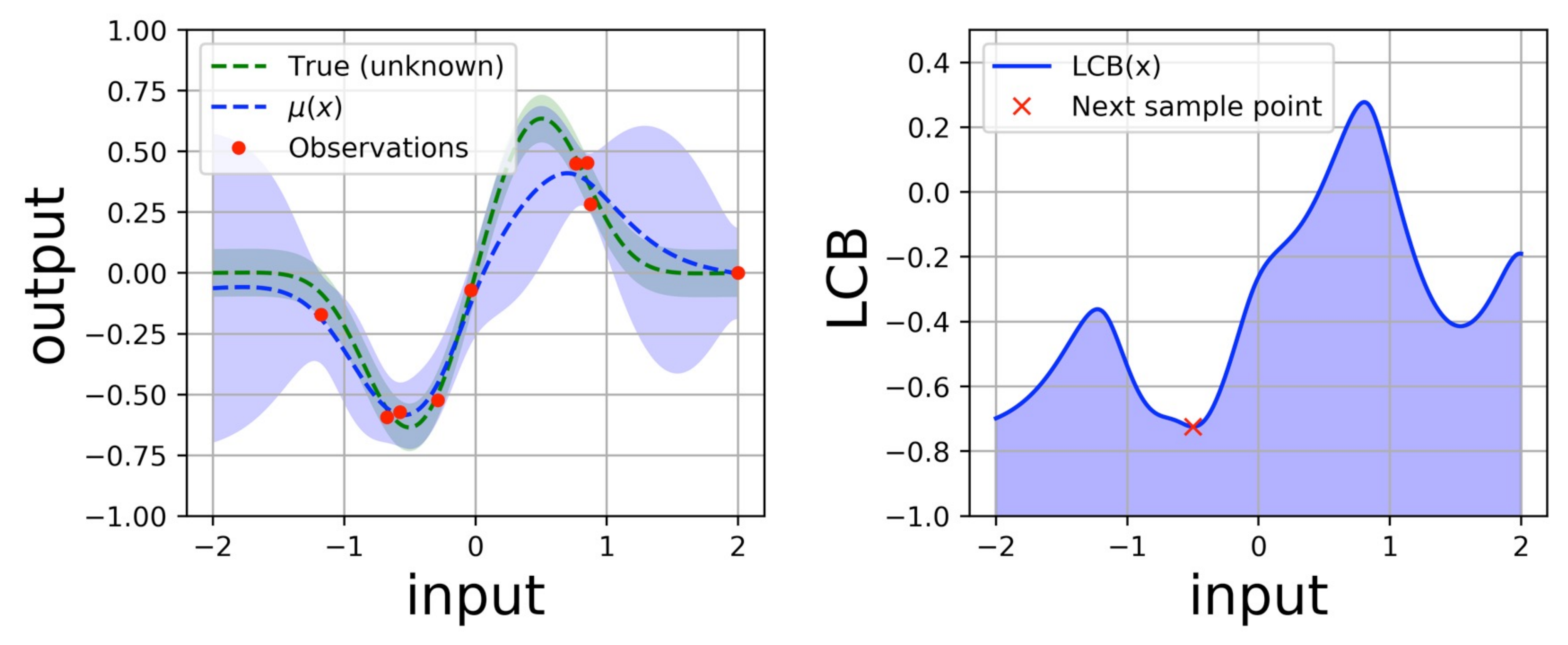
Acquisition Function

$$\mathrm{LCB}(h;\kappa) = \mu(h) - \kappa \cdot \sigma(h)$$

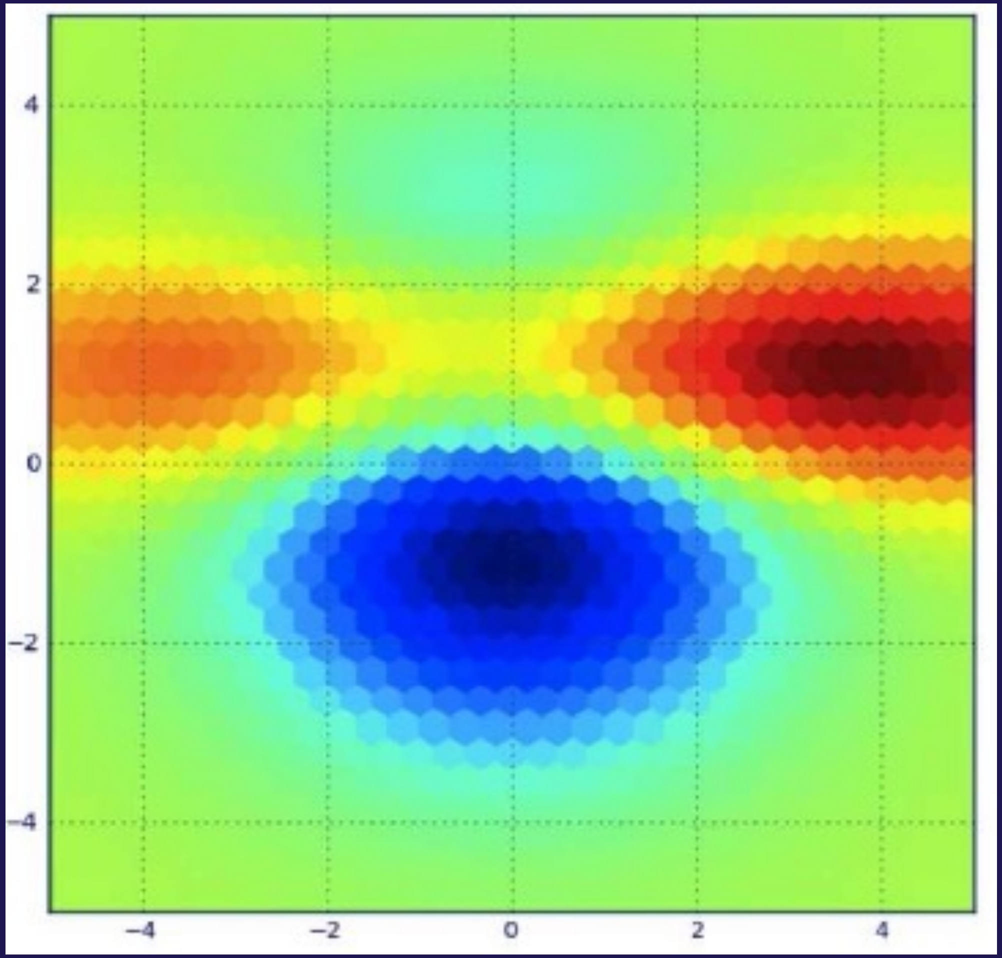


Acquisition Function

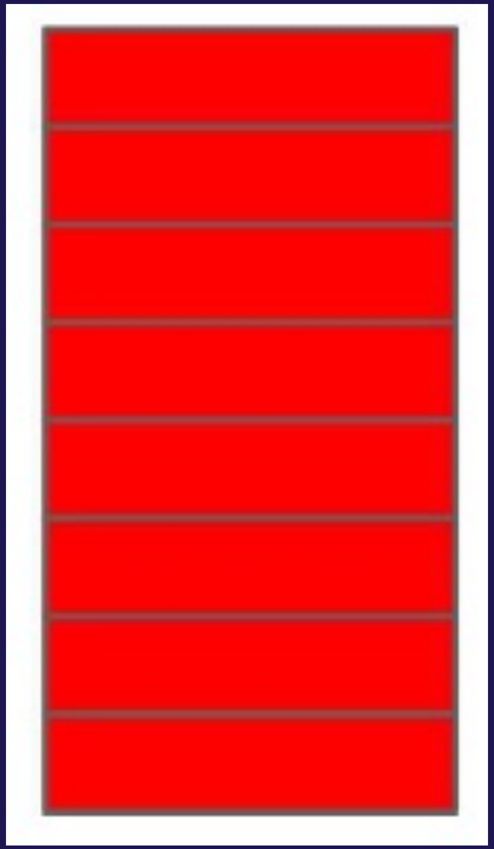
$$\mathrm{LCB}(h;\kappa) = \mu(h) - \kappa \cdot \sigma(h)$$



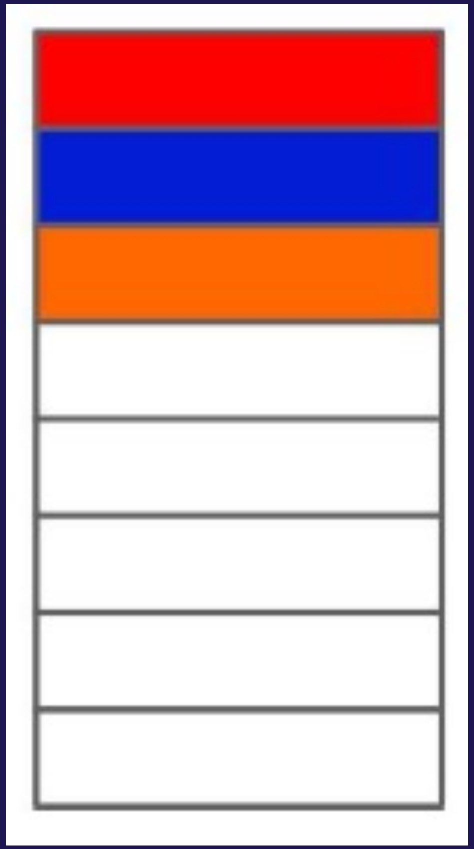
Multipoint Asynchronous Acquisition Function



Naive



Conditioned



Constant Liar Strategy for Asynchronous Update

0. Save true surrogate model and use a clone
   (re-used true when a new evaluation is done)

1. Multi-Point Acquisition (repeat to generate N configurations)

1. Select $\hat{h} = \underset{h}{\operatorname{argmin}}\, \mathrm{LCB}(h; \kappa)$

2. Fit clone with lie "L"

Hyperparameters with Constraints

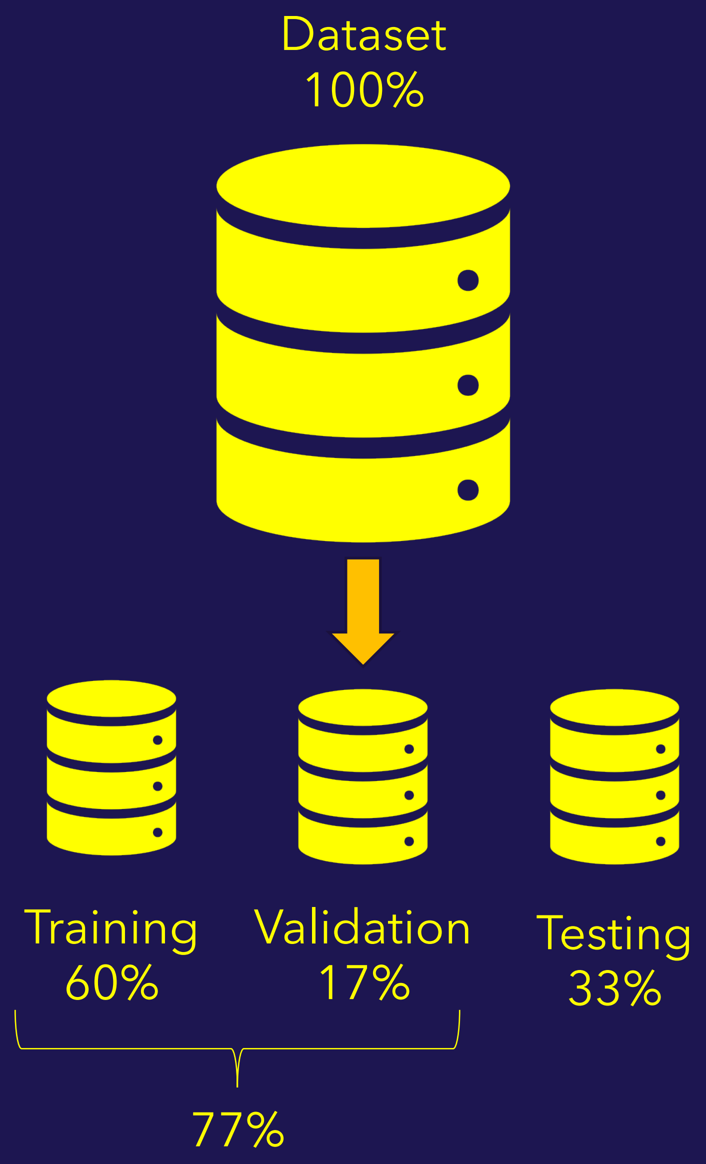
- Hierarchical hyperparameters
  - h1 “ number of layers”: (1, 10)
  - h2 “number of neurones in layer 1”: (1,100)
  - h3 “number of neurones in layer 2”: (1,100)
    - exist if number of h1 >= 2 so it is conditioned on the value of h1
  - h4 …
- Forbidden Configurations
  - h1 “ number of layers” ≠ 7

Scale with DeepHyper

- If evaluations are:
  - **Fast** (it is not useful to scale)
    - Overhead of surrogate model re-fitting
    - Overhead of communication
  - **Reasonably long** (few evaluations are performed but more can help improve the objective)
    - Increase the number of nodes used (adapt the number of DeepHyper workers)
  - **Excessively long** (the search cannot iterate, e.g. do not finish during the allocated time)
    - Speed-up the training evaluation
      By using more resources (CPUs, GPUs, Nodes) such as data-parallel training with Horovod
    - Allocate a reasonable time budget to your model computation
      By using a specialized Keras Callback to stop after some time
    - Reduce the data by sub-sampling the training data but keep the same validation data
    - Reduce the computational complexity of the model (e.g., less weights)
      By verifying tested hyperparameters (be careful with (fully-connected layers and big matrix multiplications)
    - Cache loaded data on local node memory `"$ /dev/shm" (whenever possible)`

- Scale the search space (more hyperparameters)
  - Adapt the distributed computation of the surrogate model with `"n_jobs"` (number of local processes used in parallel to fit the model)
  - Adapt the surrogate model (e.g., Extra Trees is faster to compute than Random Forest)

Problem Setup



load_data.py

```
def load_data():
    …
    return Training, Validation
```

problem.py

```
Problem = HpProblem()

Problem.add_hyperparameter(h1, (1, 10))

Problem.add_starting_point(h1=2)
```

Application to optimize

run.py

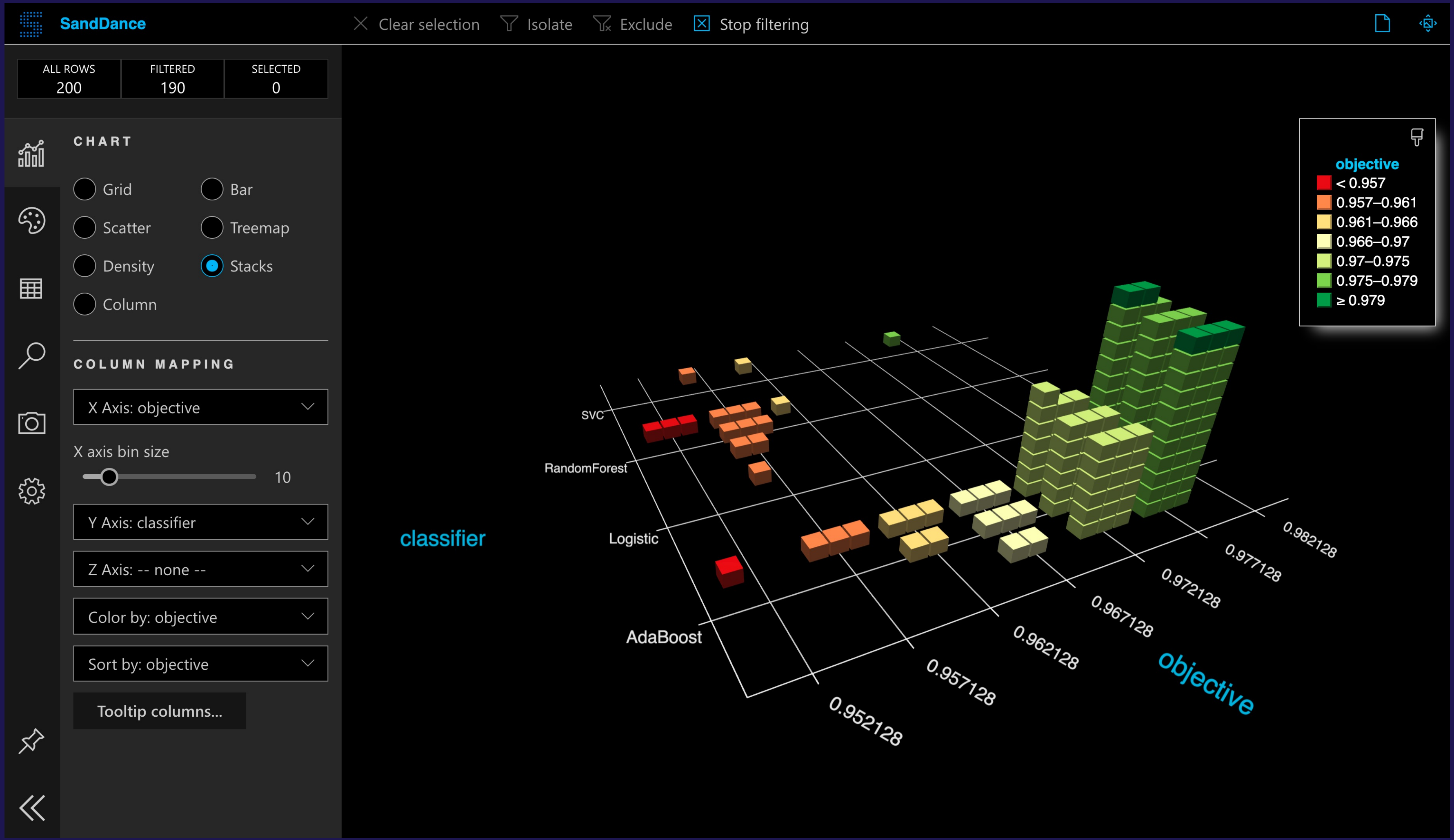
```
def run(configuration):
      set_random_state(seed)
      training, validation = load_data()
      model = create_model(configuration)
      model.fit(training)
      score = model.evaluate(validation, metric)
      objective = compute_objective(score)
      return objective
```

Understand the results (1)

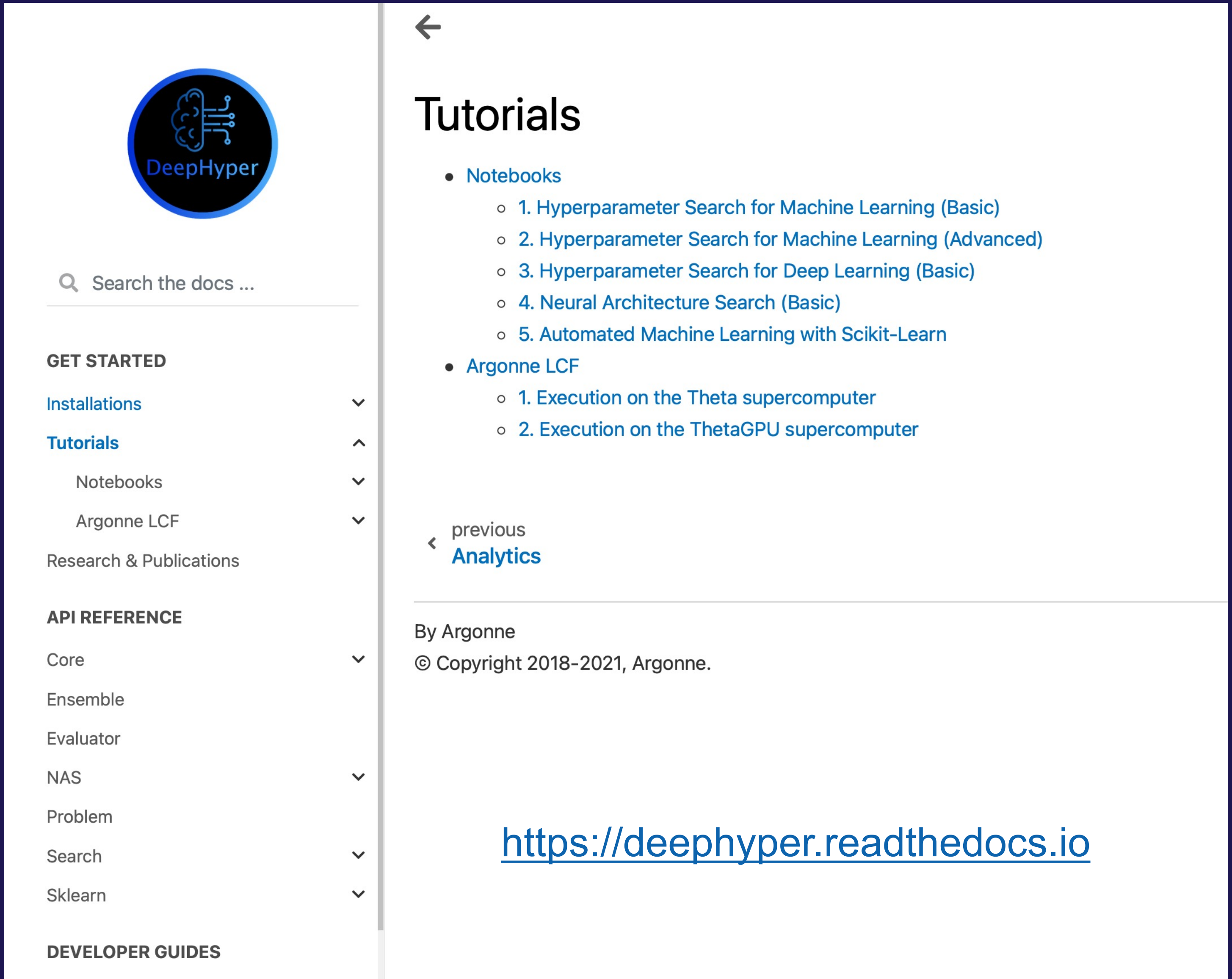
```
classifier,C,alpha,kernel,max_depth,n_estimators,n_neighbors,gamma,objective,elapsed_sec
AdaBoost,nan,nan,NA,nan,187,nan,nan,0.9627659574468085,3.8781380653381348
AdaBoost,nan,nan,NA,nan,19,nan,nan,0.9627659574468085,7.370249271392822
SVC,0.910144037187624,nan,linear,nan,nan,nan,nan,0.9574468085106383,11.247097969055176
Logistic,0.05670441459759125,nan,NA,nan,nan,nan,nan,0.9574468085106383,15.790768146514893
AdaBoost,nan,nan,NA,nan,1662,nan,nan,0.973404255319149,22.461848974227905
RandomForest,nan,nan,NA,64,561,nan,nan,0.9574468085106383,26.977345943450928
RandomForest,nan,nan,NA,15,1812,nan,nan,0.9574468085106383,33.18859791755676
```

Understand the results (2)

Visual Studio Code + SandDance



Learn more about DeepHyper



https://deephyper.readthedocs.io

Argonne NATIONAL LABORATORY

A Tutorial for Hyperparameter Optimisation

jupyter 01_HPS_basic_classification_with_tabular_data Last Checkpoint: 22/09/2021 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted

Python 3

Run

Markdown

DeepHyper

Hyperparameter search for classification with Tabular data

Reference

This tutorial is based on materials from the Keras Documentation:

- Structured data classification from scratch

Warning

By design asyncio does not allow nested event loops. Jupyter is using Tornado which already starts an event loop. Therefore the following patch is required to run this tutorial.

This tutorial should be run with `tensorflow>=2.6`.