federated_learning_basic_concepts_random_seed

June 18, 2020

1 Federated learning: Simple experiment with seed

In this notebook we provide a simple example of how to make an experiment of a federated environment with the help of this framework. We are going to use a popular dataset to start the experimentation in a federated environment. The framework provides some functions to load the Emnist Digits dataset.

This notebook is a copy of Basic Concepts notebook. The difference is that here we set a seed using Reproducibility Singleton Class in order to ensure de reproducibility of the experiment. If you execute this experiment many times, you should obtain the same results.

[1]:
```python
from shfl.private.reproducibility import Reproducibility

# Server
Reproducibility(1234)

# In case of client
# Reproducibility.get_instance().set_seed(ID)
```

[1]:
```
<shfl.private.reproducibility.Reproducibility at 0x10b830f10>
```

[2]:
```python
import matplotlib.pyplot as plt
import shfl

database = shfl.data_base.Emnist()
train_data, train_labels, test_data, test_labels = database.load_data()
```

Let's inspect some properties of the loaded data.

[3]:
```python
print(len(train_data))
print(len(test_data))
print(type(train_data[0]))
train_data[0].shape
```

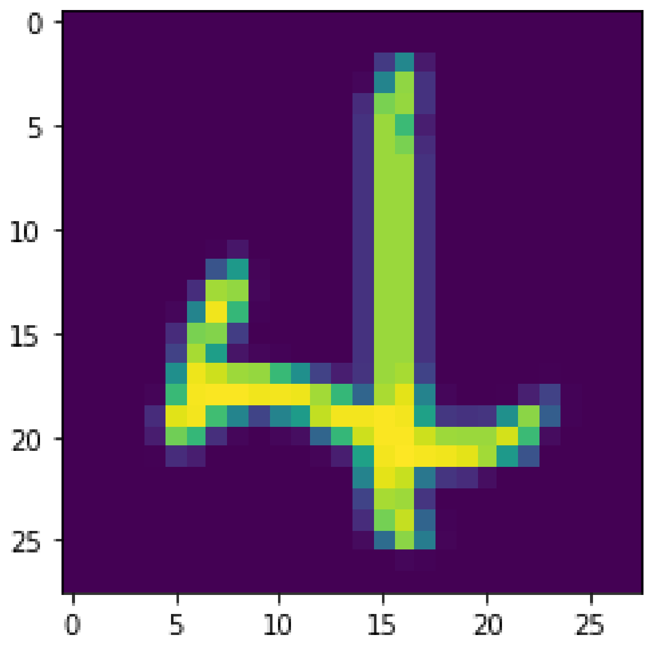
```
240000
40000
<class 'numpy.ndarray'>
```

[3]:
```
(28, 28)
```

So, as we have seen, our dataset is composed of a set of matrices of 28 by 28. Before starting with the federated scenario, we can take a look to a sample in the training data.

```
[4]: plt.imshow(train_data[0])
```

```
[4]: <matplotlib.image.AxesImage at 0x13d7258d0>
```



We are going to simulate a federated learning scenario with a set of client nodes containing private data, and a central server that will be responsible to coordinate the different clients. But, first of all, we have to simulate the data contained in every client. In order to do that, we are going to use the previously loaded dataset. The assumption in this example will be the data is distributed as a set of independent and identically distributed random variables, having every node approximately the same amount of data. There are a set of different possibilities in order to distribute the data. The distribution of the data is one of the factors that could impact more a federated algorithm. Therefore, the framework contains the implementation of some of the most common distributions that allow you to experiment different situations easily. In Federated Sampling you can dig into the options that the framework provides at the moment.

```
[5]: iid_distribution = shfl.data_distribution.IidDataDistribution(database)
     federated_data, test_data, test_labels = iid_distribution.
      ↪get_federated_data(num_nodes=20, percent=10)
```

That's it! We have created federated data from the Emnist dataset using 20 nodes and 10 percent of the available data. This data is distributed to a set of data nodes in the form of private data. Let's learn a little more about the federated data.

[6]:
```
print(type(federated_data))
print(federated_data.num_nodes())
federated_data[0].private_data
```

```
<class 'shfl.private.federated_operation.FederatedData'>
20
Node private data, you can see the data for debug purposes but the data remains
in the node
<class 'dict'>
{'5325872080': <shfl.private.data.LabeledData object at 0x13d803ad0>}
```

As we can see, private data in a node is not accesible directly but the framework provides mechanisms to use this data in a machine learning model. A federated learning algorithm is defined by a machine learning model locally deployed in each node that learns from the respective node's private data and an aggregating mechanism to aggregate the different model parameters uploaded by the client nodes to a central node. In this example we will use a deep learning model using keras to build it. The framework provides classes to allow using Tensorflow (see Basic Concepts Tensorflow) and Keras models into a federated learning scenario, your job is only to create a function acting as model builder. Moreover, the framework provides classes to allow using pretrained Tensorflow and Keras models (see Basic Concepts Pretrained Models). In this example build a Keras learning model.

[7]:
```
import tensorflow as tf

def model_builder():
    model = tf.keras.models.Sequential()
    model.add(tf.keras.layers.Conv2D(32, kernel_size=(3, 3), padding='same',␣
 ↪activation='relu', strides=1, input_shape=(28, 28, 1)))
    model.add(tf.keras.layers.MaxPooling2D(pool_size=2, strides=2,␣
 ↪padding='valid'))
    model.add(tf.keras.layers.Dropout(0.4))
    model.add(tf.keras.layers.Conv2D(32, kernel_size=(3, 3), padding='same',␣
 ↪activation='relu', strides=1))
    model.add(tf.keras.layers.MaxPooling2D(pool_size=2, strides=2,␣
 ↪padding='valid'))
    model.add(tf.keras.layers.Dropout(0.3))
    model.add(tf.keras.layers.Flatten())
    model.add(tf.keras.layers.Dense(128, activation='relu'))
    model.add(tf.keras.layers.Dropout(0.1))
    model.add(tf.keras.layers.Dense(64, activation='relu'))
    model.add(tf.keras.layers.Dense(10, activation='softmax'))

    model.compile(optimizer="rmsprop", loss="categorical_crossentropy",␣
 ↪metrics=["accuracy"])

    return shfl.model.DeepLearningModel(model)
```

Now, the only piece missing is the aggregation operator. Nevertheless, the framework provides some aggregation operators that we can use. In the following piece of code we define the federated aggregation mechanism. Moreover, we define de federated goverment based on the keras learning model, the federated data and the aggregation mechanism.

[8]:
```
aggregator = shfl.federated_aggregator.FedAvgAggregator()
federated_government = shfl.federated_government.
↪FederatedGovernment(model_builder, federated_data, aggregator)
```

If you want to see all the aggregation operators you can check the following notebook Federated Aggregation Operators. Before running the algorithm, we want to apply a transformation to the data. The good practise to do that is to define a federated operation that will ensure that the transformation is applied to the federated data in all the client nodes. We want to reshape the data, so we define the following FederatedTransformation.

[9]:
```
import numpy as np

class Reshape(shfl.private.FederatedTransformation):

    def apply(self, labeled_data):
        labeled_data.data = np.reshape(labeled_data.data, (labeled_data.data.
↪shape[0], labeled_data.data.shape[1], labeled_data.data.shape[2],1))

shfl.private.federated_operation.apply_federated_transformation(federated_data,␣
↪Reshape())
```

In addition, we want to normalize the data. We define a federated transformation using mean and standard deviation (std) parameters. We use mean and std estimated from the training set in this example. Although the ideal parameters would be an aggregation of the mean and std of each client's training datasets, we use the mean and std of the global dataset as a simple approximation.

[10]:
```
import numpy as np

class Normalize(shfl.private.FederatedTransformation):

    def __init__(self, mean, std):
        self.__mean = mean
        self.__std = std

    def apply(self, labeled_data):
        labeled_data.data = (labeled_data.data - self.__mean)/self.__std


mean = np.mean(train_data.data)
std = np.std(train_data.data)
shfl.private.federated_operation.apply_federated_transformation(federated_data,␣
↪Normalize(mean, std))
```

We are now ready to execute our federated learning algorithm.

```
[11]: test_data = np.reshape(test_data, (test_data.shape[0], test_data.shape[1],␣
      ↪test_data.shape[2],1))
      federated_government.run_rounds(3, test_data, test_labels)
```

```
Accuracy round 0
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803050>: [26.00153350830078, 0.7081500291824341]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13a611610>: [16.699399948120117, 0.7995499968528748]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803b10>: [17.581830978393555, 0.7689999938011169]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803990>: [28.322406768798828, 0.6869750022888184]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803e50>: [23.239553451538086, 0.7447749972343445]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d8031d0>: [26.507186889648438, 0.7370250225067139]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803110>: [24.91872787475586, 0.7216249704360962]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803310>: [16.83992576599121, 0.7810750007629395]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803250>: [19.653995513916016, 0.7558500170707703]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803d90>: [28.578378677368164, 0.7000749707221985]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d8034d0>: [27.314903259277344, 0.7062000036239624]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803390>: [29.45380401611328, 0.7059749960899353]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d73e550>: [29.225685119628906, 0.6867499947547913]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d74e690>: [21.673526763916016, 0.7437499761581421]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d74eb10>: [28.882831573486328, 0.7445250153541565]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d74e2d0>: [28.00448989868164, 0.683525025844574]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d7576d0>: [61.66908264160156, 0.5647249817848206]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d7573d0>: [25.390256881713867, 0.7035499811172485]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d7577d0>: [64.51156616210938, 0.5548999905586243]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d757b90>: [37.62916564941406, 0.6604250073432922]
Global model test performance : [15.991044044494629, 0.7271000146865845]
```

```
Accuracy round 1
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803050>: [14.448678016662598, 0.8658249974250793]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13a611610>: [14.665261268615723, 0.8454999923706055]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803b10>: [16.9344539642334, 0.8431000113487244]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803990>: [21.44463348388672, 0.8399749994277954]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803e50>: [16.434038162231445, 0.8399249911308289]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d8031d0>: [25.372880935668945, 0.7912750244140625]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803110>: [21.109840393066406, 0.8241999745368958]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803310>: [19.926753997802734, 0.8252999782562256]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803250>: [15.055240631103516, 0.8721250295639038]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803d90>: [21.573652267456055, 0.7998499870300293]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d8034d0>: [22.889238357543945, 0.8337500095367432]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803390>: [21.012056350708008, 0.8248000144958496]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d73e550>: [19.4395809173584, 0.8340250253677368]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d74e690>: [32.0885009765625, 0.7526749968528748]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d74eb10>: [20.388721466064453, 0.8524500131607056]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d74e2d0>: [21.156936645507812, 0.8154000043869019]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d7576d0>: [30.255231857299805, 0.7613499760627747]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d7573d0>: [19.865541458129883, 0.8258500099182129]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d7577d0>: [23.44260597229004, 0.8200749754905701]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d757b90>: [19.536724090576172, 0.8315500020980835]
Global model test performance : [13.228371620178223, 0.8762000203132629]


Accuracy round 2
```

```
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803050>: [18.647802352905273, 0.8677999973297119]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13a611610>: [16.85980224609375, 0.869700014591217]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803b10>: [20.671428680419922, 0.8512750267982483]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803990>: [23.068933486938477, 0.8503999710083008]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803e50>: [15.327696800231934, 0.865975022315979]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d8031d0>: [20.70783805847168, 0.8501750230789185]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803110>: [16.356706619262695, 0.8787999749183655]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803310>: [14.87657356262207, 0.8744500279426575]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803250>: [14.516626358032227, 0.8939499855041504]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803d90>: [25.08513832092285, 0.8401250243186951]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d8034d0>: [19.77056312561035, 0.8635749816894531]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d803390>: [19.20934295654297, 0.86080002784729]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d73e550>: [15.197554588317871, 0.8835999965667725]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d74e690>: [18.49679946899414, 0.87847501039505]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d74eb10>: [13.143366813659668, 0.8956500291824341]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d74e2d0>: [13.889097213745117, 0.8808000087738037]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d7576d0>: [17.549623489379883, 0.8471500277519226]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d7573d0>: [24.337947845458984, 0.8330749869346619]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d7577d0>: [18.916805267333984, 0.8689000010490417]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x13d757b90>: [21.717039108276367, 0.8470749855041504]
Global model test performance : [12.018816947937012, 0.9057000279426575]
```

[ ]: