Optuna

A Define-by-Run Hyperparameter Optimization Framework

第1回 ディープラーニング分散ハッカソン@東工大
2019年8月5日
柳瀬 利彦, Preferred Networks

Preferred Networks

Materials

https://bit.ly/t3-optuna

- Optuna Tutorial
- 公式Examples
- 本ハッカソン向け Optuna Examples

OPTUNA

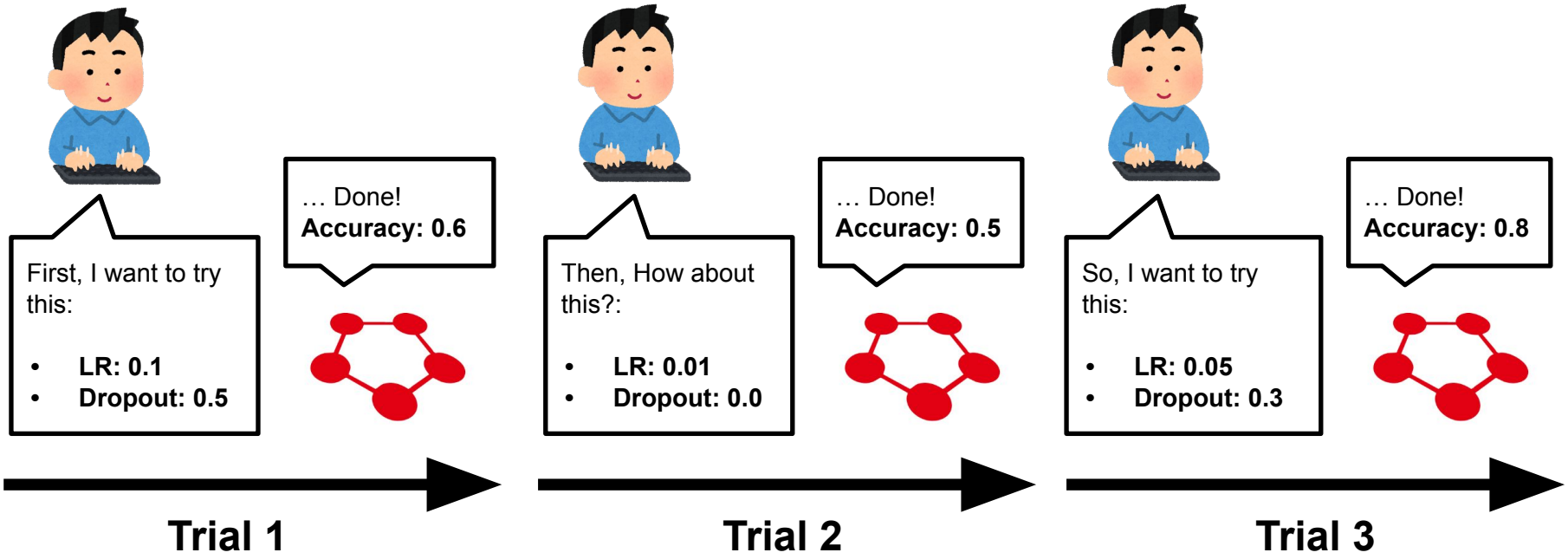
Optuna: A hyperparameter optimization framework

pypi v0.14.0 | license MIT | PASSED | docs passing | codecov 90%

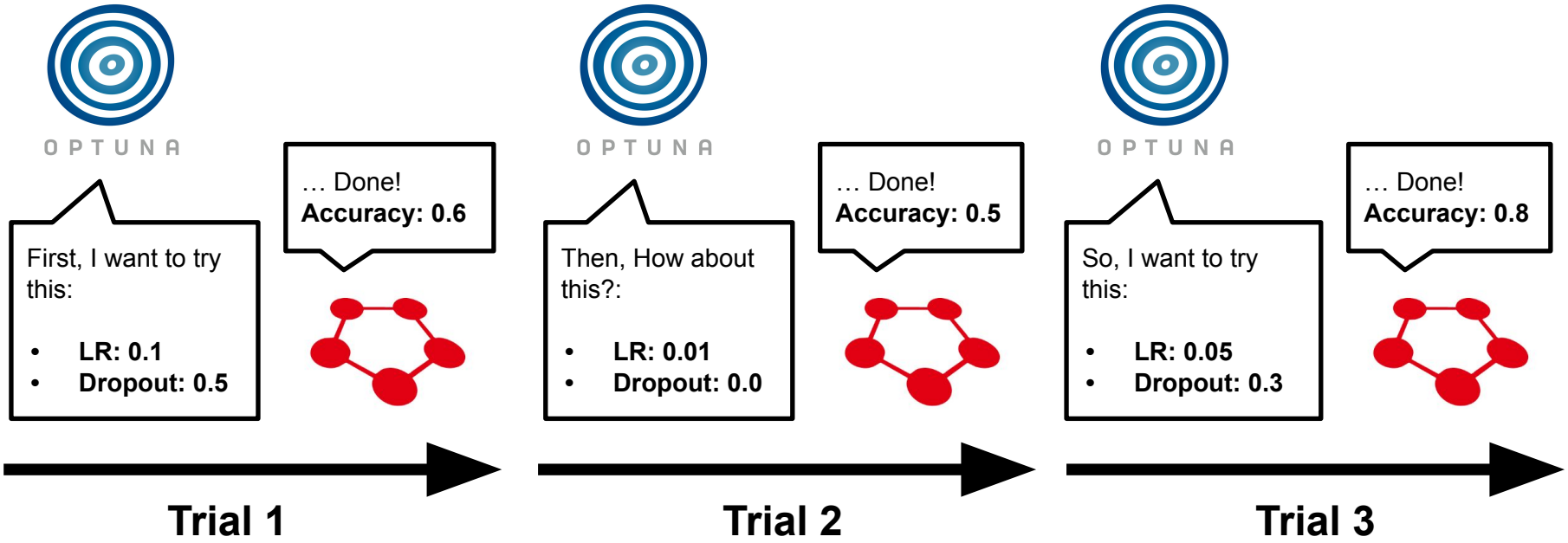
Website | **Docs** | **Install Guide** | **Tutorial**

Optuna is an automatic hyperparameter optimization software framework, particularly designed for machine learning. It features an imperative, *define-by-run* style user API. Thanks to our *define-by-run* API, the code written with Optuna enjoys high modularity, and the user of Optuna can dynamically construct the search spaces for the hyperparameters.

Hyperparameter Tuning



Hyperparameter Tuning



Quick Start

環境構築

- Python 2.7, 3.5+ をサポート
- **TensorFlowでも使えます！**
- Install Optuna by pip:

```
$ pip install optuna
```

MNIST Training (Optunaなし)

```
import sklearn
import sklearn.datasets
import sklearn.neural_network

def main():

    # ネットワーク構造の決定
    layers = [100, 100, 100]

    # 学習・評価用データの取得
    mnist = sklearn.datasets.fetch_mldata('MNIST original')
    x_train, x_test, y_train, y_test = sklearn.model_selection.train_test_split(
        mnist.data, mnist.target)

    # モデルの学習
    clf = sklearn.neural_network.MLPClassifier(hidden_layer_sizes=tuple(layers))
    clf.fit(x_train, y_train)

    # 学習したモデルの評価
    print(clf.score(x_test, y_test))

main()
```

MLPの構造を最適化!

MNIST Training (Optunaあり)

```python
import optuna
import sklearn
import sklearn.datasets
import sklearn.neural_network
```

3箇所変更

```python
def objective(trial):

    # ネットワーク構造の決定
    n_layers = trial.suggest_int('n_layers', 1, 4)
    layers = []
    for i in range(n_layers):
        layers.append(trial.suggest_int(f'n_units_{i}', 1, 100))

    # 学習・評価用データの取得
    mnist = sklearn.datasets.fetch_mldata('MNIST original')
    x_train, x_test, y_train, y_test = sklearn.model_selection.train_test_split(
        mnist.data, mnist.target)

    # モデルの学習
    clf = sklearn.neural_network.MLPClassifier(hidden_layer_sizes=tuple(layers))
    clf.fit(x_train, y_train)

    # 学習したモデルの評価
    return clf.score(x_test, y_test)
```

1 2

```python
study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=100)
```

3

```python
def objective(trial):

    # ネットワーク構造の決定
    n_layers = trial.suggest_int('n_layers', 1, 4)
    layers = []
    for i in range(n_layers):
        layers.append(trial.suggest_int(f'n_units_{i}', 1, 100))

    # 学習・評価用データの取得
    mnist = sklearn.datasets.fetch_mldata('MNIST original')
    x_train, x_test, y_train, y_test = sklearn.model_selection.train_test_split(
        mnist.data, mnist.target)

    # モデルの学習
    clf = sklearn.neural_network.MLPClassifier(hidden_layer_sizes=tuple(layers))
    clf.fit(x_train, y_train)

    # 学習したモデルの評価
    return clf.score(x_test, y_test)
```

1. 学習・評価ロジックを目的関数として定義. 評価値をreturnする.

2. suggest() でハイパーパラメタを取得

```
def objective(trial):

    # ネットワーク構造の決定
    n_layers = trial.suggest_int('n_layers', 1, 4)
    layers = []
    for i in range(n_layers):
        layers.append(trial.suggest_int(f'n_units_{i}', 1, 100))

    # 学習・評価用データの取得
    mnist = sklearn.datasets.fetch_mldata('MNIST original')
    x_train, x_test, y_train, y_test = sklearn.model_selection.train_test_split(
        mnist.data, mnist.target)

    # モデルの学習
    clf = sklearn.neural_network.MLPClassifier(hidden_layer_sizes=tuple(layers))
    clf.fit(x_train, y_train)

    # 学習したモデルの評価
    return clf.score(x_test, y_test)
```

```
    # モデルの学習
    clf = sklearn.neural_network.MLPClassifier(hidden_layer_sizes=tuple(layers))
    clf.fit(x_train, y_train)

    # 学習したモデルの評価
    return clf.score(x_test, y_test)
```

```
study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=100)
```

3. Study は実験を管理するオブジェクト
Study.optimize() でサーチ開始

Pandasで最適化結果を分析

```
study.trials_dataframe()
```

	state	value	datetime_start	datetime_complete	params				
					n_layers	n_units_0	n_units_1	n_units_2	n_units_3
0	TrialState.COMPLETE	0.063886	2019-02-12 19:48:27.508906	2019-02-12 19:49:04.886230	2	36	52.0	NaN	NaN
1	TrialState.COMPLETE	0.043600	2019-02-12 19:49:04.901480	2019-02-12 19:49:48.034385	2	45	33.0	NaN	NaN
2	TrialState.COMPLETE	0.042857	2019-02-12 19:49:48.049943	2019-02-12 19:50:11.762763	2	91	87.0	NaN	NaN
3	TrialState.COMPLETE	0.056914	2019-02-12 19:50:11.779041	2019-02-12 19:50:56.266840	1	42	NaN	NaN	NaN
4	TrialState.COMPLETE	0.295314	2019-02-12 19:50:56.281863	2019-02-12 19:51:25.699363	4	3	74.0	23.0	50.0
5	TrialState.COMPLETE	0.890171	2019-02-12 19:51:25.715028	2019-02-12 19:51:30.686130	3	20	45.0	1.0	NaN
6	TrialState.COMPLETE	0.042400	2019-02-12 19:51:30.702546	2019-02-12 19:52:16.760920	3	56	30.0	94.0	NaN

ニューラルネットの最適化ポイント

- ネットワークの形状
  - CNNのカーネルサイズ
    trial.suggest_categorical(‘ksize’, [3, 5, 7])
  - CNNのチャンネル数
    trial.suggest_int(‘n_channels’, 2, 128)
- 学習設定
  - 学習率
    trial.suggest_loguniform(‘lr’, 1e-9, 1e-1)
  - 正則化
    trial.suggest_uniform(‘dropout_rate’, 0.0, 1.0)

Pruning

Preferred Networks

Pruningの設定

Chainer, TFはExtensionを使うと1行で設定可
TF向けは@sfujiwaraさん制作！

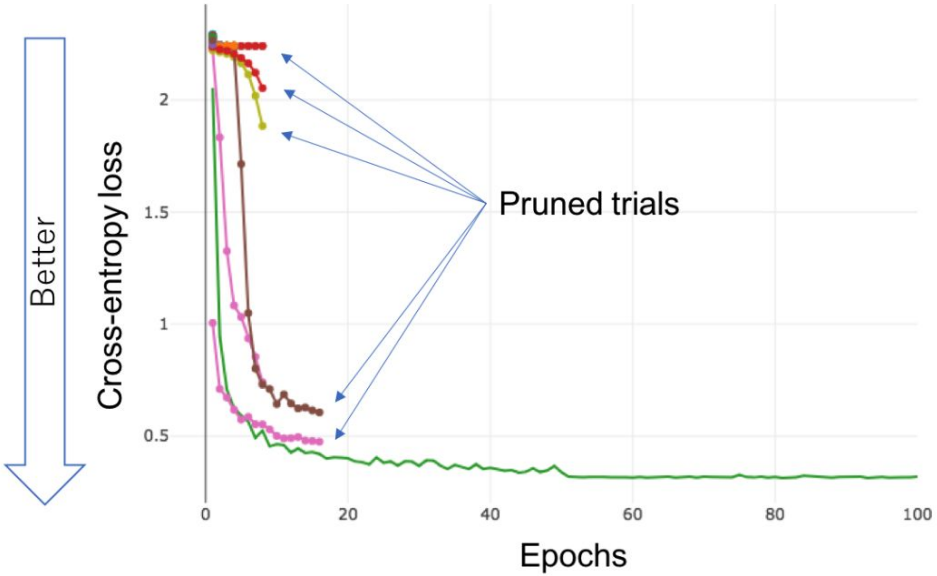
- 訓練の各イテレーションで:
  - ***report()*** と ***should_prune()*** を呼ぶ.

```
def objective(trial):
    ...
    alpha = trial.suggest_loguniform('alpha', 1e-5, 1e-1)
    clf = sklearn.linear_model.SGDClassifier(alpha=alpha)

    for step in range(100):
        clf.partial_fit(train_x, train_y, classes=classes)

        # Report intermediate objective value.
        intermediate_value = 1.0 - clf.score(test_x, test_y)
        trial.report(intermediate_value, step)

        # Handle pruning based on the intermediate value.
        if trial.should_prune():
            raise optuna.structs.TrialPruned()

    return clf.score(test_x, test_y)
```

中間結果を報告.

枝刈りの判定.

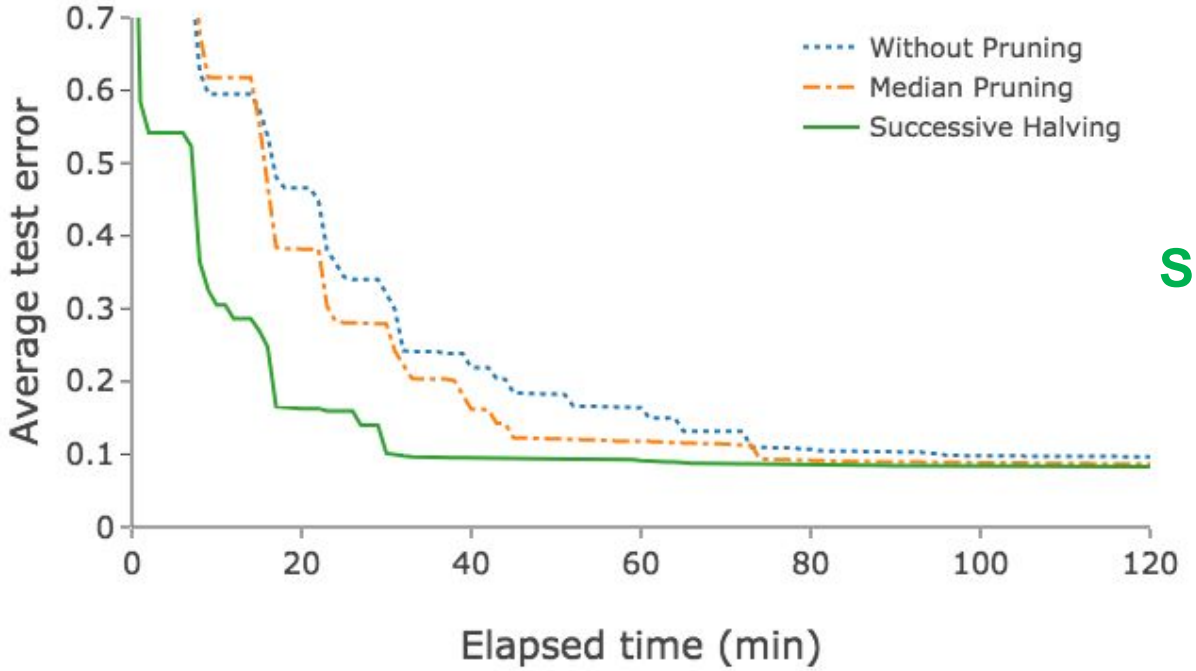
- 見込みのなさそうなTrialを自動的に停止
- 例）Validationスコアが過去のTrialのMedianより悪ければ打ち切る



Learning curves of pruned and completed trials.

https://bit.ly/t3-optuna

半分の時間で同等のエラー率



Median Pruning

各地点でMedian値を基準に，それより悪ければ止める

Successive Halving

Banditベースのアルゴリズム枝刈りの間隔に特徴（指数関数的に変化）.

Optimization of Distributed Deep Learning

Preferred Networks

Optunaの分散最適化機能

```
$ python example.py
[I 2019-05-21 11:16:43,493] Using an existing study with name 'example-study' instead of creating a new one.
[I 2019-05-21 11:16:53,916] Finished trial#9 resulted in value: 0.744140625. Current best value is 0.744140625 with parameters: {'momentum_sgd_lr': 7.944022303405195e-05, 'n_layers': 1, 'n_units_l0': 28.753028463759527, 'optimizer': 'MomentumSGD', 'weight_decay': 2.1924271434430527e-06}.
[I 2019-05-21 11:17:12,578] Finished trial#12 resulted in value: 0.6123798079788685. Current best value is 0.10862379521131516 with parameters: {'momentum_sgd_lr': 0.05635855981273252, 'n_layers': 3, 'n_units_l0': 16.903961893427955, 'n_units_l1': 11.068904780264166, 'n_units_l2': 83.07941264296184, 'optimizer': 'MomentumSGD', 'weight_decay': 3.509021664543648e-07}.
```

```
instead of creating a new one.
[I 2019-05-21 11:16:51,250] Finished trial#7 resulted in value: 0.8058143034577737. Current best value is 0.8058143034577737 with parameters: {'adam_alpha': 2.5501424552926803e-05, 'n_layers': 1, 'n_units_l0': 4.237994876174356, 'optimizer': 'Adam', 'weight_decay': 4.2909422937320695e-09}.
[I 2019-05-21 11:17:02,585] Finished trial#11 resulted in value: 0.14227764308452606. Current best value is 0.14227764308452606 with parameters: {'momentum_sgd_lr': 0.02227271795421007, 'n_layers': 3, 'n_units_l0': 6.004826062327049, 'n_units_l1': 27.182953480840045, 'n_units_l2': 24.852836817988184, 'optimizer': 'MomentumSGD', 'weight_decay': 1.378459092583759e-10}.
    total [#######################################.....] 79.79%
this epoch [################################################..] 97.87%
      187 iter, 7 epoch / 10 epochs
   22.572 iters/sec. Estimated time to finish: 0:00:02.098839.
```

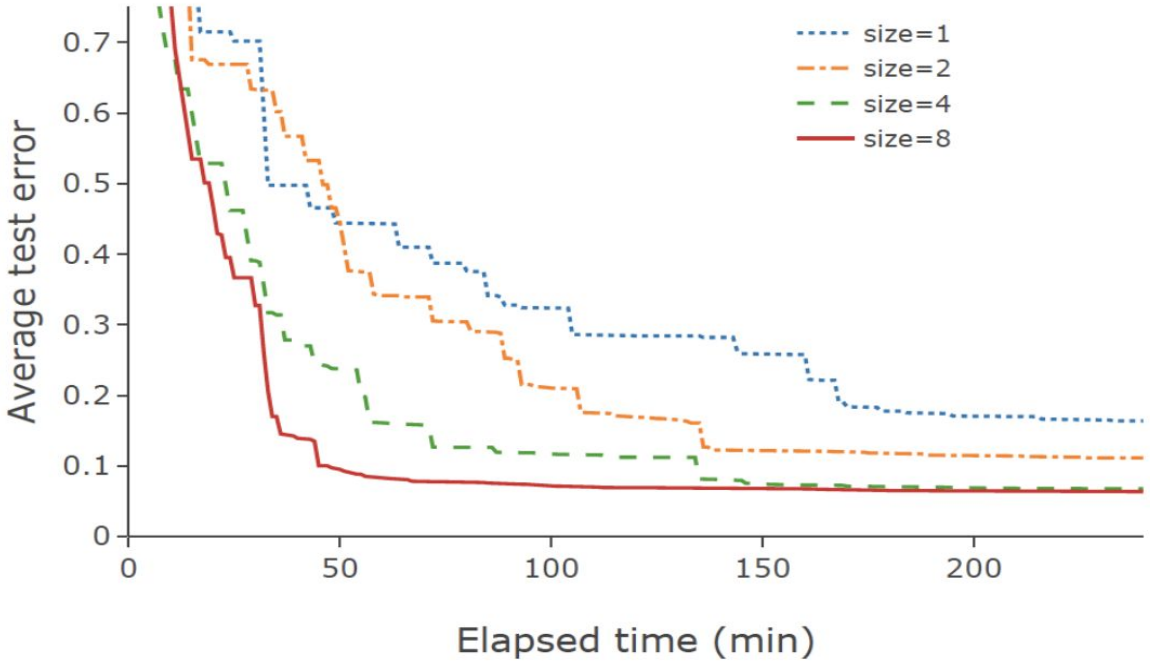
```
829. Current best value is 0.16624098271131516 with parameters: {'momentum_sgd_lr': 0.012378325131946236, 'n_layers': 3, 'n_units_l0': 10.118405976356666, 'n_units_l1': 99.22366485593328, 'n_units_l2': 6.970853785986278, 'optimizer': 'MomentumSGD', 'weight_decay': 0.0007099850265678694}.
[I 2019-05-21 11:17:07,320] Finished trial#15 resulted in value: 0.5231370180845261. Current best value is 0.14227764308452606 with parameters: {'momentum_sgd_lr': 0.02227271795421007, 'n_layers': 3, 'n_units_l0': 6.004826062327049, 'n_units_l1': 27.182953480840045, 'n_units_l2': 24.852836817988184, 'optimizer': 'MomentumSGD', 'weight_decay': 1.378459092583759e-10}.
    total [###################.........................] 38.40%
this epoch [############################################.........] 84.00%
      90 iter, 3 epoch / 10 epochs
   24.472 iters/sec. Estimated time to finish: 0:00:05.899583.
```

```
1315. Current best value is 0.16624098271131516 with parameters: {'momentum_sgd_lr': 0.012378325131946236, 'n_layers': 3, 'n_units_l0': 10.118405976356666, 'n_units_l1': 99.22366485593328, 'n_units_l2': 6.970853785986278, 'optimizer': 'MomentumSGD', 'weight_decay': 0.0007099850265678694}.
[I 2019-05-21 11:17:07,669] Finished trial#16 resulted in value: 0.10862379521131516. Current best value is 0.10862379521131516 with parameters: {'momentum_sgd_lr': 0.05635855981273252, 'n_layers': 3, 'n_units_l0': 16.903961893427955, 'n_units_l1': 11.068904780264166, 'n_units_l2': 83.07941264296184, 'optimizer': 'MomentumSGD', 'weight_decay': 3.509021664543648e-07}.
    total [###############.................................] 32.43%
this epoch [############..................................] 24.27%
      76 iter, 3 epoch / 10 epochs
   23.641 iters/sec. Estimated time to finish: 0:00:06.699203.
```

```
[I 2019-05-21 11:16:54,781] Finished trial#8 resulted in value: 0.9162409855052829. Current best value is 0.16624098271131516 with parameters: {'momentum_sgd_lr': 0.012378325131946236, 'n_layers': 3, 'n_units_l0': 10.118405976356666, 'n_units_l1': 99.22366485593328, 'n_units_l2': 6.970853785986278, 'optimizer': 'MomentumSGD', 'weight_decay': 0.0007099850265678694}.
[I 2019-05-21 11:17:05,203] Finished trial#14 resulted in value: 0.1547475978732109. Current best value is 0.14227764308452606 with parameters: {'momentum_sgd_lr': 0.02227271795421007, 'n_layers': 3, 'n_units_l0': 6.004826062327049, 'n_units_l1': 27.182953480840045, 'n_units_l2': 24.852836817988184, 'optimizer': 'MomentumSGD', 'weight_decay': 1.378459092583759e-10}.
    total [######################.........................] 49.92%
this epoch [#################################################.] 99.20%
      117 iter, 4 epoch / 10 epochs
   21.205 iters/sec. Estimated time to finish: 0:00:05.535154.
```

```
[I 2019-05-21 11:16:54,576] Finished trial#5 resulted in value: 0.16624098271131516. Current best value is 0.16624098271131516 with parameters: {'momentum_sgd_lr': 0.012378325131946236, 'n_layers': 3, 'n_units_l0': 10.118405976356666, 'n_units_l1': 99.22366485593328, 'n_units_l2': 6.970853785986278, 'optimizer': 'MomentumSGD', 'weight_decay': 0.0007099850265678694}.
[I 2019-05-21 11:17:09,047] Finished trial#13 resulted in value: 0.6319110579788685. Current best value is 0.10862379521131516 with parameters: {'momentum_sgd_lr': 0.05635855981273252, 'n_layers': 3, 'n_units_l0': 16.903961893427955, 'n_units_l1': 11.068904780264166, 'n_units_l2': 83.07941264296184, 'optimizer': 'MomentumSGD', 'weight_decay': 3.509021664543648e-07}.
    total [##########..........................................] 20.05%
this epoch [.....................................................]  0.53%
      47 iter, 2 epoch / 10 epochs
   24.64 iters/sec. Estimated time to finish: 0:00:07.604414.
```

ほぼ線形にスケール



Effect of parallelization sizes of 1, 2, 4, and 8.

https://bit.ly/t3-optuna

分散深層学習（データ並列）のチューニング

ChainerMN: **ChainerMNStudyを使う**

```
chainermn_study = optuna.integration.ChainerMNStudy(study, comm)
chainermn_study.optimize(objective, n_trials=10)
```

Tensorflow + Horovod: **MPIStudyを使う**

```
mpi_study = optuna.integration.MPIStudy(study, comm)
mpi_study.optimize(objective, n_trials=10)
```

TSUBAMEでOptunaを実行するには

Tensorflow/Chainer対応のexampleを提供

- シングルノードでOptuna
- マルチノードでOptuna

https://bit.ly/t3-optuna