Distributed Hyperparameter Optimization and Model Search with Examples using SHADHO

Jeffery Kinnison, Sadaf Ghaffari, Nathaniel Blanchard, Walter Scheirer

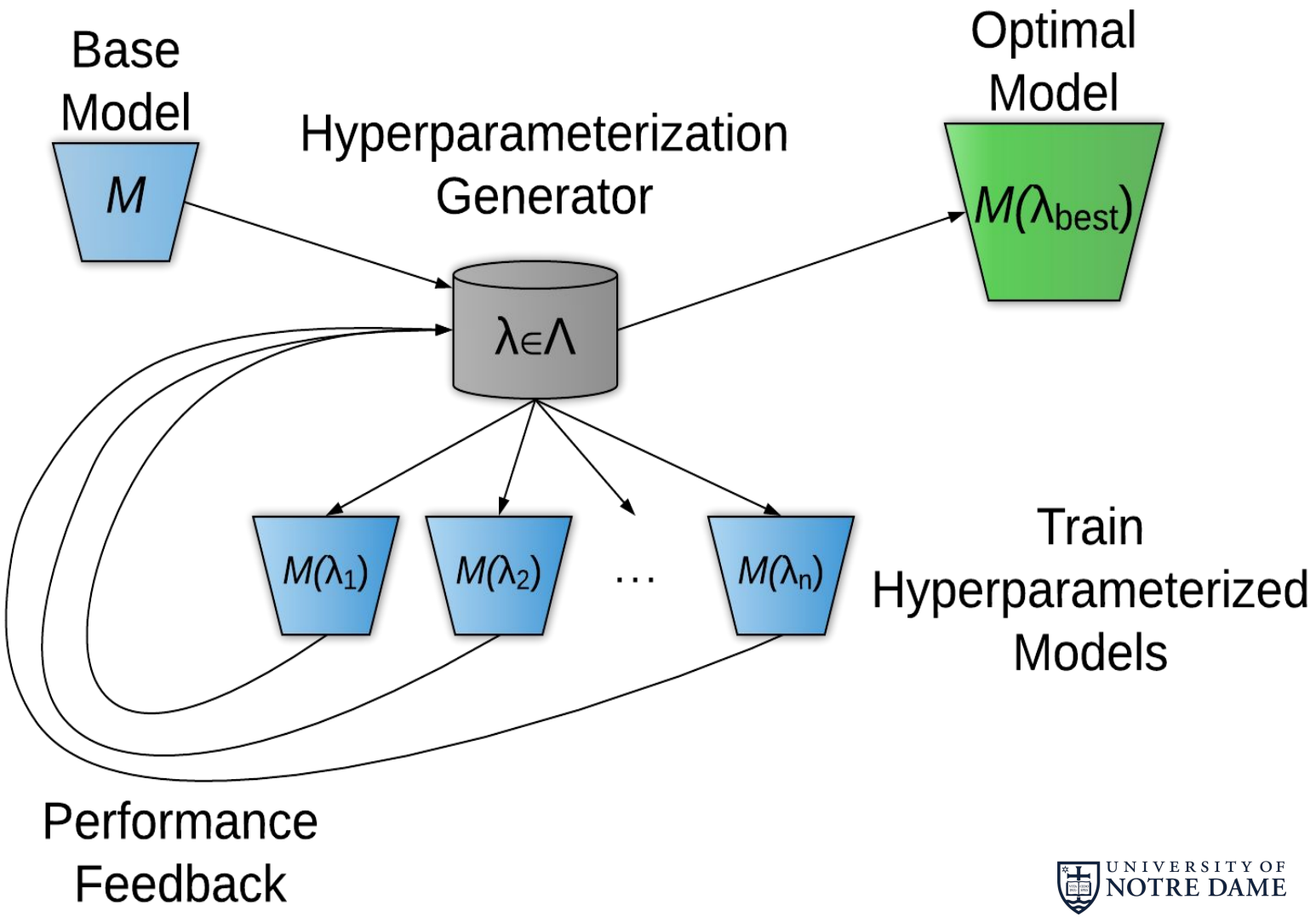
Overview

1. Scalable Hardware-Aware Distributed Hyperparameter Optimization
2. A First Example
3. Optimizing SVM
4. Optimizing Neural Network Structure
5. Going Forward

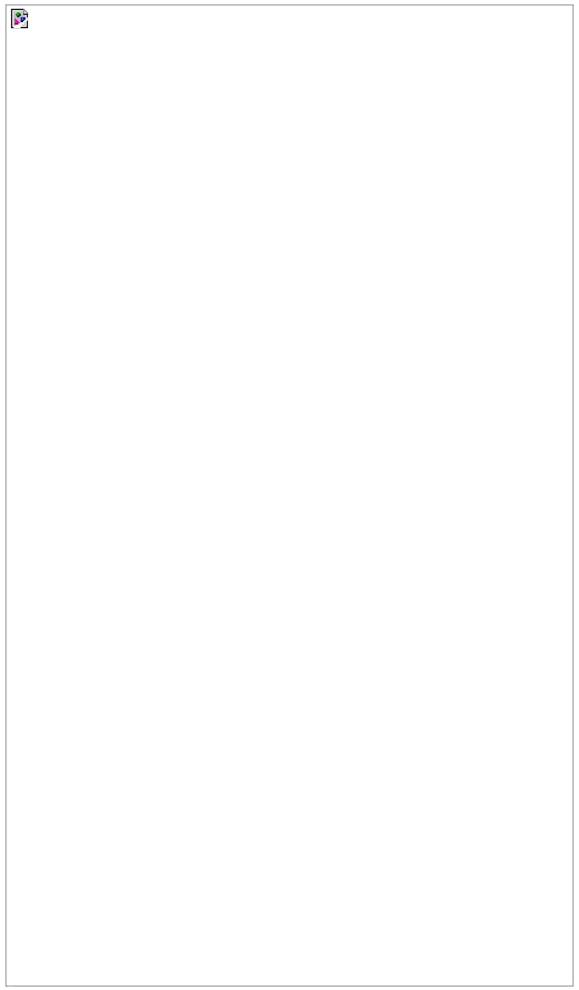
1. Scalable Hardware-Aware Distributed Hyperparameter Optimization

https://jeffkinnison.github.io/shadho-tutorial/

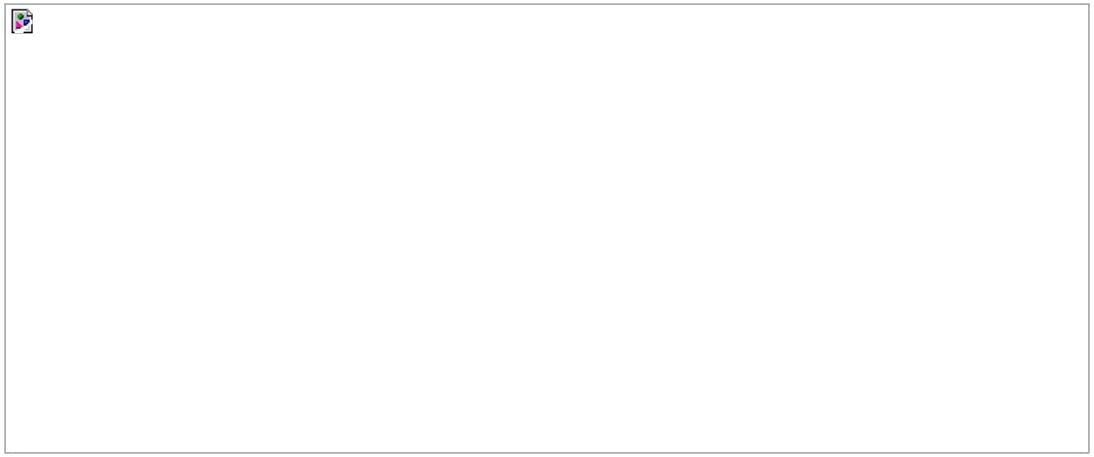
Distributed Hyperparamter Optimization



Structure of a Hyperparameter Search

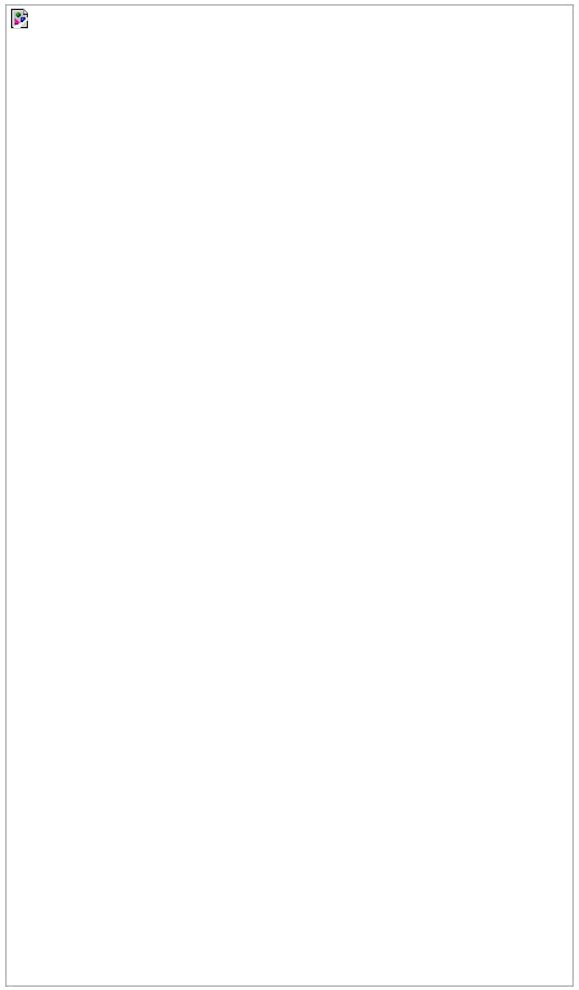


CNN Architecture Search Domains

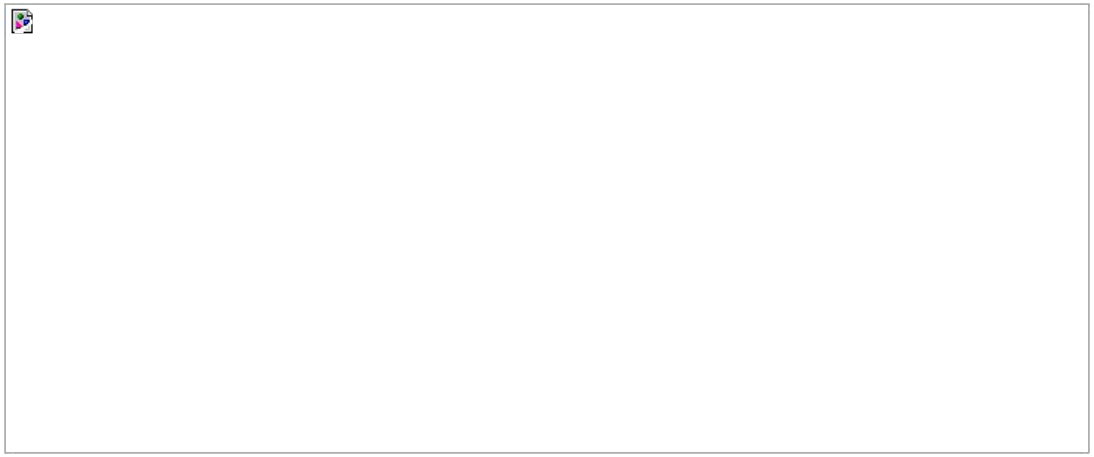


SVM Kernel Hyperparameter Domains

Structure of a Hyperparameter Search

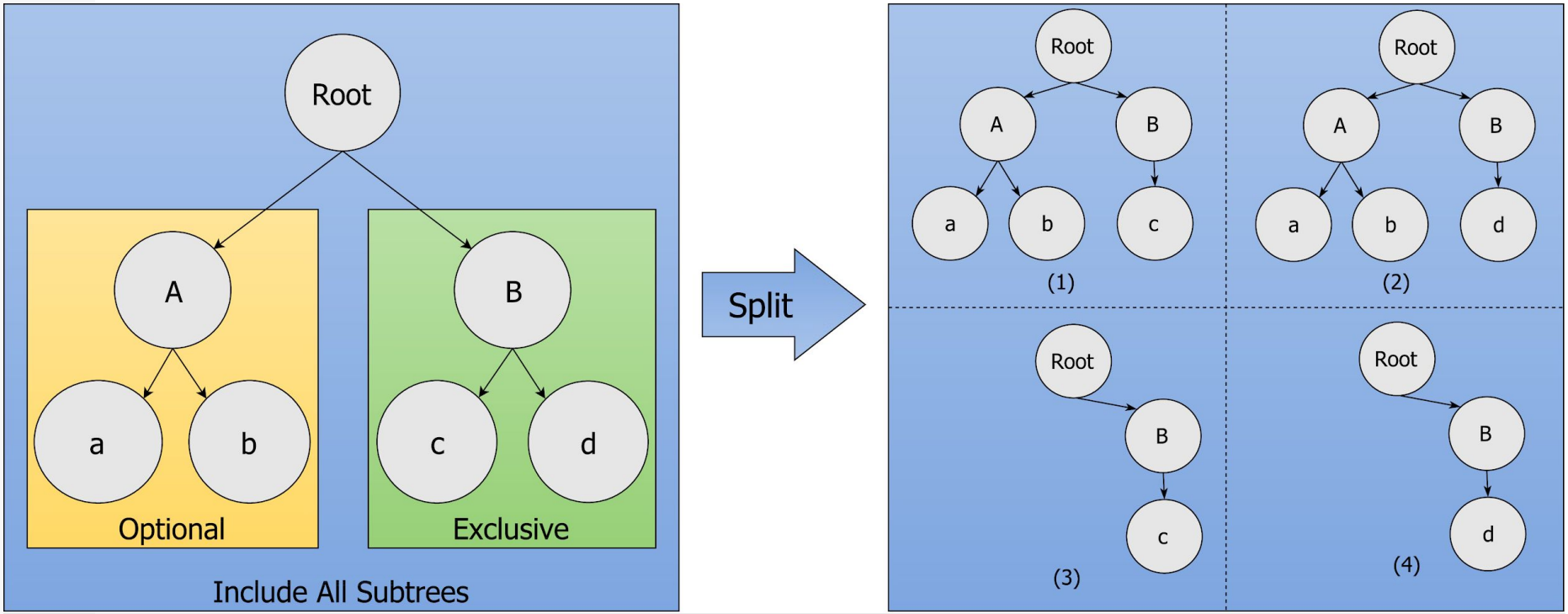


CNN Architecture Search Domains



SVM Kernel Hyperparameter Domains

Properties of Search Spaces



UNIVERSITY OF NOTRE DAME

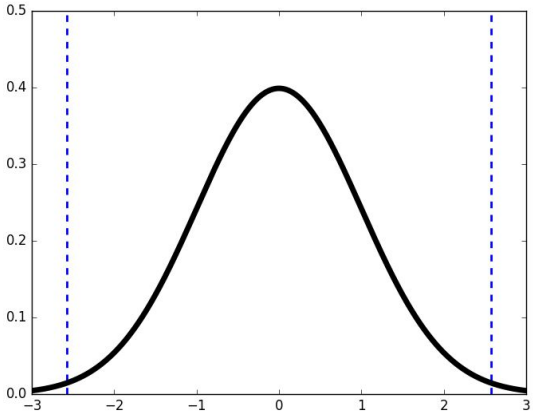
Kinnison, Jeffery, Nathaniel Kremer-Herman, Douglas Thain, and Walter Scheirer. "Shadho: Massively scalable hardware-aware distributed hyperparameter optimization." In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 738-747. IEEE, 2018.

Heuristics for Scheduling: Complexity

Combinatorial size of hyperparameter search domains in a model.

Discrete Domain: ['a', 'b', 'c', 'd'] => 4

Continuous Domain:



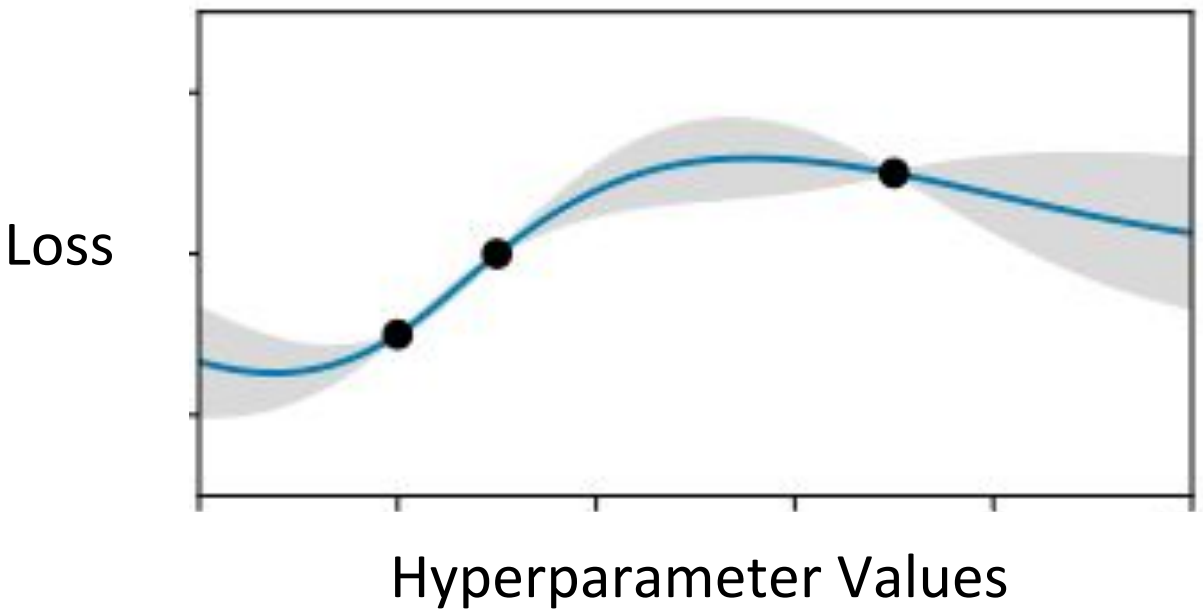
=> 5.15

UNIVERSITY OF NOTRE DAME

Kinnison, Jeffery, Nathaniel Kremer-Herman, Douglas Thain, and Walter Scheirer. "Shadho: Massively scalable hardware-aware distributed hyperparameter optimization." In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 738-747. IEEE, 2018.

Heuristics for Scheduling: Uncertainty

Approximation of confidence in model performance across different hyperparameterizations.



Kinnison, Jeffery, Nathaniel Kremer-Herman, Douglas Thain, and Walter Scheirer. "Shadho: Massively scalable hardware-aware distributed hyperparameter optimization." In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 738-747. IEEE, 2018.

Compute Classes

Observation: Different hardware offers different running time performance.

Strategy: Group compute nodes with common hardware together for predictable performance.

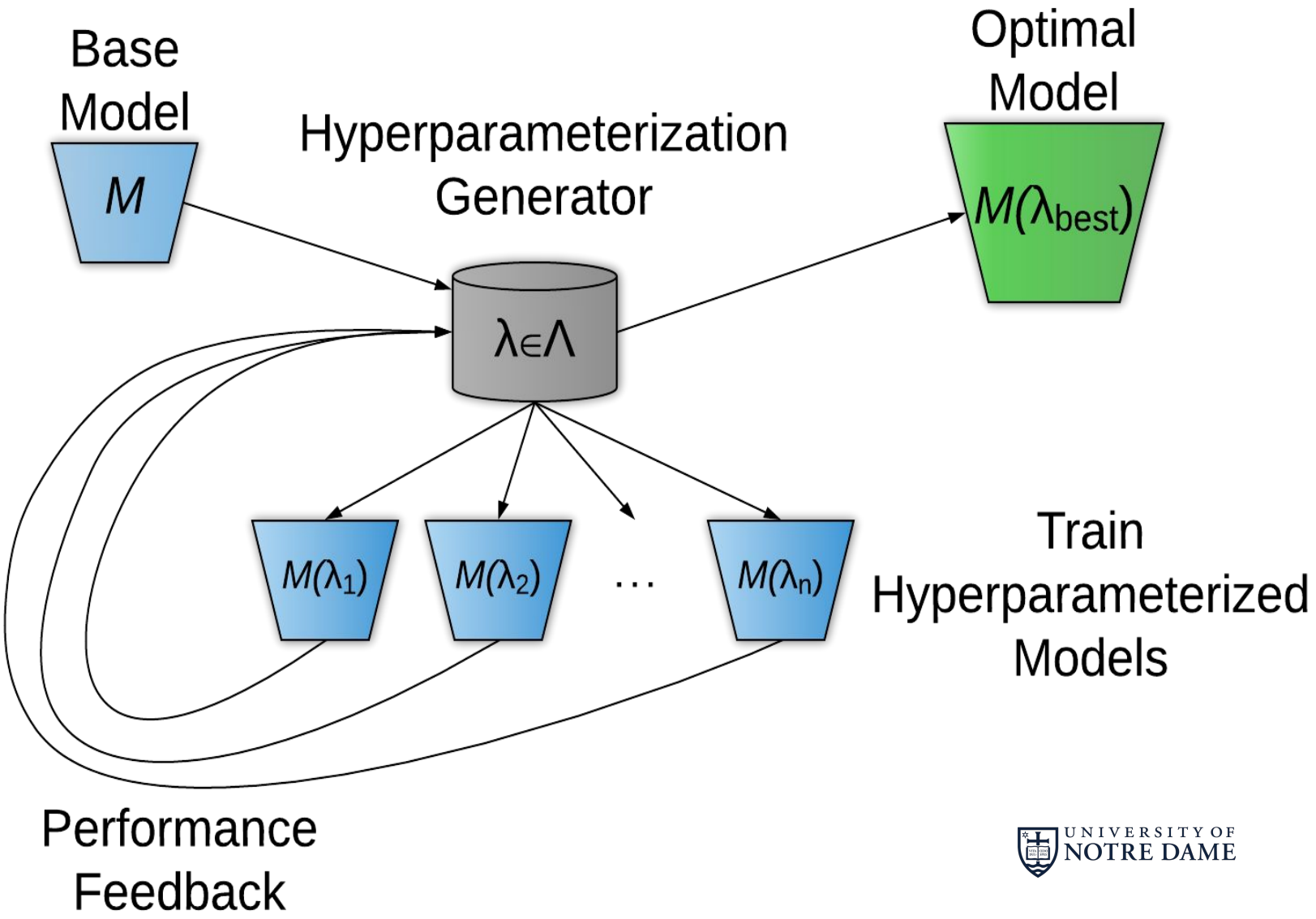
Kinnison, Jeffery, Nathaniel Kremer-Herman, Douglas Thain, and Walter Scheirer. "Shadho: Massively scalable hardware-aware distributed hyperparameter optimization." In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 738-747. IEEE, 2018.

SHADHO Optimzation

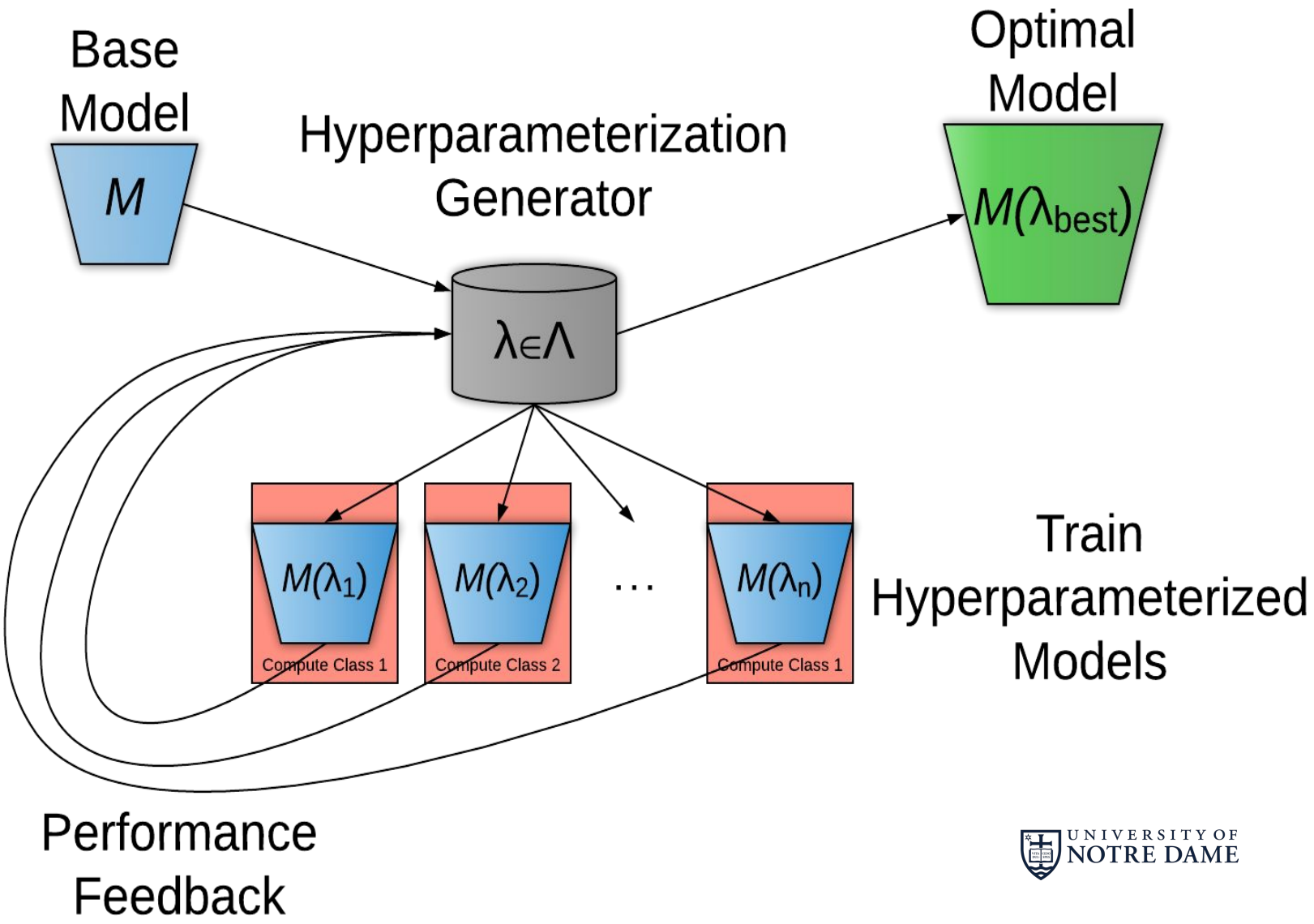
1. Organize hyperparameter searches into a forest of non-overlapping search spaces
2. Order trees in the forest by their structure (search **complexity**) and performance variability (search **uncertainty**)
3. Assign trees to **compute classes**, with high-complexity, high-priority trees assigned to higher-performing hardware
4. Evaluate parameterized models on assigned hardware
5. Update priority and repeat

Kinnison, Jeffery, Nathaniel Kremer-Herman, Douglas Thain, and Walter Scheirer. "Shadho: Massively scalable hardware-aware distributed hyperparameter optimization." In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 738-747. IEEE, 2018.

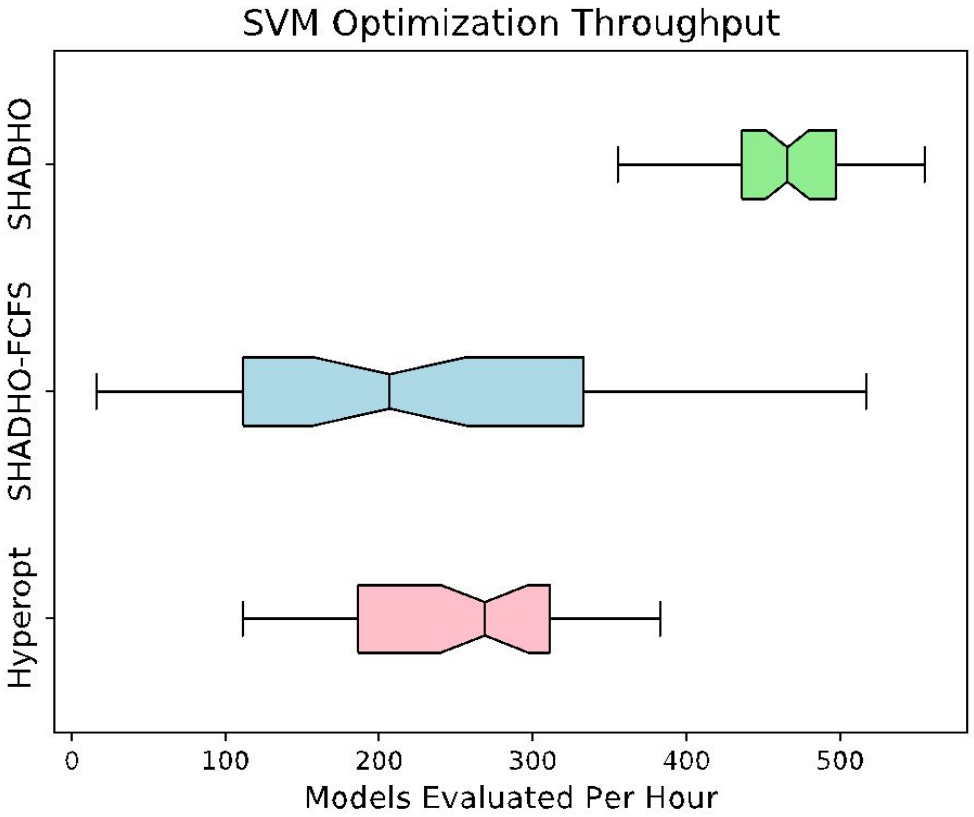
Distributed Hyperparamter Optimization



SHADHO Optimization



SVM Kernel Optimization for MNIST



On average over 48 trials, SHADHO increased search throughput by 1.8x.

UNIVERSITY OF NOTRE DAME

Kinnison, Jeffery, Nathaniel Kremer-Herman, Douglas Thain, and Walter Scheirer. "Shadho: Massively scalable hardware-aware distributed hyperparameter optimization." In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 738-747. IEEE, 2018.

Defining a Search Space

Continuous Domains

Draw values from a continuous probability distribution.

Discrete/Categorical Domains

Randomly sample from a set of values.

Grid Domains

Exhaustively search over a set of values.

Dependent Domains

Compute a value in response to another hyperparameter.

Kinnison, Jeffery, Nathaniel Kremer-Herman, Douglas Thain, and Walter Scheirer. "Shadho: Massively scalable hardware-aware distributed hyperparameter optimization." In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 738-747. IEEE, 2018.

Defining a Search Space

Hierarchical Domains

Organize domains hierarchically into trees.

Repeating Domains

Create a repeating structure and search over subsets.

Sequential Domains

Set up ordered sequences of domains.

Kinnison, Jeffery, Nathaniel Kremer-Herman, Douglas Thain, and Walter Scheirer. "Shadho: Massively scalable hardware-aware distributed hyperparameter optimization." In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 738-747. IEEE, 2018.

Available Sampling Methods

Grid Search

Exhaustively sample over a regular grid.

Random Search

Randomly sample with no guidance.

Bayesian Optimization

- Gaussian Processes
- Random Forests
- TPE

Kinnison, Jeffery, Nathaniel Kremer-Herman, Douglas Thain, and Walter Scheirer. "Shadho: Massively scalable hardware-aware distributed hyperparameter optimization." In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 738-747. IEEE, 2018.

Getting SHADHO

https://github.com/jeffkinnison/shadho

```
pip3 install shadho

python3 -m shadho.install.workqueue
```

UNIVERSITY OF NOTRE DAME

Kinnison, Jeffery, Nathaniel Kremer-Herman, Douglas Thain, and Walter Scheirer. "Shadho: Massively scalable hardware-aware distributed hyperparameter optimization." In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 738-747. IEEE, 2018.

2. A First Example

https://jeffkinnison.github.io/shadho-tutorial/

UNIVERSITY OF NOTRE DAME

Optimizing a Convex Function

Example 1: *sin(x) * cos(y)*

In this example, we will:

- Create continuous and categorical search domains
- Set up a SHADHO driver
- Run the search locally

Optimizing a Convex Function

Example 1: Creating the search space

```
from shadho import Shadho, spaces

search_space = {
        'x': spaces.uniform(0, 2 * math.pi),
        'y': spaces.choice(list(np.linspace(0, 2 * math.pi, 1000)))
    }
```

Search Domains

(Continuous) x ~ U(0, 2π)

(Categorical) y ~ grid of 1000 points in [0, 2π]

Optimizing a Convex Function

Example 1: Setting up the objective

```
def objective(params):
    x = params['x']
    y = params['y']

    return np.sin(x) * np.cos(y)
```

Receives a dictionary of parameters

Returns a floating-point result

Optimizing a Convex Function

Example 1: Setting up the driver

```
opt = Shadho(
        'convex-tutorial',  # Name of this experiment
        objective,          # The function to optimize
        search_space,       # The search space to sample
        method='random',    # The sampling method
        timeout=30          # The time to run the search (s).
    )
```

Pass the objective function to the driver with the search space.

Optimizing a Convex Function

Example 1: Setting up the driver

```
opt = Shadho(
        'convex-tutorial',  # Name of this experiment
        objective,          # The function to optimize
        search_space,       # The search space to sample
        method='random',    # The sampling method
        timeout=30          # The time to run the search (s).
    )

opt.run()
```

Optimizing a Convex Function

Example 1: Running the search

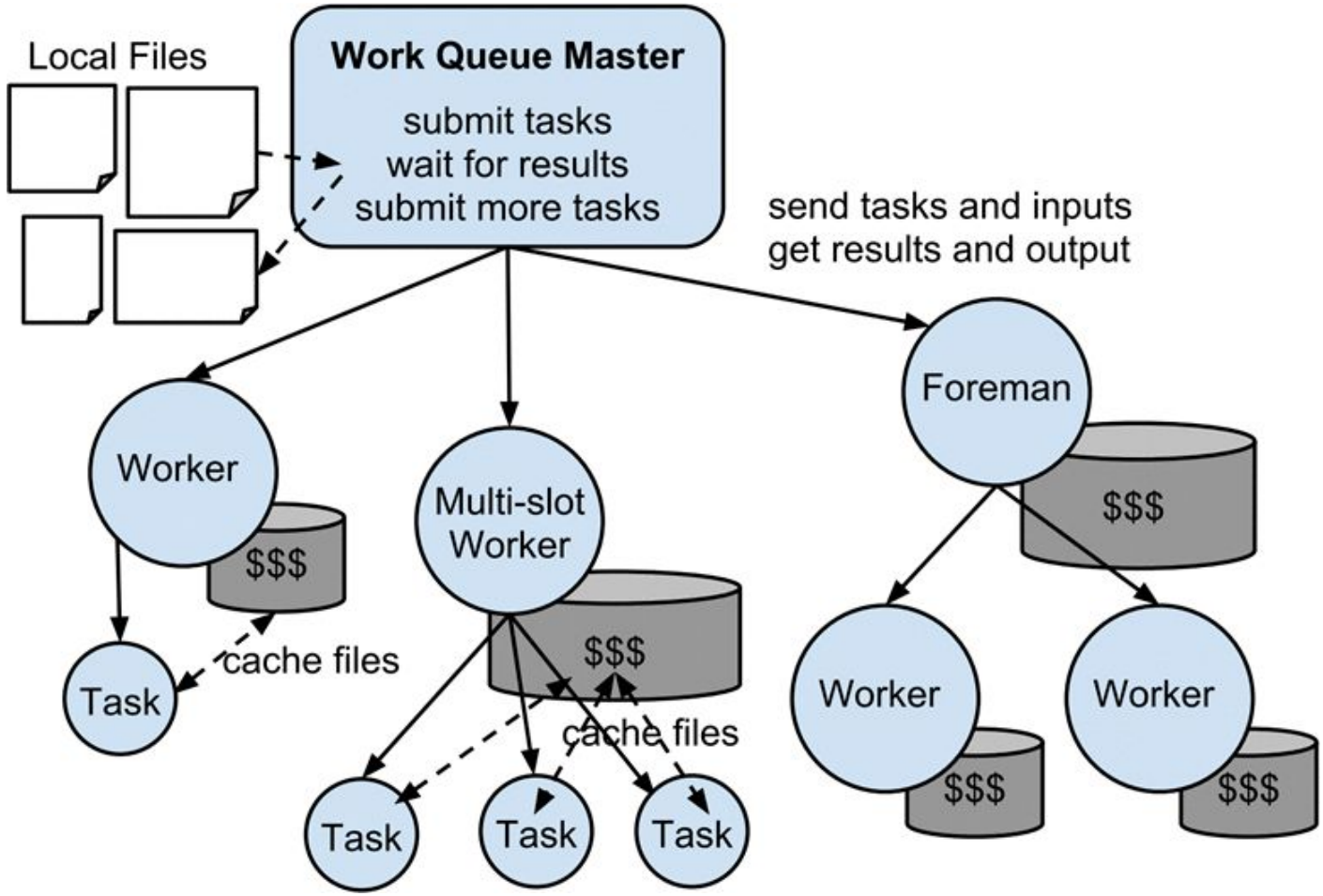
```
opt = Shadho(
        'convex-tutorial',  # Name of this experiment
        objective,          # The function to optimize
        search_space,       # The search space to sample
        method='random',    # The sampling method
        timeout=30          # The time to run the search (s).
    )

opt.run()
```

3. Distributed Search with SHADHO

https://jeffkinnison.github.io/shadho-tutorial/

UNIVERSITY OF NOTRE DAME

Distributing Trials with Work Queue



UNIVERSITY OF NOTRE DAME

Albrecht, Michael, Dinesh Rajan, and Douglas Thain. "Making work queue cluster-friendly for data intensive scientific applications." In *2013 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 1-8. IEEE, 2013.

Distributing Trials with Work Queue

- Master/worker model -> workers ping master for tasks
- Master exists as a public-facing server
- Workers can connect from any domain
- Workers advertise hardware capabilities of host machine
- Workers operate in a sandbox, cache persistent files
- Communicates in files sent to worker/returned to master

Distributing Trials

Example 2: *sin(x) * cos(y)*

In this example, we will:

- Convert the previous example to distributed
- Demonstrate running a SHADHO worker

Optimizing a Convex Function

Example 2: Creating the search space

```
from shadho import Shadho, spaces

search_space = {
        'x': spaces.uniform(0, 2 * math.pi),
        'y': spaces.choice(list(np.linspace(0, 2 * math.pi, 1000)))
    }
```

Search Domains

(Continuous) x ~ U(0, 2π)

(Categorical) y ~ grid of 1000 points in [0, 2π]

Optimizing a Convex Function

Example 2: Setting up the driver

```
opt = Shadho(
        'distributed-tutorial',  # Name of this experiment
        'bash evaluate.sh',      # The function to optimize
        search_space,            # The search space to sample
        method='random',         # The sampling method
        timeout=30               # The time to run the search (s)
    )
```

Pass a command to be run on each worker to the driver with the search space.

Optimizing a Convex Function

Example 2: Setting up the objective (*objective.py*)

```python
import math

def objective(params):
    x = params['x']
    y = params['y']

    return math.sin(x) * math.cos(y)

if __name__ == '__main__':
    import shadho_worker
    shadho_worker.run(objective)
```

The objective function is moved to its own file.

Wrap the objective with shadho_worker, which handles parsing hyperparameters.

Optimizing a Convex Function

Example 2: Setting up the objective (*evaluate.sh*)

```
#!/usr/bin/env bash

# Load any environment to run the objective.

python3 objective.py
```

The worker will run this file. Use it like any other shell script, job file, etc.

Optimizing a Convex Function

Example 2: Adding *evaluate.sh* and *objective.py*

```
opt.add_input_file('evaluate.sh')
opt.add_input_file('objective.py')
```

Give SHADHO the path to the input files.

These files will be cached on every worker between trials.

Optimizing a Convex Function

Example 2: Running the driver and worker

Driver - In one terminal:

```
python3 driver.py
```

Worker - In another terminal:

```
python3 -m shadho.workers.workqueue \
    -M distributed-tutorial  # experiment key
```

6. Optimizing SVM

https://jeffkinnison.github.io/shadho-tutorial/

Optimizing SVM Kernels

Example 2: *SVM for MNIST Classification*

In this example, we will:

- Create hierarchical search spaces
- Work with compute classes
- Return additional results

Optimizing SVM Kernels

Example 2: Setting up the search space

```
C = spaces.uniform(-1000, 2000)
gamma = spaces.log10_uniform(-5, 8)
coef0 = spaces.uniform(-1000, 2000)
degree = [2, 3, 4, 5, 6, 7]
```

Search domains can be stored and used multiple times.

Optimizing SVM Kernels

Example 2: Setting up the search space

```
search_space = {
    'linear': {
        'C': C,
    },
    'rbf': {
        'C': C,
        'gamma': gamma,
    },
    'sigmoid': {
        'C': C,
        'gamma': gamma,
        'coef0': coef0,
    },
    'poly': {
        'C': C,
        'gamma': gamma,
        'coef0': coef0,
        'degree': degree
    },
    'exclusive': True
}
```

Each kernel is stored independently.

The “exclusive” flag tells SHADHO to sample each independently.

Optimizing SVM Kernels

Example 2: Setting up the driver

```
opt = Shadho(
        'svm-tutorial',      # The experiment key
        'bash evaluate.sh',  # The command to run on the worker
        search_space,        # The search space
        method='random',     # The sampling method to use
        timeout=120          # The amount of time to run (s)
)
```

Optimizing SVM Kernels

Example 2: Setting up the objective

```
from sklearn.svm import svc

def main(params):
    kernel = list(params.keys())[0]
    kernel_params = params[kernel]

    ...

    svc = SVC(kernel=kernel, **kernel_params)
```

Optimizing SVM Kernels

Example 2: Setting up the objective

```
def main(params):

    ...

    out = {
        'loss': loss,
        'accuracy': accuracy,
        'precision': precision,
        'recall': recall,
        'train_time': train_time,
        'test_time': test_time
    }

    return out

if __name__ == '__main__':
    import shadho_worker
    shadho_worker.run(main)
```

Return multiple values in a dictionary and SHADHO will record them all.

“loss” is considered the objecctive.

Optimizing a Convex Function

Example 2: Adding *evaluate.sh* and *train_svm.py*

```
opt.add_input_file('evaluate.sh')
opt.add_input_file('train_svm.py')
opt.add_input_file('mnist.npz')
```

In addition to code, we can send small datasets to be cached on workers.

Optimizing SVM Kernels

Example 2: Setting up the compute classes

```
opt.add_compute_class('16-core', 'cores', 16, max_tasks=20)
opt.add_compute_class('8-core', 'cores', 8, max_tasks=25)
opt.add_compute_class('4-core', 'cores', 4, max_tasks=50)
```

Compute classes are given a name, a resource grouping them, the amount of that resource, and the maximum expected number of workers.

Optimizing a SVM Kernels

Example 2: Running the driver and worker

Driver - In one terminal:

```
python3 driver.py
```

Worker - In another terminal:

```
python3 -m shadho.workers.workqueue \
    -M svm-tutorial \
    --cores 4
```

5. Optimizing Neural Network Structure

https://jeffkinnison.github.io/shadho-tutorial/

Optimizing Neural Network Structure

Example 2: *CNN for CIFAR-10 Classification*

In this example, we will:

- Demonstrate the object-oriented spaces API
- Use repeating and conditional domains
- Set up compute classes for GPU models

Optimizing Neural Network Structure

Example 4: *CNN for CIFAR-10 Classification*

Libraries Used:

- shadho
- pytorch
- torchvision (data & augmentation)
- ignite (training loop)

Optimizing Neural Network Structure

Example 2: Setting up the search space

```
activations = ['glu', 'leaky_relu', 'prelu', 'relu', 'selu', 'sigmoid', 'tanh']
batch_norm = spaces.log10_uniform(-4, 4)

conv_layer = spaces.scope(
    out_filters=spaces.log2_randint(4, 10),
    kernel_shape=spaces.randint(1, 10, step=2),
    activation=activations,
    batch_norm=batch_norm
)
```

Optimizing Neural Network Structure

Example 2: Setting up the search space

```
conv_layer.padding = spaces.dependent(
    conv_layer.kernel_shape,
    callback=lambda x: int(x // 2))
```

Dependent domains can be set up to react to other hyperparameter values.

This space implements “same” padding in response to generated kernel shapes

Optimizing Neural Network Structure

Example 2: Setting up the search space

```
conv_layers = spaces.repeat(conv_layer, 6)
```

```
dense_layer = spaces.scope(
    out_features=spaces.log2_randint(7, 13),
    activation=activations,
    batch_norm=batch_norm,
)

dense_layers = spaces.repeat(dense_layer, 3)
```

Domains may also be repeated *n* times to easily set up repeating structures.

Optimizing Neural Network Structure

Example 2: Building the network

```
layer_input = in_channels
for i, layer in enumerate(hyperparameters['conv_layers']):
    layers.append(nn.Conv2d(
        layer_input,
        layer['out_features'],
        layer['kernel_shape'],
        padding=layer['padding']))
    layers.append(activations[layer['activation']]())
    layers.append(nn.BatchNorm2d(layer['out_features']))
    layer_input = layer['out_features']

    if i % 2 == 1:
        layers.append(nn.MaxPool2d(2))
        data_shape = [int(d // 2) for d in data_shape]
```

Optimizing Neural Network Structure

Example 4: Building the network

```
layer_input = layer_input * np.product(data_shape)
layers.append(nn.Flatten())
for layer in hyperparameters['dense_layers']:
    layers.append(nn.Linear(layer_input, layer['out_features']))
    layers.append(activations[layer['activation']]())
    layer_input = layer['out_features']
```

6. Going Forward

https://jeffkinnison.github.io/shadho-tutorial/

Adding to the Software

- More sampling methods (e.g., population-based)
- Additional hyperparameter domains
- Automatic compute class detection

Updating the Process

- Explore uncertainty quantification and hyperparameter importance methods as heuristics
- Develop new search strategies
- Extending beyond hyperparameters

THANKS!

Any questions?