federated_learning_basic_concepts

June 18, 2020

1 Federated learning: Simple experiment

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.

[1]:
```
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.

[2]:
```
print(len(train_data))
print(len(test_data))
print(type(train_data[0]))
train_data[0].shape
```

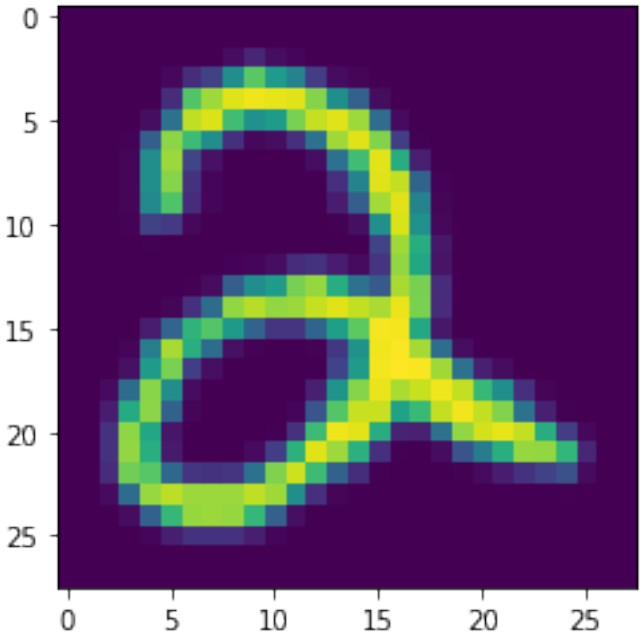
```
240000
40000
<class 'numpy.ndarray'>
```

[2]:
```
(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 at a sample in the training data.

[3]:
```
plt.imshow(train_data[0])
```

[3]:
```
<matplotlib.image.AxesImage at 0x142a5c910>
```



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 factor that could impact more to a federated algorithm. Therefore, the framework contains the implementation of some of the most common distribution that allow you experiment different situations easily. In this Federated Sampling you can dig into the options that the framework provides at the moment.

[4]:
```
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 a set of data nodes containing private data.
Let's learn a little more about the federated data.

[5]:
```
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'>
{'5368308880': <shfl.private.data.LabeledData object at 0x1409f2410>}
```

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.

[6]:
```
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 missing piece 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 the federated goverment based on the keras learning model, the federated data and the aggregation mechanism.

[7]:
```
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. A good practice 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.

[8]:
```
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.

[9]:
```
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.

[10]:
```
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 0x1409e2bd0>: [22.48199462890625, 0.7477499842643738]
```

```
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142a81a90>: [19.504619598388672, 0.7735000252723694]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142b436d0>: [21.34568214416504, 0.7403749823570251]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142b43f90>: [46.420963287353516, 0.6281999945640564]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142b43bd0>: [35.758663177490234, 0.7120500206947327]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142b43190>: [20.658639907836914, 0.7733500003814697]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142a911d0>: [27.328189849853516, 0.7294250130653381]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142a91350>: [19.064477920532227, 0.775825023651123]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142a91210>: [25.649051666259766, 0.7170500159263611]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142a91e90>: [17.36447525024414, 0.7831249833106995]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142a91ed0>: [27.77058219909668, 0.6686999797821045]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142a91950>: [24.73395538330078, 0.7268000245094299]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac3c90>: [30.221622467041016, 0.6709499955177307]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac35d0>: [29.076478958129883, 0.6456249952316284]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac3850>: [24.378236770629883, 0.7342000007629395]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac3050>: [26.278146743774414, 0.7451750040054321]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac3d10>: [23.44240951538086, 0.7418749928474426]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac3b10>: [24.969972610473633, 0.735450029373169]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac3f10>: [17.630361557006836, 0.7837499976158142]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac3110>: [32.493309020996094, 0.657800018787384]
Global model test performance : [19.0091495513916, 0.7213500142097473]



Accuracy round 1
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x1409e2bd0>: [20.916851043701172, 0.823074996471405]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142a81a90>: [14.88000202178955, 0.8565499782562256]
Test performance client <shfl.private.federated_operation.FederatedDataNode
```

```
object at 0x142b436d0>: [17.021211624145508, 0.8233500123023987]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142b43f90>: [50.92515182495117, 0.731249988079071]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142b43bd0>: [28.847049713134766, 0.7884250283241272]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142b43190>: [22.447240829467773, 0.8138499855995178]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142a911d0>: [26.028194427490234, 0.8169000148773193]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142a91350>: [17.808589935302734, 0.8561750054359436]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142a91210>: [23.098587036132812, 0.8356249928474426]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142a91e90>: [54.00608825683594, 0.679099977016449]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142a91ed0>: [24.039583206176758, 0.7912499904632568]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142a91950>: [20.355661392211914, 0.8317000269889832]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac3c90>: [40.551612854003906, 0.7125999927520752]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac35d0>: [28.25297737121582, 0.7662000060081482]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac3850>: [21.95855140686035, 0.8338500261306763]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac3050>: [30.376869201660156, 0.8117499947547913]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac3d10>: [21.918569564819336, 0.824150025844574]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac3b10>: [20.857391357421875, 0.847474992275238]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac3f10>: [20.780052185058594, 0.8320500254631042]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac3110>: [50.917938232421875, 0.6514250040054321]
Global model test performance : [14.437987327575684, 0.8705999851226807]



Accuracy round 2
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x1409e2bd0>: [21.35908317565918, 0.8571000099182129]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142a81a90>: [12.020793914794922, 0.8989499807357788]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142b436d0>: [15.64585018157959, 0.8693000078201294]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142b43f90>: [29.53978729248047, 0.8052250146865845]
```

```
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142b43bd0>: [21.734251022338867, 0.8600999712944031]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142b43190>: [20.923110961914062, 0.8526999950408936]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142a911d0>: [26.619680404663086, 0.8422750234603882]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142a91350>: [22.782472610473633, 0.8413000106811523]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142a91210>: [18.17073631286621, 0.8747000098228455]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142a91e90>: [17.85953712463379, 0.8637999892234802]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142a91ed0>: [27.1968936920166, 0.7798500061035156]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142a91950>: [20.03327751159668, 0.8478500247001648]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac3c90>: [23.048200607299805, 0.8307999968528748]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac35d0>: [24.574413299560547, 0.8080999851226807]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac3850>: [13.142659187316895, 0.8985000252723694]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac3050>: [15.5021390914917, 0.8804000020027161]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac3d10>: [21.068500518798828, 0.8367249965667725]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac3b10>: [24.773849487304688, 0.8399249911308289]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac3f10>: [16.578609466552734, 0.87642502784729]
Test performance client <shfl.private.federated_operation.FederatedDataNode
object at 0x142ac3110>: [38.41090393066406, 0.7545999884605408]
Global model test performance : [13.033045768737793, 0.8998500108718872]
```

[ ]: