Better and Faster Hyperparameter Optimization with Dask-ML

Scott Sievert
@stsievert

ANACONDA

Tom Augspurger
@TomAugspurger

DASK

Matthew Rocklin
@mrocklin

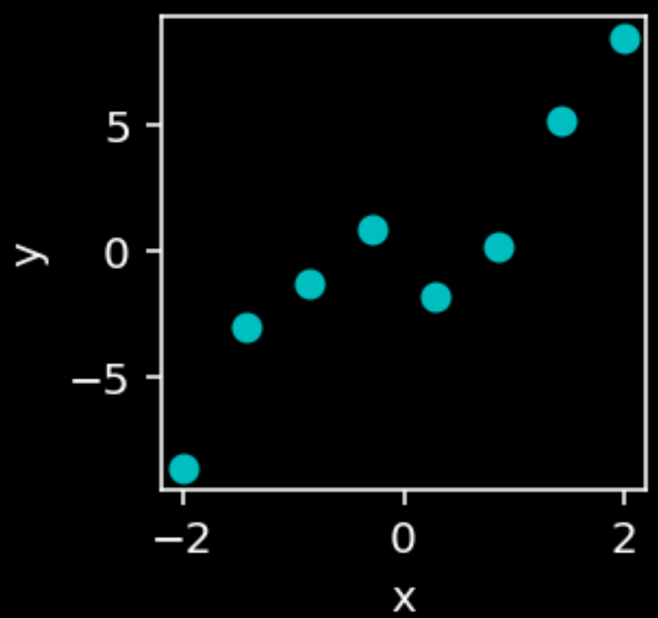
WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Tweet by @stsievert under #SciPy2019 tag https://github.com/stsievert/talks

- What is a hyperparameter?
- What's hyperparameter optimization?
- What new opportunities can Dask enable?
- How should the chosen algorithm be used, and how does it perform?

What is a hyperparameter?

Train data



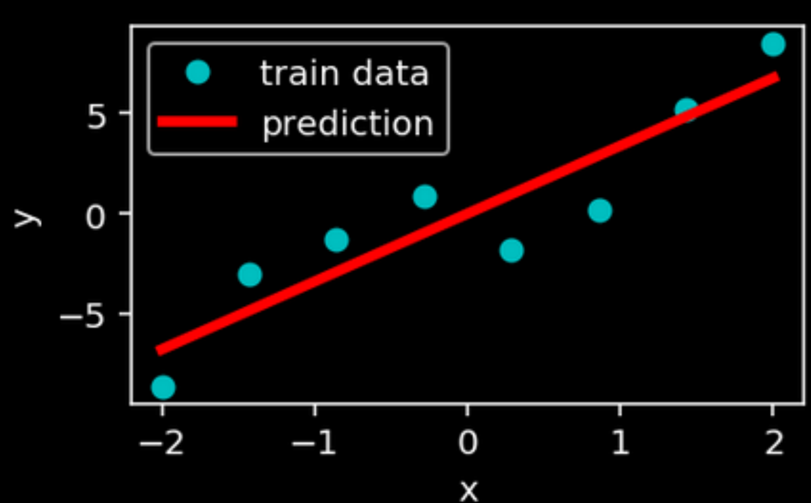
A free parameter not learned from data.

Typically used to define model structure.

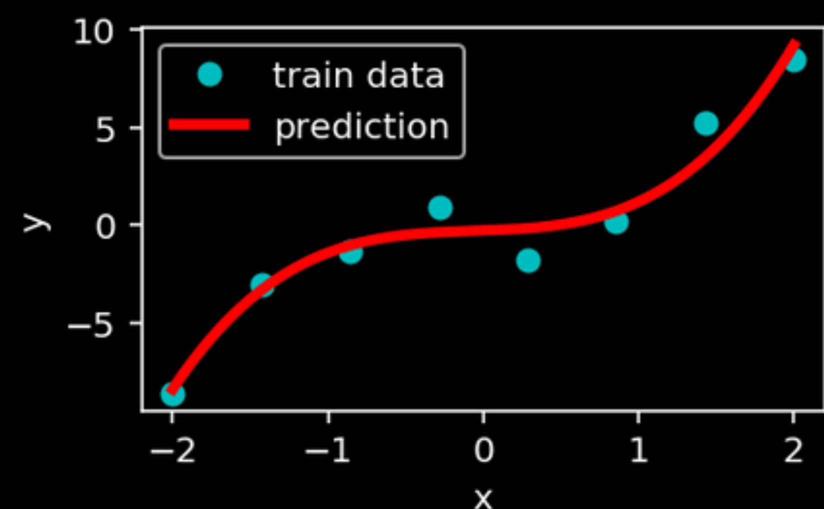
Model: polynomials of degree d

```
y = x**d + x**(d-1) + … + x
```

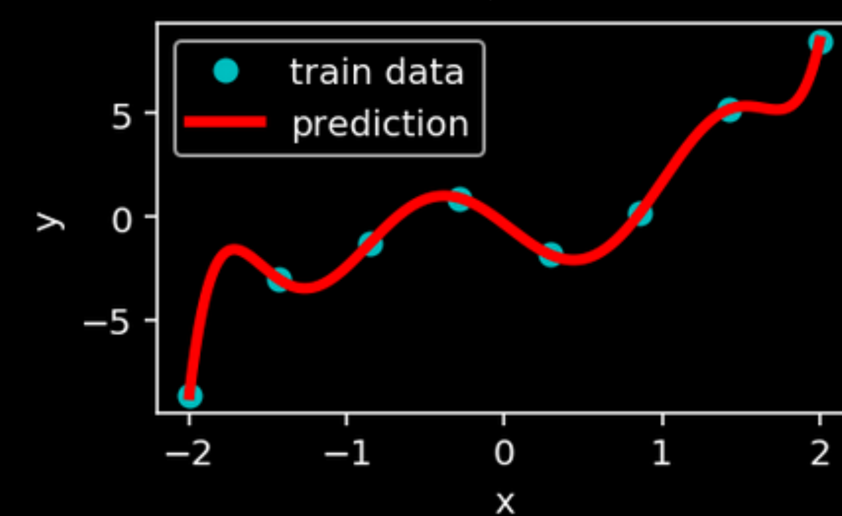
d=1



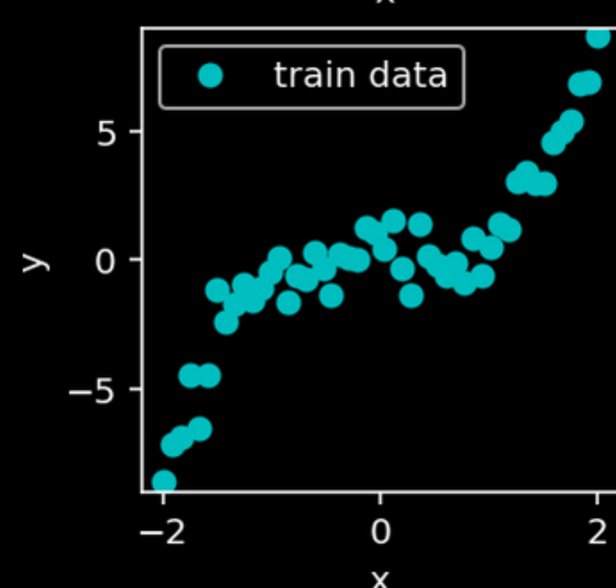
d=3



d=



Unseen validation data



https://github.com/stsievert/talks

Example

Distill

How to Use t-SNE Effectively

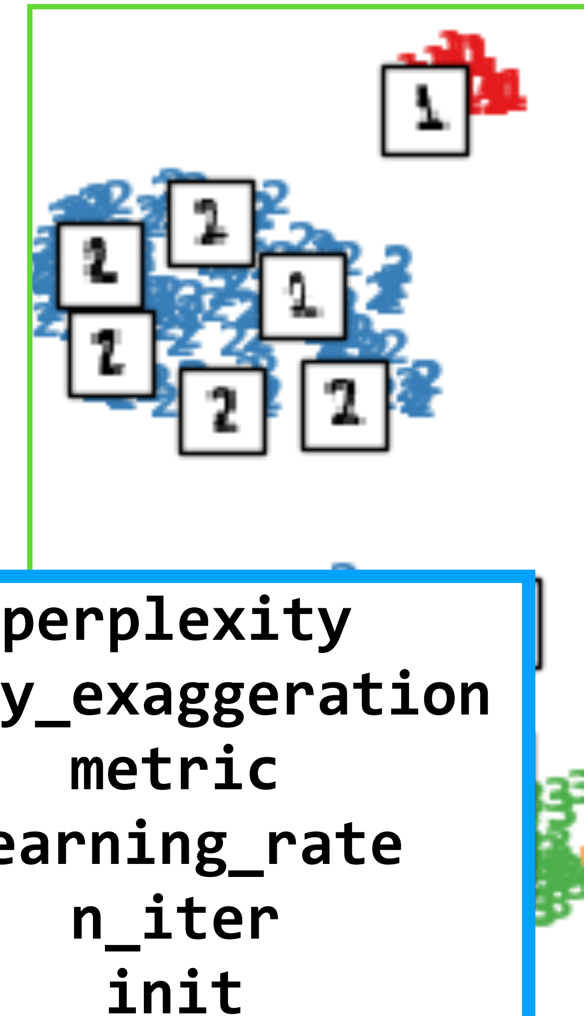
MARTIN WATTENBERG
Google Brain

FERNANDA VIÉGAS
Google Brain

IAN JOHNSON
Google Cloud

Oct. 13
2016

Citation:
Wattenberg, et al., 2016

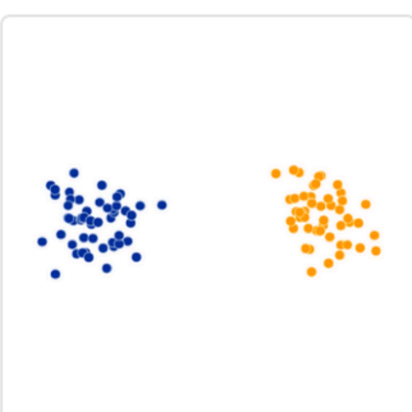


⋮

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 conside clusters in a 2D plane, as shown in the lefthand diagram. (For clarity, the



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

Dask-ML will find the *best* set of hyperparameters quickly

What other algorithms solve the “hyperparameter optimization” problem?

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

https://github.com/stsievert/talks

RandomizedSearchCV

✓ Simple implementation

✓ Easy to parallelize

✗ Does not limit computation

Computation will be significant for any complicated hyperparameter search*.

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

https://github.com/stsievert/talks

Hyperparameter optimization
+
Dask-ML

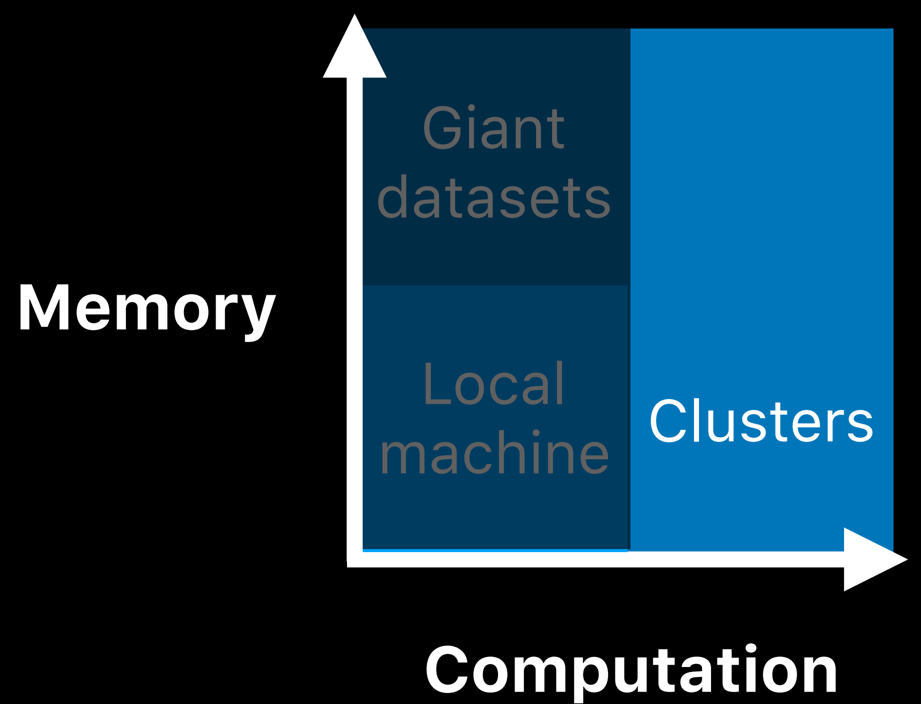
What algorithm is implemented in Dask-ML?

Why is it well-suited for Dask?

Dask

Dask natively scales Python

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

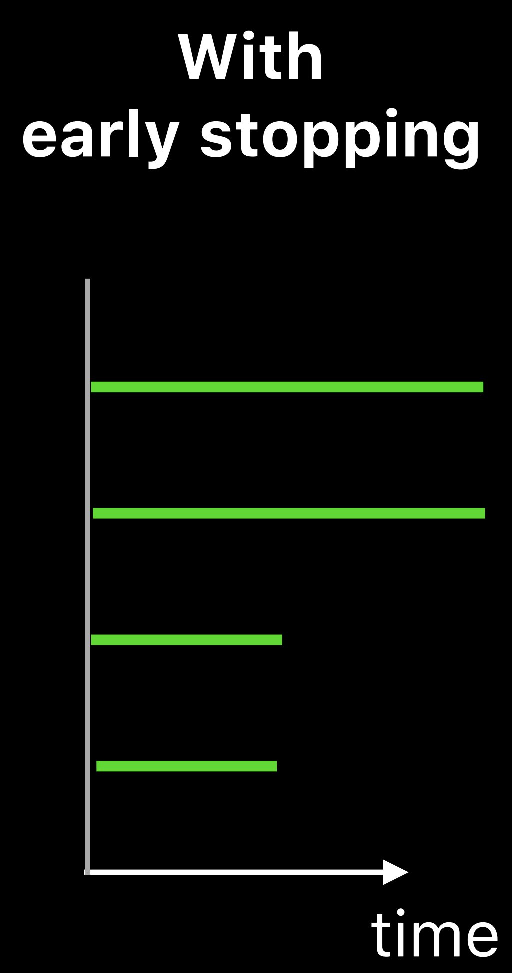


Selling points:
Easy to use
Declarative
Diagnostics

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

How can the computation be limited?

Early stopping of low performing models



Dask already has an implementation of `RandomizedSearchCV`

by Jim Crist, @jcrist

This naturally requires `partial_fit` or `warm_start`

https://github.com/stsievert/talks

Hyperband

Principled early stopping scheme
for random hyperparameter selection.

Hyperband $\boxed{\text{will*}}$ 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

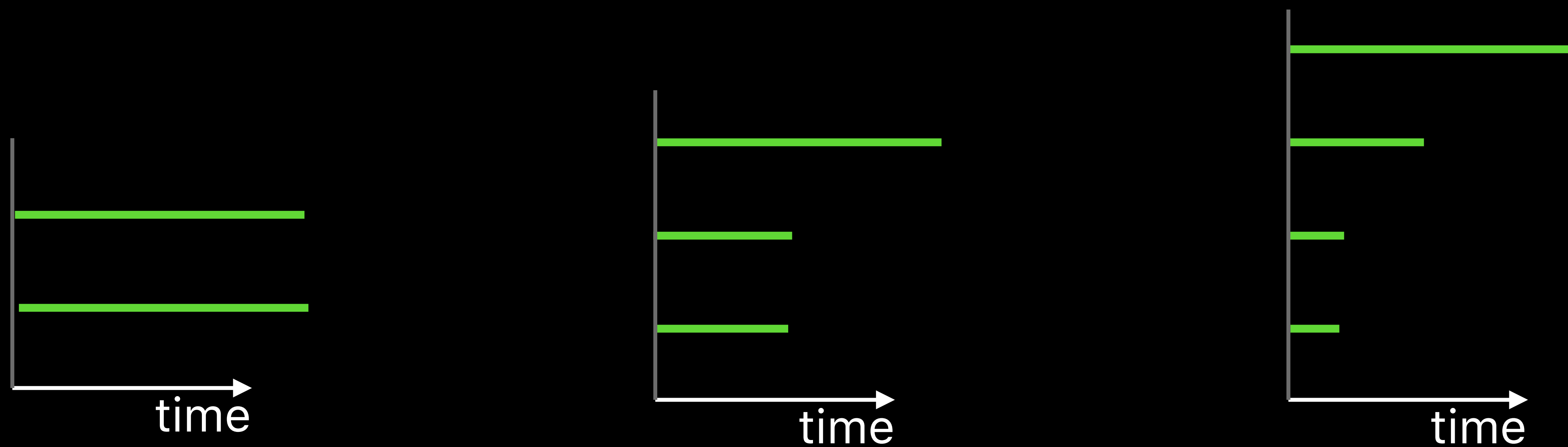
L Li, K Jamieson, G DeSalvo, A Rostamizadeh… - arXiv preprint arXiv …, 2016 - jmlr.org

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 …

*with high probability

https://github.com/stsievert/talks

Hyperband: intuition



Training time most important

Hyperparameter exploration most important

Sweeps over relative importance of training time

Hyperband architecture

Hyperband is an early stopping scheme for randomized search.

Number of models is reduced

```python
def early_stopping(n_models: int, calls: int, max_iter: int):
        -> BaseEstimator:
    models = [get_model() for _ in range(n_models)]
    while True:
        models = [train(m, calls) for m in models]
        models = top_k(models, k=len(models) // 3)  # 3 aka agressiveness
        calls *= 3
        if len(models) < 3:
            return top_k(models, k=1)

def hyperband(max_iter: int) -> BaseEstimator:
    brackets = [… for b in range(formula(max_iter))]

    # Each tuple is (num_models, n_init_calls)
    final_models = [early_stopping(n, r, max_iter) for n, r in brackets]
    return top_k(final_models, k=1)
```

https://github.com/stsievert/talks

Hyperband

- ✓ Simple implementation
- ✓ Easy to parallelize
- ✓ Effectively limits computation for complicated search spaces
- ✓ Mathematical justification

Hyperband in Dask-ML

Dask enables better performance.

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

What inputs are required?

How does it perform?

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

* to my knowledge

Dask-ML implementation:

Example 1

Laptop w/ 4 cores

Scikit-learn model

Synthetic simulation

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

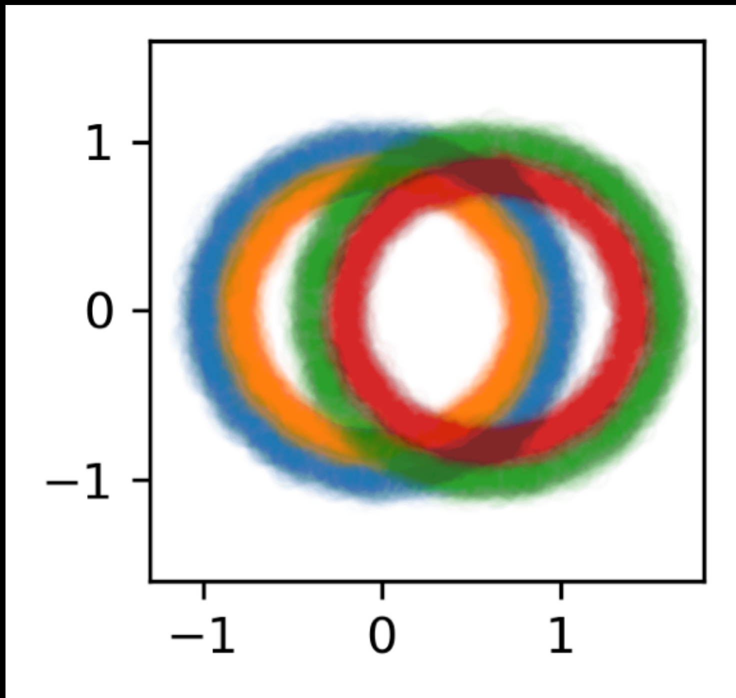
Model

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

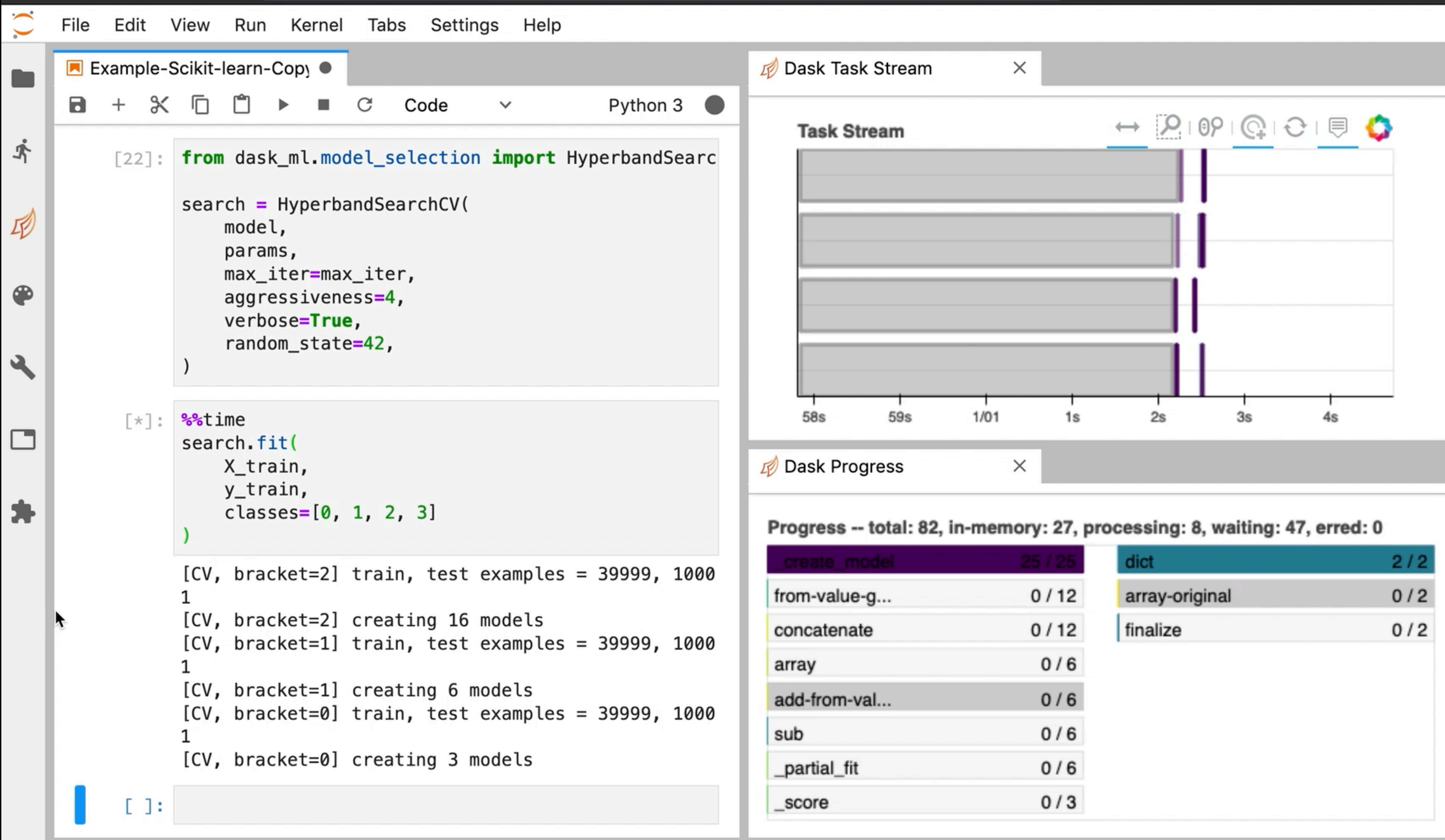
HyperbandSearchCV usage

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

```
X_train, y_train = rechunk(X, y, chunks=n_examples // n_params)
max_iter = n_params
```

```
search = HyperbandSearchCV(
    model, params,
    max_iter=n_params,
    patience=True,
    tol=0.001,
)
```

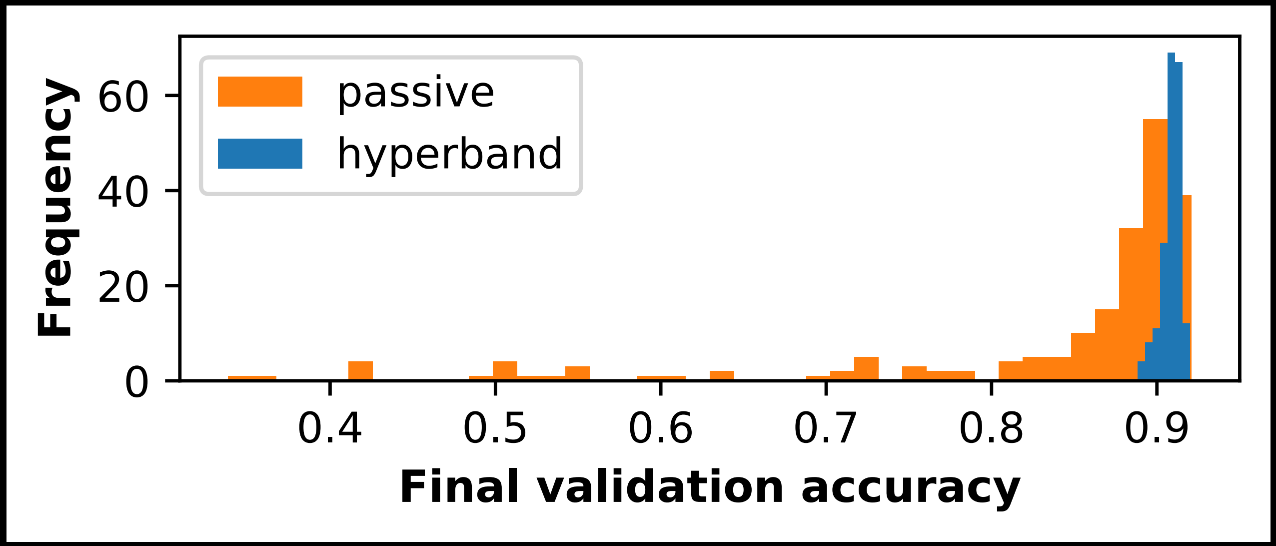
```
search.fit(X_train, y_train)
```



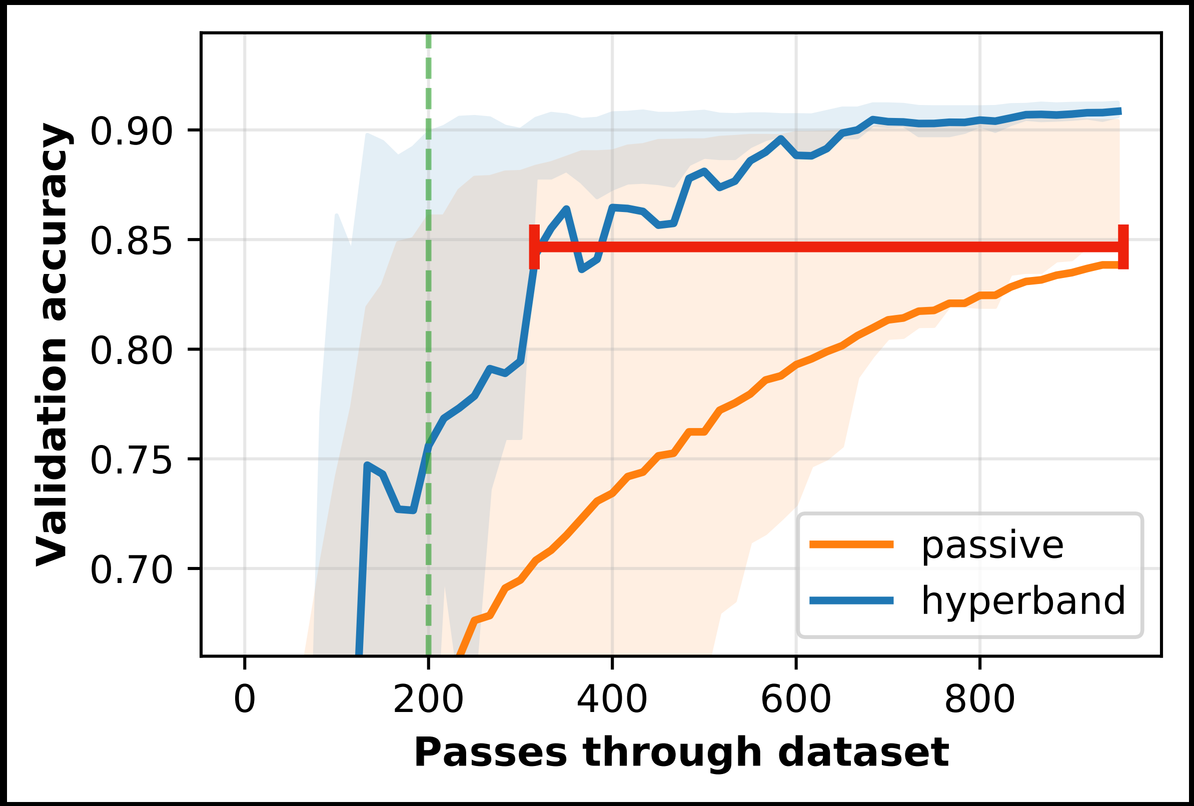

```
search.metadata
```

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

How do HyperbandSearchCV and RandomizedSearchCV perform?



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



In these experiments, HyperbandSearchCV...

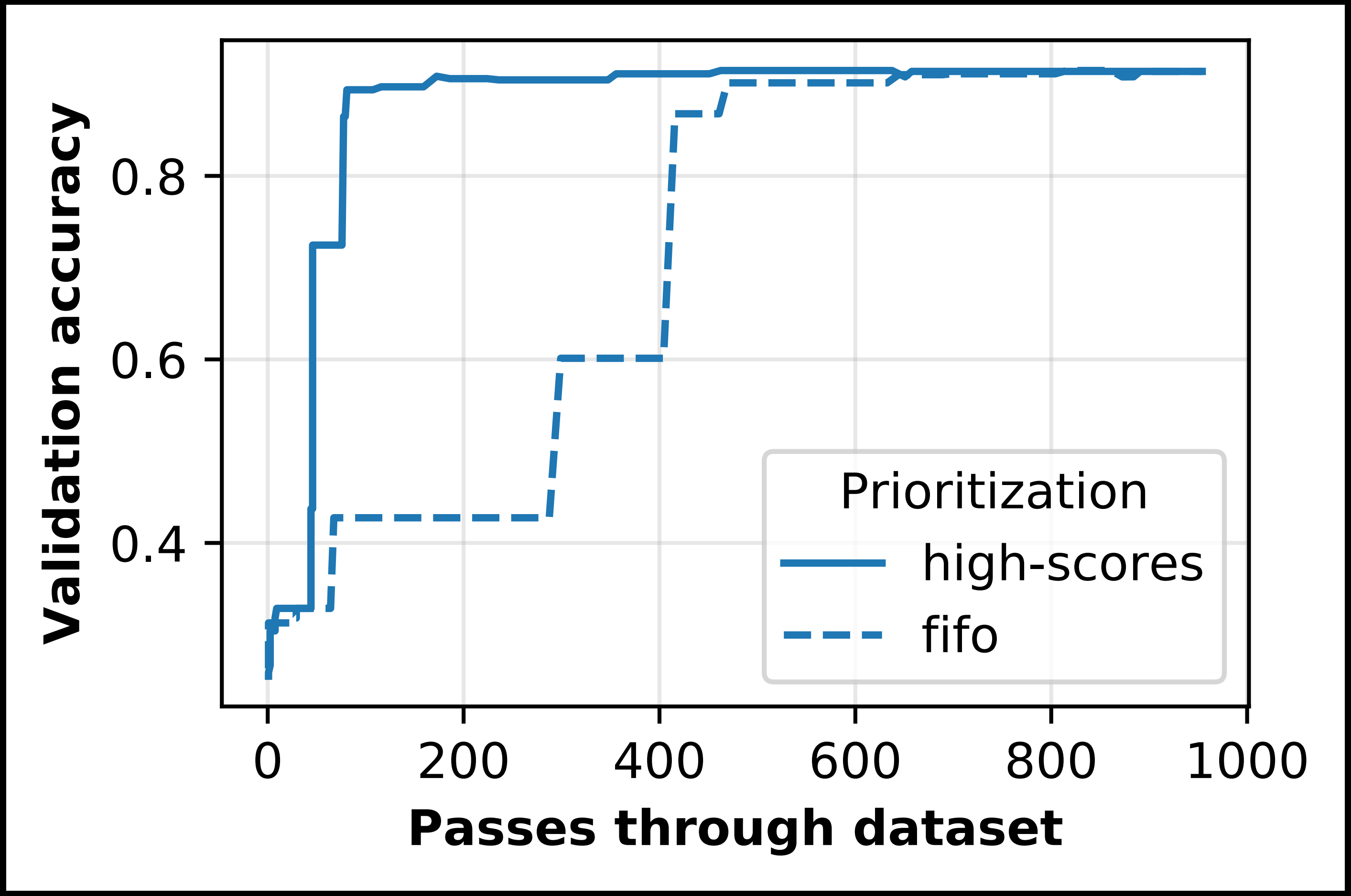
- finds high performing hyperparameters with high confidence
- requires 1/3rd less data than RandomizedSearchCV to reach a particular validation accuracy

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

Parallel experiment

Dataset

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

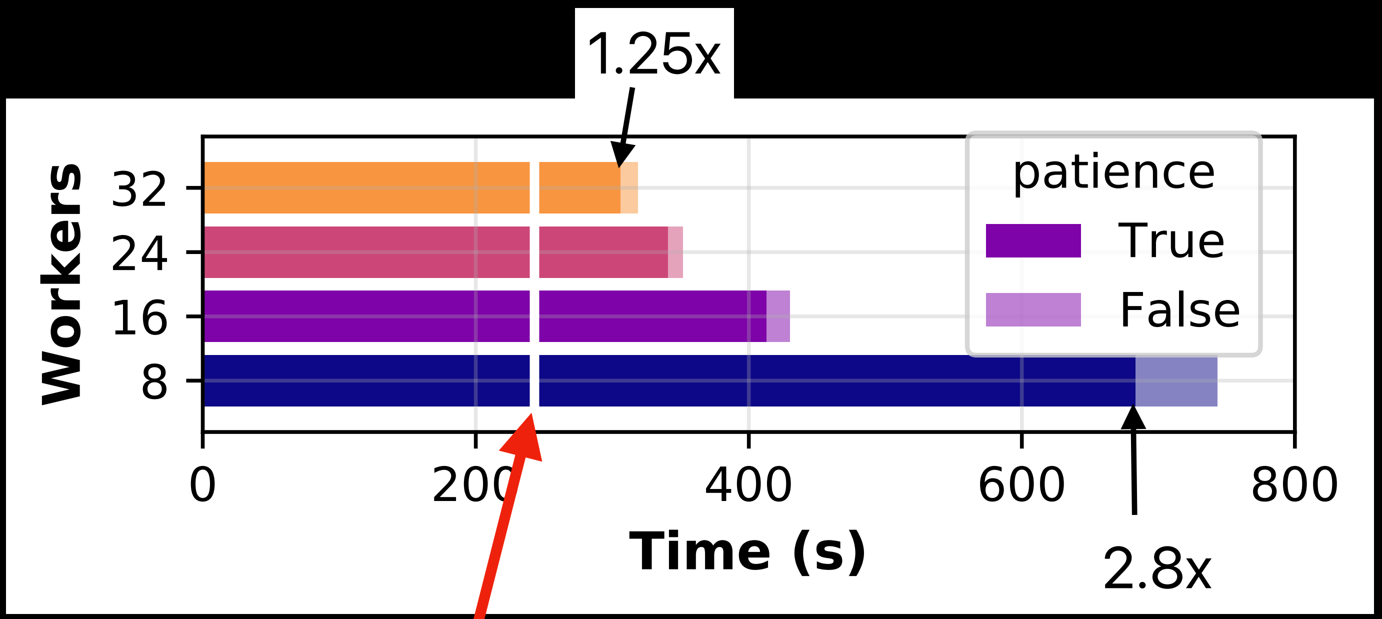
Parallel experiment

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

```
# … 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



time required to train one model

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.

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

Thanks!

Questions?

https://github.com/stsievert/talks

Future work

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

This will treat dataset size as the scarce resource, not number of `partial_fit` calls.

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*.