Better and Faster Hyperparameter Optimization with Dask-ML

Scott Sievert
@stsievert

Tom Augspurger
@TomAugspurger

Matthew Rocklin
@mrocklin

https://github.com/stsievert/talks

- What problem do I want to solve?
- What tool do I want to use?
- What new opportunities can the tool Dask enable?
- How should the chosen algorithm be used, and how does it perform?

https://github.com/stsievert/talks

What is a hyperparameter?

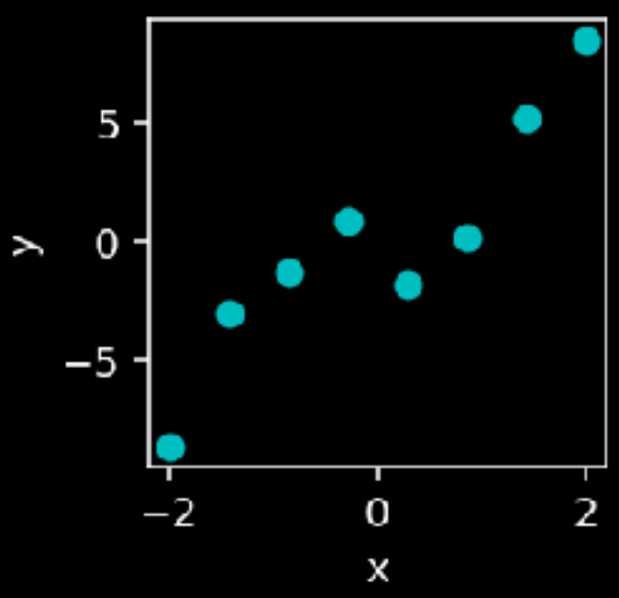
A free parameter not learned from data.

Typically used to define model structure.

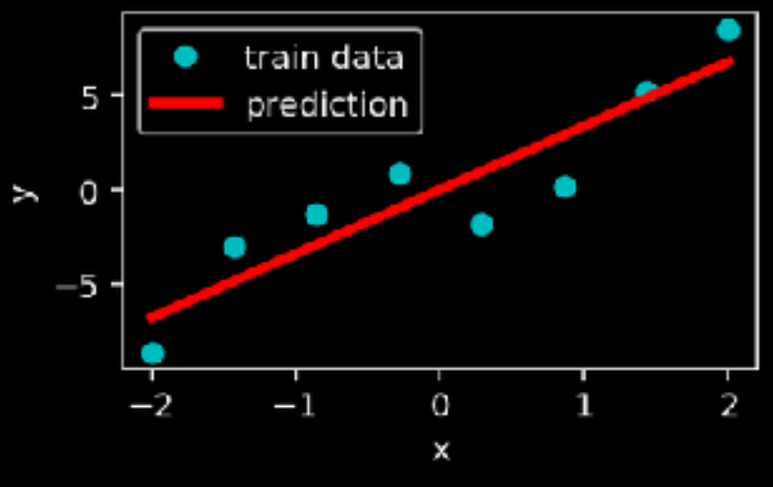
Model: polynomials of degree d

```
y = wd*x**d + wd1*x**(d-1) + … + w1*x
```

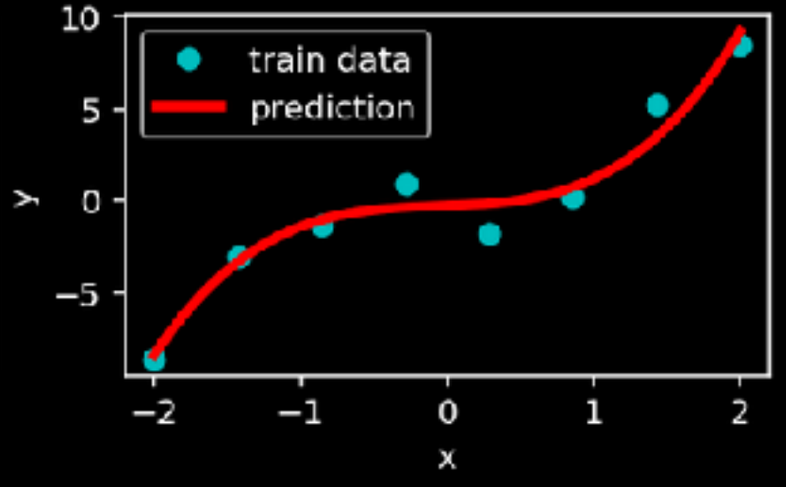
Train data



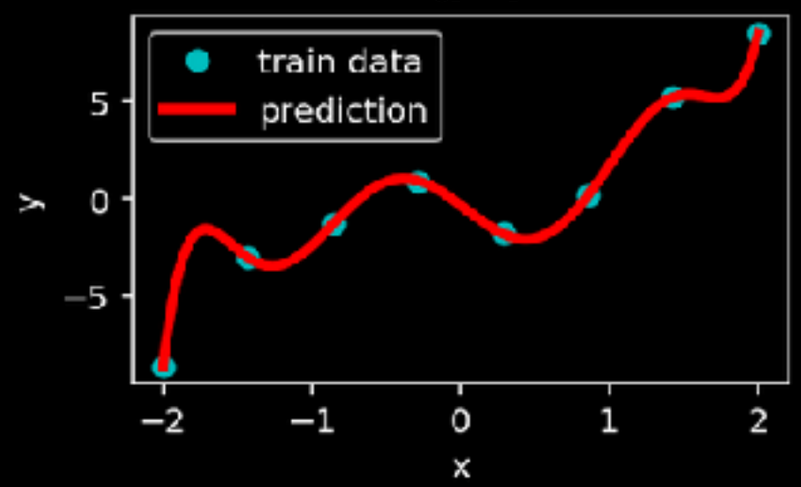
d=1



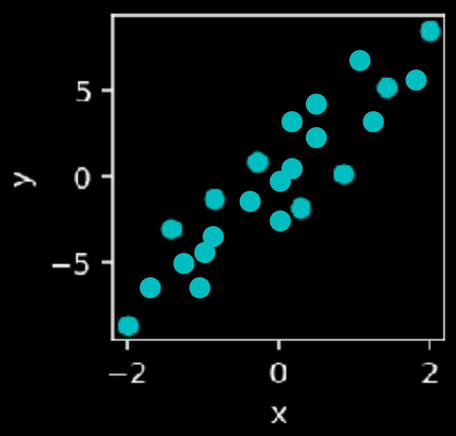
d=3



d=7



Unseen validation data



https://github.com/stsievert/talks

Example

Distill

How to Use t-SNE Effectively

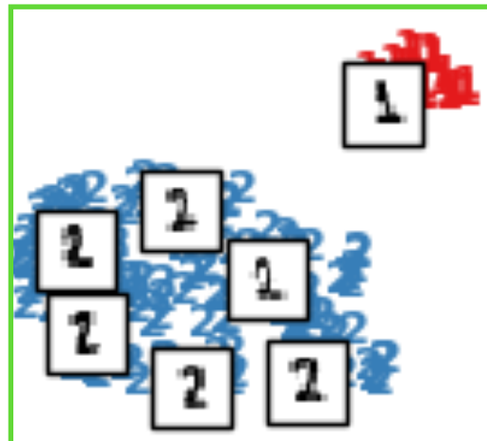
MARTIN WATTENBERG
Google Brain

FERNANDA VIÉGAS
Google Brain

IAN JOHNSON
Google Cloud

Oct. 13
2016

Citation:
Wattenberg, et al., 2016



perplexity
early_exaggeration
metric
learning_rate
n_iter
init

1. Those hyperparameters really matter

Let's start with the "hello world" of t-SNE: a data set of two widely separated clusters. To make things as simple as possible, we'll consi clusters in a 2D plane, as shown in the lefthand diagram. (For clarity,

Original

Scikit-learn docs recommend perplexity values between 5–50

https://distill.pub/2016/misread-tsne/

What's hyperparameter optimization?

Finding the *best* set of hyperparameters

Unfortunately, hyperparameter optimization will require a lot of computation.*

* Bergstra, Bardenet, Bengio, & Kégl. (2011). Algorithms for hyper-parameter optimization.

https://github.com/stsievert/talks

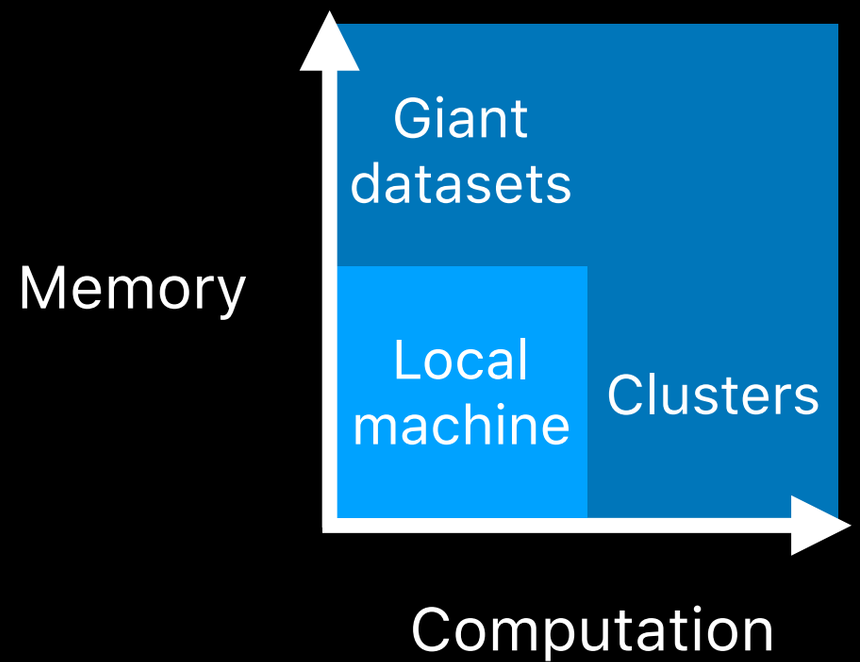
What tool do I want to use to manage computation?

https://github.com/stsievert/talks

Dask

Dask natively scales Python

Dask provides advanced parallelism for analytics,
enabling performance at scale for the tools you love



What makes Dask different?

Easy to use
Declarative
Diagnostics

https://github.com/stsievert/talks

Dask is easy to use

Dask mirrors the NumPy/Pandas/Scikit-learn APIs.

In [1]:
```
# Read one CSV with Panadas
import pandas import pd
df = pd.read_csv("data/snow-2020-01.csv")

idx = df["snow_fall"] > 0
df.loc[idx, "snow_fall"].mean()
```

Out [1]: 3.2

In [2]:
```
# Read 100's of CSVs with Dask
import dask.dataframe import dd
df = dd.read_csv("data/snow-*.csv")

idx = df["snow_fall"] > 0
df.loc[idx, "snow_fall"].mean().compute()
```

Out [2]: 2.4

https://github.com/stsievert/talks

Dask is declarative

Dask works on almost all computational systems.
If you can use these systems, you can use Dask.

- Single machine (i.e., your laptop)
- Manual config (IP addresses)
- Kubernetes
- Hadoop (/YARN)
- PBS, Slurm, MOAB, SGE, LSF, and HTCondor.

https://distributed.dask.org/
https://kubernetes.dask.org/
https://yarn.dask.org/
https://jobqueue.dask.org
https://cloudprovider.dask.org/

Dask has great diagnostics

Easy to visualize what your cluster is doing

https://docs.dask.org/en/latest/diagnostics-distributed.html

What algorithm is now implemented in Dask-ML?

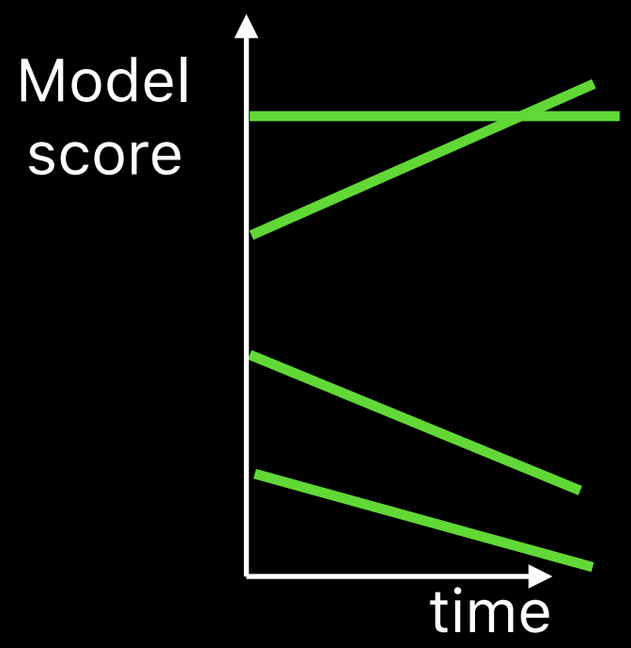
First, what algorithm is widely used?

https://github.com/stsievert/talks

Popular algorithm for hyperparameter optimization

Scikit-learn's **`RandomizedSearchCV`**:

1. Randomly pick hyperparameters
2. Create models with those hyperparameters
3. Train model to completion
4. Report validation score

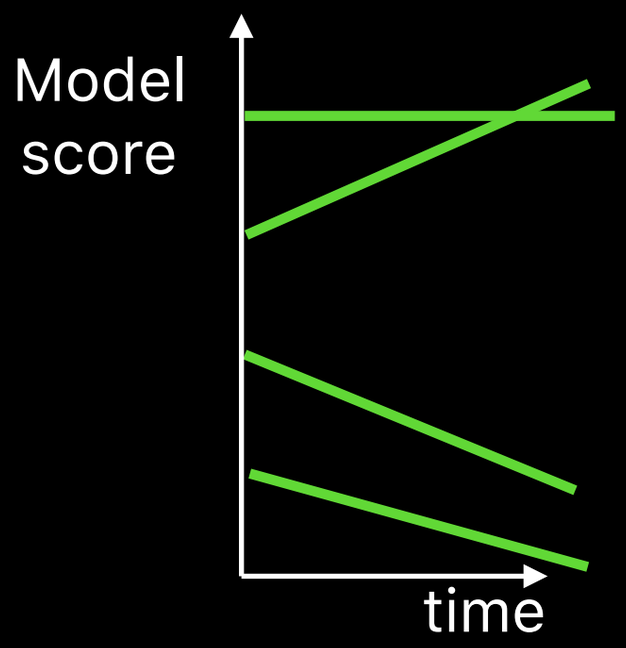


RandomizedSearchCV

✓ Simple implementation

✓ Easy to parallelize

✗ Does not limit computation



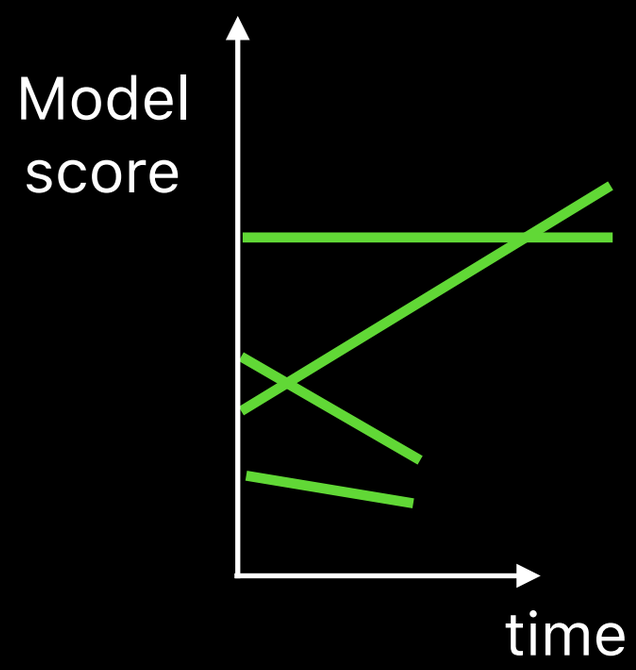
https://github.com/stsievert/talks

`RandomizedSearchCV` has nice features,
but can have excessive computation

How can the computation be limited?

Early stopping of low performing models

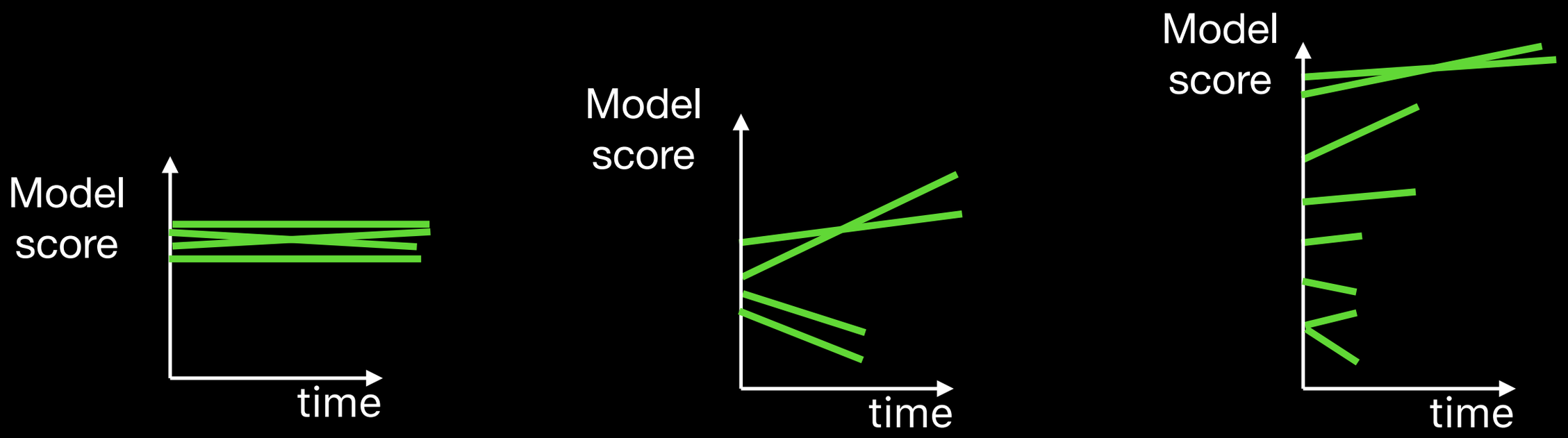
With
early stopping



https://github.com/stsievert/talks

Hyperband

Hyperband is a principled early stopping scheme
for random hyperparameter selection.



Training time most important

Hyperparameter exploration most important

Sweeps over relative importance of training time

[PDF] **Hyperband**: A novel bandit-based approach to hyperparameter optimization [PDF] jmlr.org

L Li, K Jamieson, G DeSalvo, A Rostamizadeh… - arXiv preprint arXiv …, 2016 - jmlr.org

Performance of machine learning algorithms depends critically on identifying a good set of hyperparameters. While recent approaches use Bayesian optimization to adaptively select configurations, we focus on speeding up random search through adaptive resource …

☆ Cited by 224 Related articles All 14 versions »

https://github.com/stsievert/talks

Hyperband

Principled early stopping scheme
for random hyperparameter selection.

Hyperband will* return high performing models
with minimal training:

Number of `partial_fit` calls

Corollary 1. *(informal presentation of [LJD+18, Theorem 5] and surrounding discussion) Assume the loss at iteration k decays like* $(1/k)^{1/\alpha}$*, and the validation losses* ν *approximately follow the cumulative distribution function* $F(\nu) = (\nu - \nu_*)^{\beta}$ *with optimal validation loss* ν_* *with* $\nu - \nu_* \in [0,1]$ *.*

Then for any $T \in \mathbb{N}$*, let* $\widehat{i}_T$ *be the empirically best performing model when models are stopped early according to the infinite horizon Hyperband algorithm when* T *resources have been used to train models. Then with probability* $1-\delta$*, the empirically best performing model* $\widehat{i}_T$ *has loss*

$$\nu_{\widehat{i}_T} \leq \nu_* + c\left(\frac{\overline{\log}(T)^3 \cdot a}{T}\right)^{1/\max(\alpha,\,\beta)}$$

for some constant c *and* $a = \overline{\log}(\log(T)/\delta)$ *where* $\overline{\log}(x) = \log(x\log(x))$.

By comparison, finding the best model without the early stopping Hyperband performs (i.e., randomized searches and training until completion) after T *resources have been used to train models has loss*

$$\nu_{\widehat{i}_T} \leq \nu_* + c\left(\frac{\log(T) \cdot a}{T}\right)^{1/(\alpha+\beta)}$$

Close to the lower bound on the “number of resources” required

[PDF] **Hyperband**: A novel bandit-based approach to hyperparameter optimization [PDF] jmlr.org

L Li, K Jamieson, G DeSalvo, A Rostamizadeh… - arXiv preprint arXiv …, 2016 - jmlr.org

Performance of machine learning algorithms depends critically on identifying a good set of hyperparameters. While recent approaches use Bayesian optimization to adaptively select configurations, we focus on speeding up random search through adaptive resource …

☆ 99 Cited by 224 Related articles All 14 versions »

*with high probability

https://github.com/stsievert/talks

Hyperband

✓ Simple implementation

✓ Easy to parallelize

✓ Effectively limits computation for complicated search spaces

✓ Mathematical justification

https://github.com/stsievert/talks


```
from dask_ml.model_selection import HyperbandSearchCV
```

This code is available Dask-ML,
Dask's machine learning library.

Dask-ML documentation: https://ml.dask.org/
Installation: https://ml.dask.org/install.html
Hyperparameter optimization docs:
https://ml.dask.org/hyper-parameter-search.html

https://github.com/stsievert/talks

Hyperband in Dask-ML

Dask enables better performance.

How is Hyperband used?

This is the first Hyperband implementation
with an advanced job scheduler*

* to my knowledge

https://github.com/stsievert/talks

Examples

Takeaways:

1. Hyperband outperforms random search

2. `HyperbandSearchCV` works with many datatypes
   (NumPy, CuPy & Dask arrays, Pandas & Dask dataframes, PyTorch Tensors, etc).

3. `HyperbandSearchCV` works with

...and almost any model with a Scikit-learn API

4. Dask (greatly) aids the Hyperband implantation

https://github.com/stsievert/talks

Dask-ML implementation:

Example 1

Laptop w/ 4 cores

Scikit-learn model

https://github.com/stsievert/talks

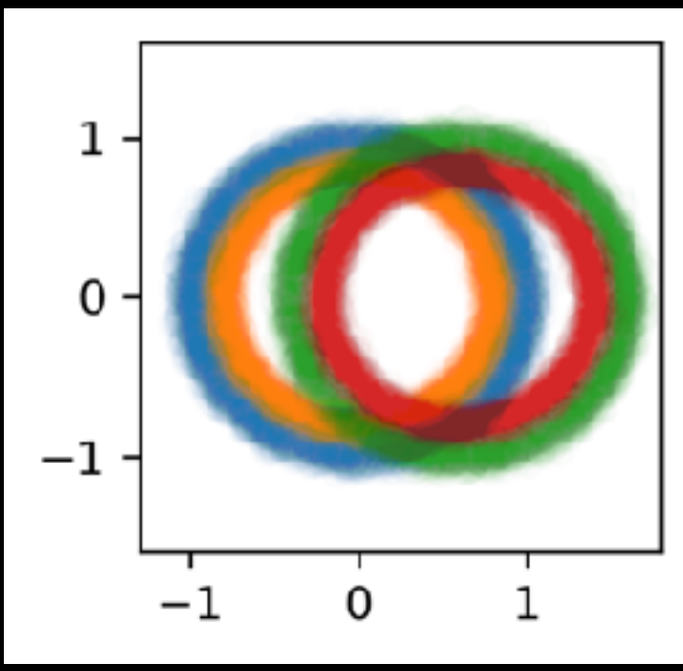
Synthetic simulation

Use case: initial exploration on data scientist's personal laptop.

Model
Scikit-learn's neural network MLPClassifier

```
from sklearn.neural_network import MLPClassifier

model = MLPClassifier(solver="sgd", …)
```

Dataset



Search space

Discrete hyperparameters: 50 unique choices

4 continuous hyperparameters

```
params = {
    "batch_size": …,  # 5 choices
    "learning_rate": …,  # 2 choices
    "hidden_layer_sizes": …,  # 5 choices
    "alpha": …,  # continuous
    "power_t": …,  # continuous
    "momentum": …,  # continuous
    "learning_rate_init": …,  # continuous
}
```

https://github.com/stsievert/dask-hyperband-comparison

https://github.com/stsievert/talks

HyperbandSearchCV usage

```
n_examples = 50 * len(X_train)
n_params = 299
```

```
max_iter = n_params
chunk_size = n_examples // n_params
```

```
search = HyperbandSearchCV(
    model, params,
    max_iter=max_iter,
    aggressiveness=4,
)

X_train, y_train = rechunk(X_train, y_train, chunks=chunk_size)
```

```
search.metadata
search.fit(X_train, y_train)
```

https://ml.dask.org/modules/generated/dask_ml.model_selection.HyperbandSearchCV.html

https://github.com/stsievert/talks

File Edit View Run Kernel Tabs Settings Help

Example-Scikit-learn-Copy

Code Python 3

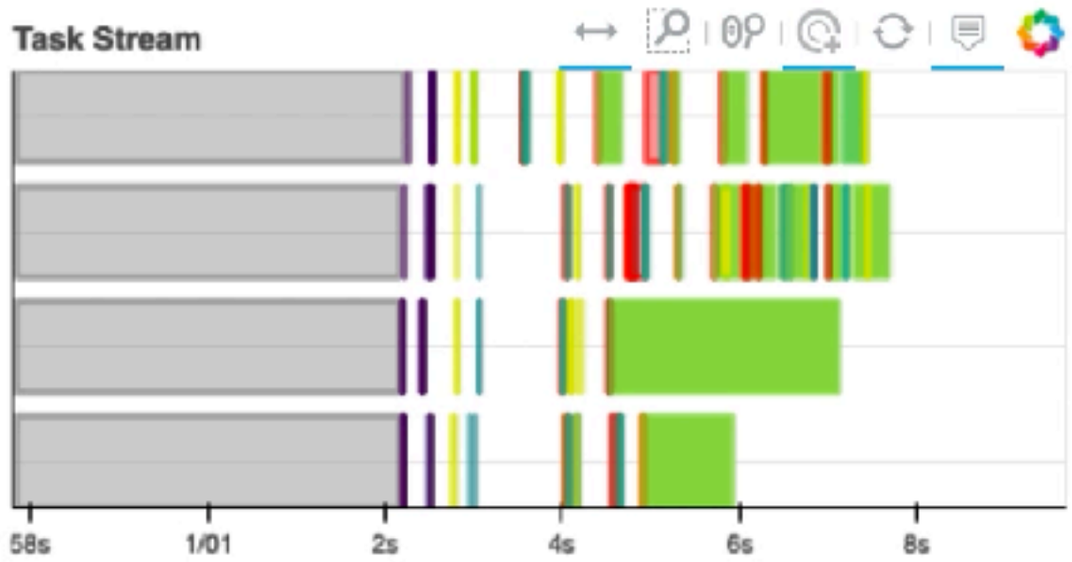
```
[22]: from dask_ml.model_selection import HyperbandSearc

search = HyperbandSearchCV(
    model,
    params,
    max_iter=max_iter,
    aggressiveness=4,
    verbose=True,
    random_state=42,
)
```

```
[*]: %%time
search.fit(
    X_train,
    y_train,
    classes=[0, 1, 2, 3]
)
```

```
[CV, bracket=2] train, test examples = 39999, 1000
1
[CV, bracket=2] creating 16 models
[CV, bracket=1] train, test examples = 39999, 1000
1
[CV, bracket=1] creating 6 models
[CV, bracket=0] train, test examples = 39999, 1000
1
[CV, bracket=0] creating 3 models
```

```
[ ]:
```

Dask Task Stream



Dask Progress

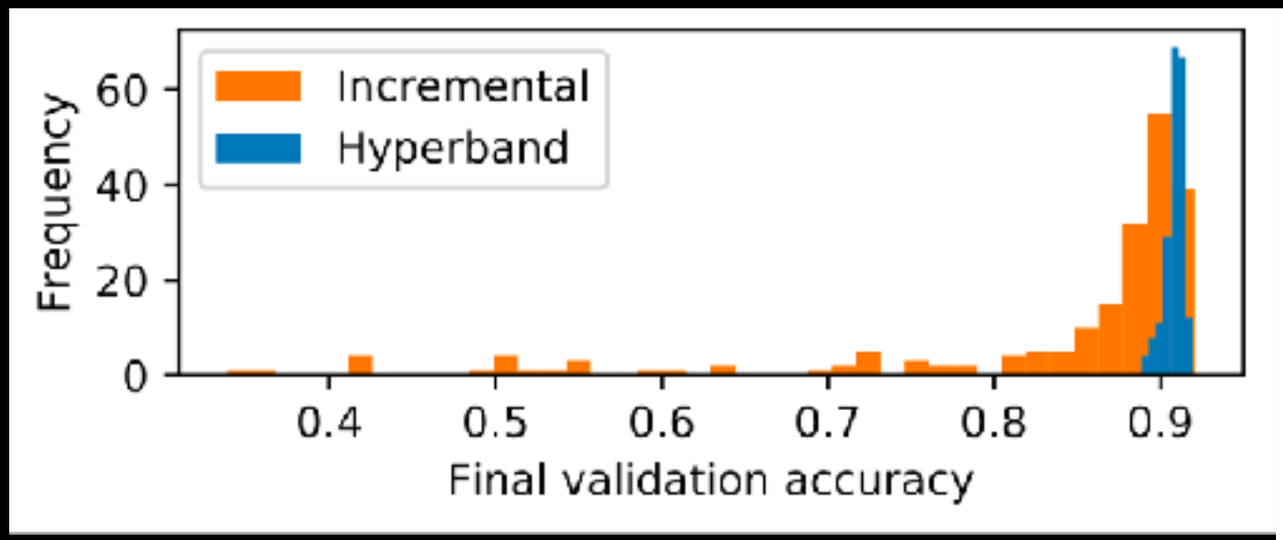
Progress -- total: 222, in-memory: 63, processing: 11, waiting: 64, erred: 0

_partial_fit	10 / 50	finalize	4 / 6
from-value-g...	32 / 36	array-original	2 / 2
concatenate	32 / 36		
_create_model	25 / 25		
_score	5 / 25		
add-from-val...	18 / 18		
sub	16 / 18		
array	6 / 6		

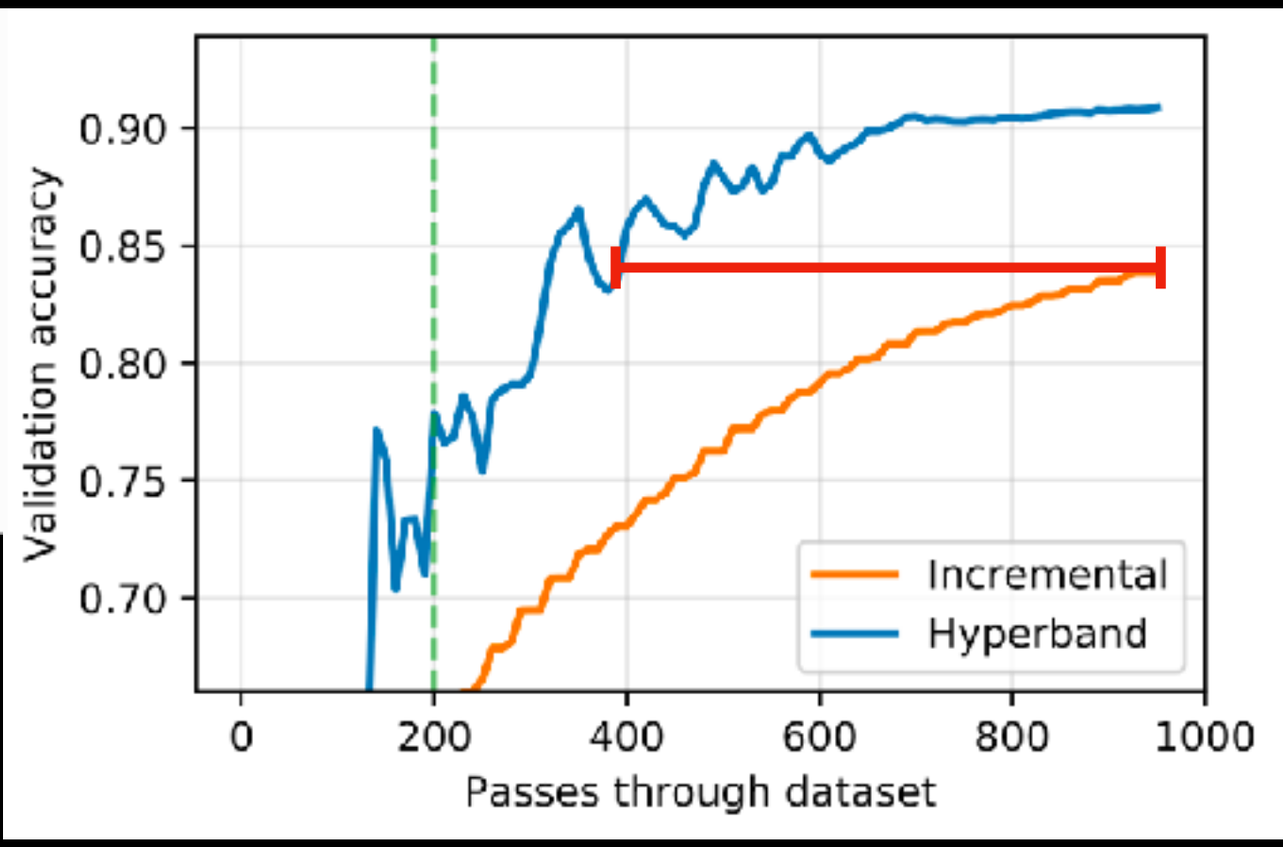
```
search.best_estimator_
search.best_params_
```

https://github.com/stsievert/talks

How do `HyperbandSearchCV` and `RandomizedSearchCV` perform?



The worst of the `hyperband` runs performs better than 50% of the `passive` runs.



In these experiments, `HyperbandSearchCV`...

- finds high performing hyperparameters with high confidence
- requires 1/3rd less data than RandomizedSearchCV to reach a particular validation accuracy

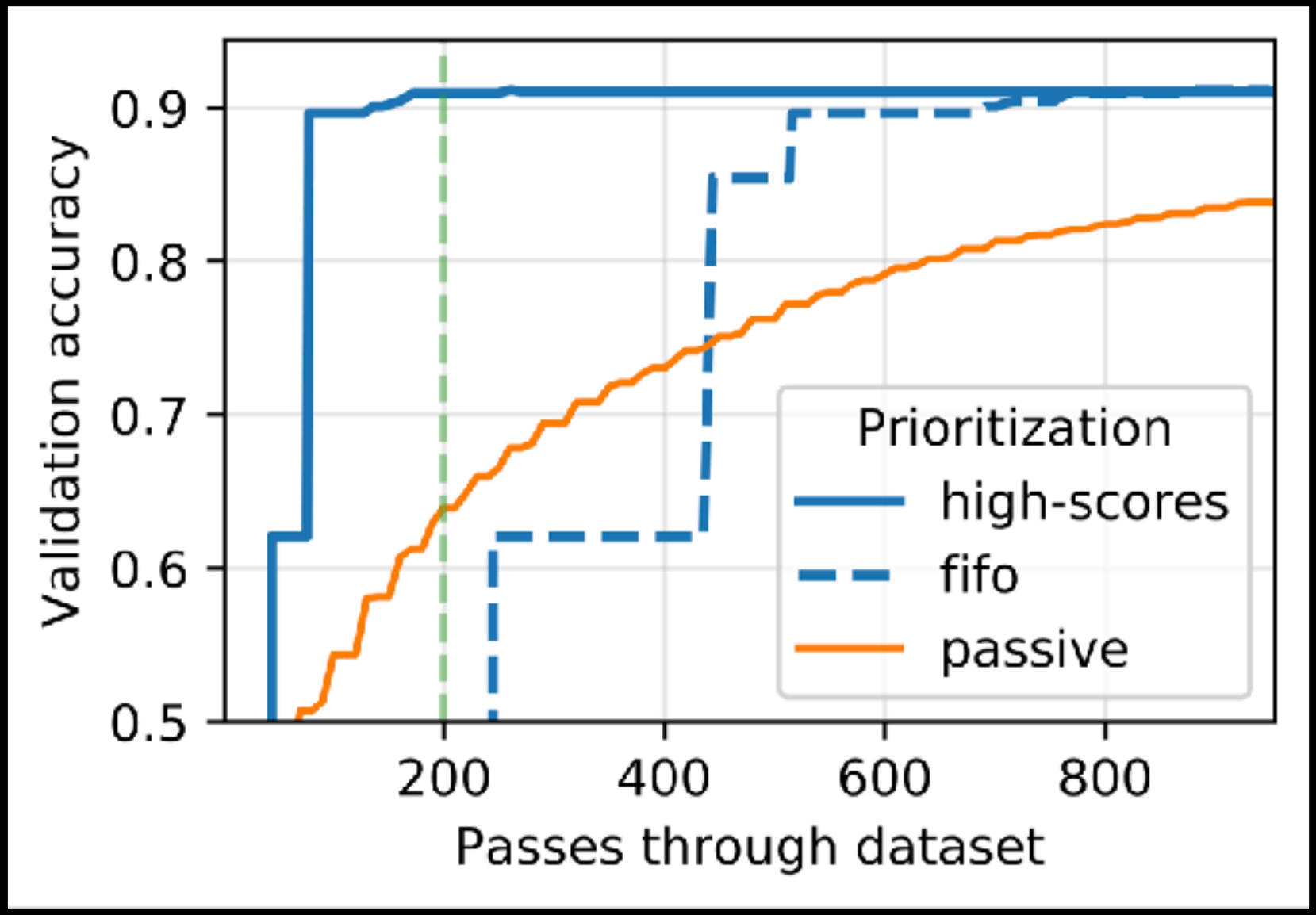
200 runs with different random seeds.

https://github.com/stsievert/dask-hyperband-comparison

https://github.com/stsievert/talks

How does Dask help Hyperband?

Dask assigns higher priority to models with higher scores.



Serial environments benefit the most from this.

https://github.com/stsievert/talks

Dask implementation:

Example 2

Deep learning model with PyTorch

Cluster w/ up to 32 workers

https://github.com/stsievert/talks

Parallel experiment

Use case: many computational resources, trying to optimize hyperparameters

Model

Custom built neural network with PyTorch (with wrapper Skorch)

```
from autoencoder import Autoencoder
from skorch import NeuralNetRegressor
```

PyTorch

```
import torch.nn as nn


class Autoencoder(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(1, 32, 3, 1)
        …

    def forward(self, x):
        x = self.conv1(x)
        …
        return output

model = NeuralNetRegressor(Autoencoder, …)
```

https://github.com/stsievert/talks

Parallel experiment

Use case: many computational resources, trying to optimize hyperparameters

Model

Custom built neural network with PyTorch (with wrapper Skorch)

```
from autoencoder import Autoencoder
from skorch import NeuralNetRegressor

model = NeuralNetRegressor(Autoencoder, …)
```

PyTorch

Search space

- 4 discrete hyperparameters w/ 160 unique combos
- 3 continuous hyperparameters

```
params = {
    "module__activation": …,  # 4 choices
    "module__init": …,  # 4 choices
    "batch_size": …,  # 5 choices
    "optimizer": …,  # 2 choices
    "optimizer__momentum": …,  # continuous
    "optimizer__lr": …,  # continuous
    "weight_decay": …,  # continuous
}
```

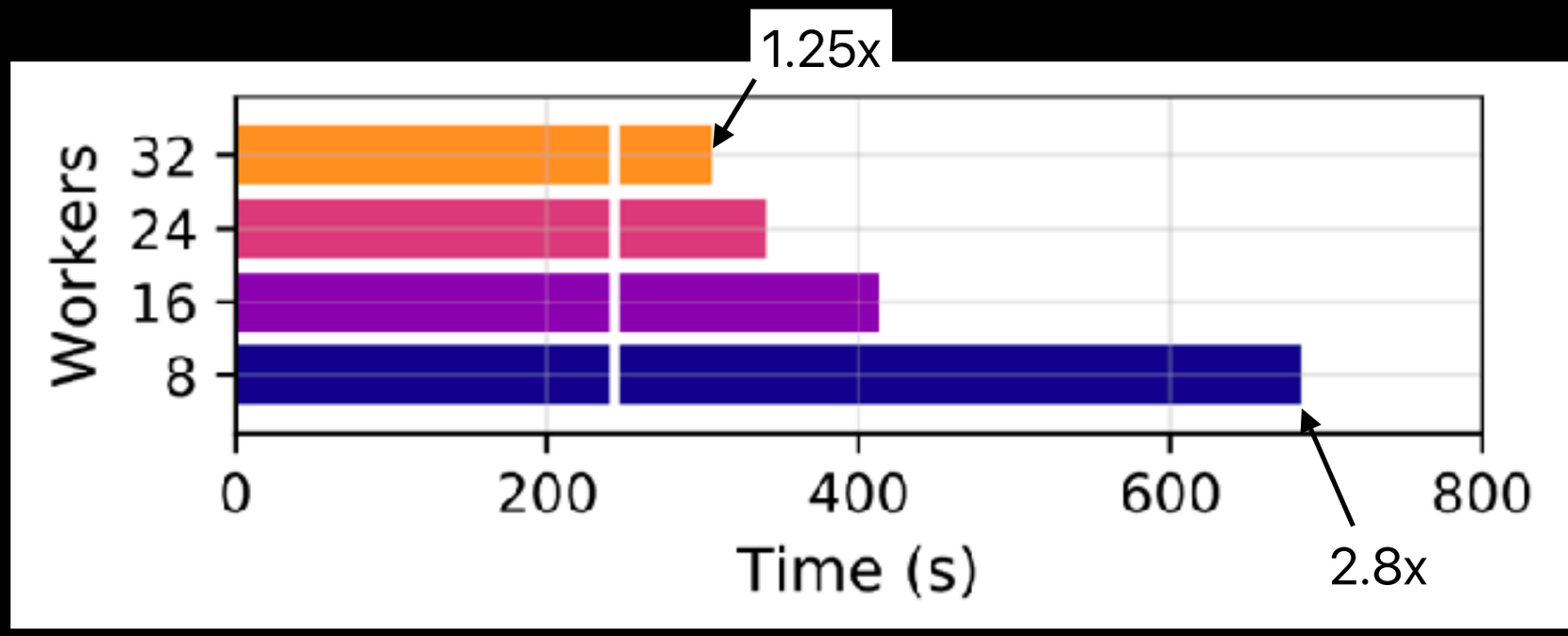
https://github.com/stsievert/talks

Parallel experiment

How does `HyperbandSearchCV` behave when the number of workers is varied?

```python
# … model/params/train data/n_params definition

search = HyperbandSearchCV(
    model, params,
    max_iter=n_params,
    patience=True,
    tol=0.001,
)
search.fit(X_train, y_train)
```

In this experiment, `HyperbandSearchCV` speedups saturate around 16–24 workers



https://github.com/stsievert/talks

Benefits of using Dask-ML for hyperparameter optimization

To find the best hyperparameters,
Dask-ML will...

- return high scoring models with certainty
- require ~1/3rd of the data `RandomizedSearchCV` requires in a serial environment
- require ~1.5x the time required for one model in a parallel environment

Dask-ML's hyperparameter optimization finds high performing hyperparameters quickly.

Thanks!

Questions?

https://github.com/stsievert/talks

Future work

Better support very exploratory hyperparameter searches.

Extend to case where models don't need `partial_fit`.

(i.e., dataset size as the scarce resource, not number of `partial_fit` calls)

https://github.com/stsievert/talks

There is an asynchronous version of Hyperband. Is that part of future work?

No. Dask's advanced task scheduling eliminates the need for that algorithm.

Specifically, the asynchronous variant is designed to reduce time to solution when brackets are run *in serial*.

https://github.com/stsievert/talks