Model Deployment

Serialization and Pickling

dφ

Democratizing Data Science Learning

dφ DPhi

Learning Objectives

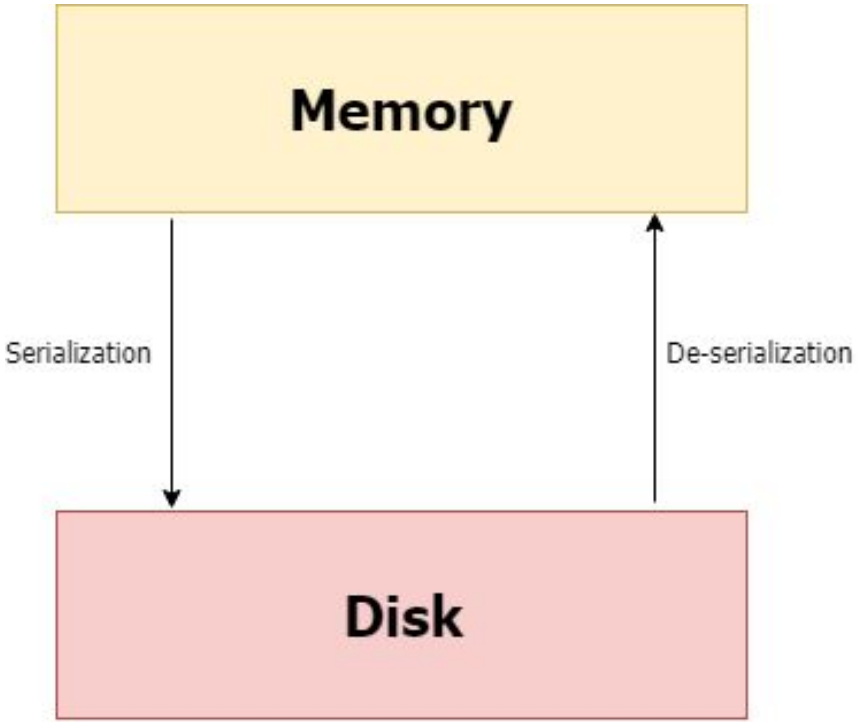
- **Serialization and De-serialization**
- **Pickle Approach**
- **Joblib Approach**
- **Manual Save and Restore to JSON approach**
- **Comparison of the different approaches**
- **Serializing Deep Learning Models**

Serialization and De-serialization

Serialization in machine learning means we're saving the model in a file so that we can:

- reuse it to make predictions,
- compare it with other models,
- or even save the hassle of training a model over and over again.

In simple terms, serialization = saving a model
Deserialization = loading/ reading a model



DPhi

Saving and loading a Machine Learning Model

Saving and loading an ML Model

There are 3 main approaches of Saving and Reloading an ML Model -

1) Pickle Approach

2) Joblib Approach

3) Manual Save and Restore to JSON approach

1. Pickle

Pickle is a commonly used approach for saving a model. The pickle interface provides four methods: dump, dumps, load, and loads.

- The dump() method serializes to an open file (file-like object).
- The dumps() method serializes to a string.
- The load() method deserializes from an open file-like object.
- The loads() method deserializes from a string.

Don’t get confused with the terminologies at the moment, let’s dive into implementation and it’ll get easier.

STEP 1: Import pickle Package

```
import pickle
```

Pickle

STEP 2: Save the Model to file in the current working directory (Pickling)

Let’s name our saved file ‘Pickle_RL_Model.pkl’. You can also add any other path where you wish to save the file.

To open the file for writing, simply use the open() function.

- The first argument should be the name of your file.
- The second argument is 'wb'.

  The w means that you'll be writing to the file, and b refers to binary mode.
  This means that the data will be written in the form of byte objects. If you forget the b, a “TypeError: must be str, not bytes” will be returned. You may sometimes come across a slightly different notation; w+b, but don't worry, it provides the same functionality.

Pickle

Once the file is opened for writing, you can use pickle.dump(), which takes two arguments: the object you want to pickle and the file to which the object has to be saved. In this case, the former will be our model - 'LR_Model', while the latter will be 'file'.

```
Pkl_Filename = "Pickle_RL_Model.pkl"

with open(Pkl_Filename, 'wb') as file:
    pickle.dump(LR_Model, file)
```

Now, a new file named 'Pickle_RL_Model.pkl' should have appeared in the same
directory as your Python script/notebook.

Pickle

STEP 3: Load the Model back from file (Unpickling)

The process of loading a pickled file back into a Python program is similar to the one you saw previously:

Use the open() function again, but this time with 'rb' as second argument (instead of wb). The r stands for read mode and the b stands for binary mode. You'll be reading a binary file. Assign this to ‘file’. Next, use pickle.load(), with ‘file’ as argument, and assign it to ‘Pickled_LR_Model’.

```
with open(Pkl_Filename, 'rb') as file:
    Pickled_LR_Model = pickle.load(file)

Pickled_LR_Model
```

```
LogisticRegression(C=0.1, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=20,
                   multi_class='warn', n_jobs=3, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)
```

Pickle

STEP 4: Use the Reloaded Model to calculate the accuracy score and predict target values

Now, you can simply use your model like you did before! Here, we have used the score() method to find out model's score but you can use any other function for evaluation metric like accuracy_score()

```
# Calculate the Score
score = Pickled_LR_Model.score(Xtest, Ytest)
# Print the Score
print("Test score: {0:.2f} %".format(100 * score))

# Predict the Labels using the reloaded Model
Ypredict = Pickled_LR_Model.predict(Xtest)

Ypredict
```

```
Test score: 91.11 %


array([2, 0, 2, 2, 2, 2, 2, 0, 0, 2, 0, 0, 0, 2, 2, 0, 1, 0, 0, 2, 0, 2,
       1, 0, 0, 0, 0, 0, 0, 2, 2, 0, 2, 0, 1, 2, 2, 1, 1, 0, 2, 0, 1, 0,
       2])
```

DPhi

Dealing with large files - Compressing pickle files

If you are saving a large dataset and your pickled file takes up a lot of space, you may want to compress it. (Not necessarily in our example)

This can be done using bzip2 or gzip. They both compress files, but bzip2 is a bit slower. gzip, however, produces files about twice as large as bzip2.

Remember that compression and serialization is not the same!

Dealing with large files - Compressing pickle files

You can start by importing bzip2 with “import bz2”. Importing pickle is done the same way as before.

Here, we are taking the example of a model called dogs_dict that we wish to compress and store in a file named smallerfile.

```
import bz2
import pickle


sfile = bz2.BZ2File('smallerfile', 'w')
pickle.dump(dogs_dict, sfile)
```

A new file named smallerfile should have appeared. Keep in mind that the difference in file size compared to an uncompressed version will not be noticeable with small object structures.

Pickle

PROs of Pickle :

1) Saving and restoring our learning models is quick - we can do it in two lines of code.

2) It is useful if you have optimized the model's parameters on the training data, so you don't need to repeat this step again.

CONs of Pickle :

1) It doesn't save the test results or any data.

2) Lambda functions can't be pickled. So if you try to apply it to a lambda function, it will fail.

There is a solution for this. dill is a package similar to pickle that can serialize lambda functions, among other things. Its use is almost identical to pickle.

When Not To Use pickle

If you want to use data across different programming languages, pickle is not recommended. Its protocol is specific to Python, thus, cross-language compatibility is not guaranteed.

The same holds for different versions of Python itself. Unpickling a file that was pickled in a different version of Python may not always work properly, so you have to make sure that you're using the same version and perform and update if necessary.

You should also try not to unpickle data from an untrusted source. Malicious code inside the file might be executed upon unpickling.

2. Joblib

The Joblib Module is available from Scikit Learn package and is intended to be a replacement for Pickle, **for objects containing large data**. It provides utilities for saving and loading Python objects that make use of NumPy data structures, efficiently.

This approach will save our ML Model in the pickle format only but we don't need to load additional libraries as the 'Pickling' facility is available within Scikit Learn package itself which we will use invariably for developing our ML models.

Step 1: Import Joblib Module from Scikit Learn

```
from sklearn.externals import joblib
```

Step 2: Save model (LR_Model here) to file in the current working directory(joblib_file)

```
joblib_file = "joblib_RL_Model.pkl"
joblib.dump(LR_Model, joblib_file)
```

```
['joblib_RL_Model.pkl']
```

DPhi

Joblib

Step 3: Reload the saved Model using Joblib

```
joblib_LR_model = joblib.load(joblib_file)
```

Step 4: Use the Reloaded Joblib Model to calculate the accuracy score and predict target values

```
# Calculate the Score
score = joblib_LR_model.score(Xtest, Ytest)
# Print the Score
print("Test score: {0:.2f} %".format(100 * score))

# Predict the Labels using the reloaded Model
Ypredict = joblib_LR_model.predict(Xtest)

Ypredict
```

```
Test score: 91.11 %


array([2, 0, 2, 2, 2, 2, 2, 0, 0, 2, 0, 0, 0, 2, 2, 0, 1, 0, 0, 2, 0, 2,
       1, 0, 0, 0, 0, 0, 0, 2, 2, 0, 2, 0, 1, 2, 2, 1, 1, 0, 2, 0, 1, 0,
       2])
```

DPhi

Joblib

PROs of Joblib :

1) The Joblib library offers a bit simpler workflow compared to Pickle.

2) While Pickle requires a file object to be passed as an argument, Joblib works with both file objects and string filenames.

3) In case our model contains large arrays of data, each array will be stored in a separate file, but the save and restore procedure will remain the same.

4) Joblib also allows different compression methods, such as 'zlib', 'gzip', 'bz2', and different levels of compression.

3. Manual Save and Restore to JSON

Whenever we want to have full control over the save and restore process, the best way is to build our own functions manually.

JSON stands for JavaScript Object Notation. It's a lightweight format for data-interchange, that is easily readable by humans.

The text in JSON is done through quoted-string which contains value in key-value mapping within { }. It is similar to the dictionary in Python. Look at the example of a JSON on the right. Doesn't it look familiar?

```
{
    "people1": [
        {
            "name": "Nikhil",
            "website": "gfg.com",
            "from": "Delhi"
        },
        {
            "name": "Abhinav",
            "website": "google.com",
            "from": "Mumbai"
        }
    ],
    "people2": [
        {
            "name": "Anshul",
            "website": "apple.com",
            "from": "Chennai"
        }
    ]
}
```

We will be dealing with some JSON objects in this tutorial. That is why it is good to get familiar with them.
However, we'll not be saving a model with JSON since its a comparatively longer method and out of the scope of the basic and most used techniques we wish to cover at the moment.

A comparison of the available methods

- The pickle API can be used for serializing standard Python objects.
- The joblib API can be used for efficiently serializing large Python objects with NumPy arrays.
- Although it was derived from JavaScript, JSON is standardized and language-independent. This is a serious advantage over pickle. It's also more secure and much faster than pickle.
- However, if you only need to use Python, then the pickle module is still a good choice for its ease of use and ability to reconstruct complete Python objects.
- An alternative is cPickle. It is nearly identical to pickle, but written in C, which makes it up to 1000 times faster. For small files, however, you won't notice the difference in speed. Both produce the same data streams, which means that Pickle and cPickle can use the same files.

Saving and loading a Deep Learning Model

Saving and loading a Deep Learning Model

A Keras model consists of multiple components:

1. An architecture, or configuration, which specifies what layers the model contain, and how they're connected.

2. A set of weights values (the "state of the model").

3. An optimizer (defined by compiling the model).

4. A set of losses and metrics (defined by compiling the model or calling add_loss() or add_metric()).

Saving and loading a Deep Learning Model

The Keras API makes it possible to save all of these pieces to disk at once, or to only selectively save some of them:

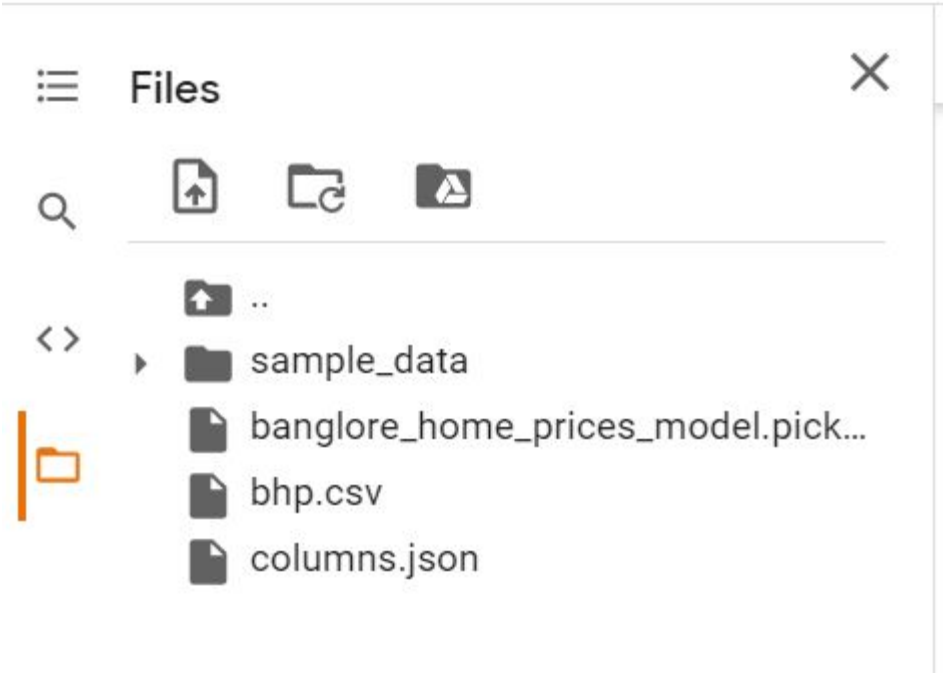
1. Saving everything into a single archive in the TensorFlow SavedModel format (or in the older Keras H5 format). This is the standard practice.
2. Saving the architecture / configuration only, typically as a JSON file.
3. Saving the weights values only. This is generally used when training the model.

Feel free to go through this resource to learn more about saving Deep Learning models:
https://www.tensorflow.org/guide/keras/save_and_serialize#weights_only_saving_in_savedmodel_format

Where to find the saved files

After following the steps in the notebook, a pickle file and a json file will be saved.

- If you're working with Jupyter Notebook/ Python file, you'll find the files in the same folder where your notebook is present.
- If you're working with Google Colab, you'll be able to locate the files on the left side of Colab Notebook. Please download them from there as we'll be working locally from now on.



DPhi

References

- https://www.kaggle.com/prmohanty/python-how-to-save-and-load-ml-models
- https://www.datacamp.com/community/tutorials/pickle-python-tutorial
- https://code.tutsplus.com/tutorials/serialization-and-deserialization-of-python-objects-part-1--cms-26183#:~:text=It%20is%20a%20native%20Python,dumps%2C%20load%2C%20and%20loads.

DPhi

Slide Download Link

You can download this unit from the below link:

https://docs.google.com/presentation/d/1xJVS0TUSQNfR8okbePvjp9kaQ20xKSz5wZwqiAn-c8E/edit?usp=sharing

DPhi

That’s it for this unit. Thank you!

Feel free to post any queries on Discuss.

DPhi