100 XP

Running a hyperparameter tuning experiment

3 minutes

In Azure Machine Learning, you can tune hyperparameters by running a *hyperdrive* experiment.

Creating a training script for hyperparameter tuning

To run a hyperdrive experiment, you need to create a training script just the way you would do for any other training experiment, except that your script ***must***:

- Include an argument for each hyperparameter you want to vary.
- Log the target performance metric. This enables the hyperdrive run to evaluate the performance of the child runs it initiates, and identify the one that produces the best performing model.

For example, the following example script trains a logistic regression model using a **--regularization** argument to set the *regularization rate* hyperparameter, and logs the *accuracy* metric with the name **Accuracy**:

Python Copy

```python
import argparse
import joblib
from azureml.core import Run
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

# Get regularization hyperparameter
parser = argparse.ArgumentParser()
parser.add_argument('--regularization', type=float, dest='reg_rate', default=0.01)
args = parser.parse_args()
reg = args.reg_rate

# Get the experiment run context
run = Run.get_context()
```

```python
# load the training dataset
data = run.input_datasets['training_data'].to_pandas_dataframe()

# Separate features and labels, and split for training/validatiom
X = data[['feature1','feature2','feature3','feature4']].values
y = data['label'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)

# Train a logistic regression model with the reg hyperparameter
model = LogisticRegression(C=1/reg, solver="liblinear").fit(X_train, y_train)

# calculate and log accuracy
y_hat = model.predict(X_test)
acc = np.average(y_hat == y_test)
run.log('Accuracy', np.float(acc))

# Save the trained model
os.makedirs('outputs', exist_ok=True)
joblib.dump(value=model, filename='outputs/model.pkl')

run.complete()
```

Note

Note that in the Scikit-Learn **LogisticRegression** class, **C** is the *inverse* of the regularization rate; hence C=1/reg.

Configuring and running a hyperdrive experiment

To prepare the hyperdrive experiment, you must use a **HyperDriveConfig** object to configure the experiment run, as shown in the following example code:

Python Copy

```python
from azureml.core import Experiment
from azureml.train.hyperdrive import HyperDriveConfig, PrimaryMetricGoal

# Assumes ws, script_config and param_sampling are already defined

hyperdrive = HyperDriveConfig(run_config=script_config,
                              hyperparameter_sampling=param_sampling,
                              policy=None,
                              primary_metric_name='Accuracy',
                              primary_metric_goal=PrimaryMetricGoal.MAXIMIZE,
                              max_total_runs=6,
                              max_concurrent_runs=4)

experiment = Experiment(workspace = ws, name = 'hyperdrive_training')
hyperdrive_run = experiment.submit(config=hyperdrive)
```

Monitoring and reviewing hyperdrive runs

You can monitor hyperdrive experiments in Azure Machine Learning studio, or by using the Jupyter Notebooks **RunDetails** widget.

The experiment will initiate a child run for each hyperparameter combination to be tried, and you can retrieve the logged metrics these runs using the following code:

Python Copy

```python
for child_run in run.get_children():
    print(child_run.id, child_run.get_metrics())
```

You can also list all runs in descending order of performance like this:

Python Copy

```python
for child_run in hyperdrive_run.get_children_sorted_by_primary_metric():
    print(child_run)
```

To retrieve the best performing run, you can use the following code:

Python Copy

```python
best_run = hyperdrive_run.get_best_run_by_primary_metric()
```

Next unit: Exercise - Tune hyperparameters

Continue >