Deeplearning4j - 使用nd4j导入tensorflow模型

在dl4j-example里面新增了模型导入的例子，这里简单的说一下。

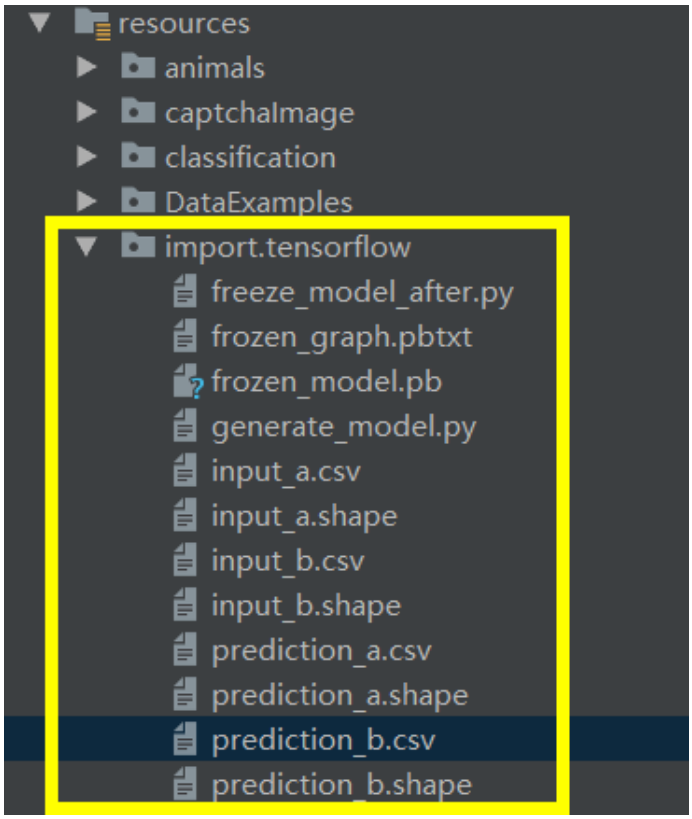
在dl4j新版本的特性介绍：https://github.com/deeplearning4j/deeplearning4j-docs/blob/releasenotes_100a/releasenotes.md 中，对于nd4j的模型导入进行了特别强调。

```
ND4J: New Features

Technology preview of tensorflow import added (supports 1.4.0 and up)
```

其中一项就是对于tf模型的导入提供了功能预览版本，所支持的tf版本为1.4版本及其以上。

并于最近增加了导入tensorflow模型的示例代码，导入模型为 MINST 手写数字分类模型。从代码注释上面来看，因为是预览版本，目前只支持cpu运行，还不支持gpu的加速。



并且提供了如上的文件，用于本次示例的测试。

1. freeze_model_after.py 和 generate_model.py 是生成模型的 python 文件。
2. frozen_model.pb 为tensorflow的模型文件

3. input_*.csv 和 input_*.shape为配套的特征数据文件。csv文件存放的是特征数据，一个特征一行；shape文件保存的是输入模型时的形状。
4. prediction 文件同理，为预测的标签数据文件。

注： 使用该示例的时候，最好 IDE 已经安装了的相对的 lombok 插件。

资源文件夹

```
1.  //Python code for this can be found in resources/import/tensorflow under
    generate_model.py and freeze_model_after.py
2.  //Input node/Placeholder in this graph is names "input"
3.  //Output node/op in this graph is names "output"
4.  public final static String BASE_DIR = "import/tensorflow";
```

首先定义了一个根目录用于寻找对应的文件。

接下来在主函数中获取模型文件的绝对路径

```
1.  final String FROZEN_MLP = new ClassPathResource(BASE_DIR + "/frozen_model.pb").
    getFile().getPath();
```

读取tf中的占位符输入

```
1.  //Load placeholder inputs and corresponding predictions generated from
    tensorflow
2.  Map<String, INDArray> inputsPredictions = readPlaceholdersAndPredictions();
```

这里面所用`readPlaceholdersAndPredictions`方法的全部代码如下：

```
1.  //A simple helper function to load the inputs and corresponding outputs
    generated from tensorflow
2.  //Two cases: {input_a,prediction_a} and {input_b,prediction_b}
3.  protected static Map<String, INDArray> readPlaceholdersAndPredictions() throws
    IOException {
4.      String[] toReadList = {"input_a", "input_b", "prediction_a", "prediction_b"
    };
5.      Map<String, INDArray> arraysFromPython = new HashMap<>();
6.      for (int i = 0; i < toReadList.length; i++) {
7.          String varShapePath = new ClassPathResource(BASE_DIR + "/" + toReadList
```

```
[i] + ".shape").getFile().getPath();
8.          String varValuePath = new ClassPathResource(BASE_DIR + "/" + toReadList
[i] + ".csv").getFile().getPath();
9.          int[] varShape = Nd4j.readNumpy(varShapePath, ",").data().asInt();
10.         float[] varContents = Nd4j.readNumpy(varValuePath).data().asFloat();
11.         arraysFromPython.put(toReadList[i], Nd4j.create(varContents).reshape(va
rShape));
12.     }
13.     return arraysFromPython;
14. }
```

这段代码不难理解，就是把前缀为`toReadList`数组内容中的数据成对读取出来，并且转换成为INDArray对象，并且返回回去。

模型读取

```
1.  //Load the graph into samediff
2.  val graph = TFGraphMapper.getInstance().importGraph(new File(FROZEN_MLP));
```

这里面的 val 并非java 10提供的变量自动推断，而是 lombok 所提供的功能。

数据关联

```
1.  //libnd4j executor
2.  //running with input_a array expecting to get prediction_a
3.  graph.associateArrayWithVariable(inputsPredictions.get("input_a"), graph.variab
leMap().get("input"));
```

这段代码是将从文件中读取出来的 `input_a` INDArray关联模型的数据。

模型预测

```
1.  val executioner = new NativeGraphExecutioner();
2.  val results = executioner.executeGraph(graph); //returns an array of the
outputs
3.  INDArray libnd4jPred = ((INDArray[]) results)[0];
4.  System.out.println("LIBND4J exec prediction for input_a:\n" + libnd4jPred);
```

模型预测，并且获取模型的结果输出。并且将其打印到控制台上。

结果判断

```
if (libnd4jPred.equals(inputsPredictions.get("prediction_a"))) {
    //this is true and therefore predictions are equal
    System.out.println("Predictions are equal to tensorflow");
} else {
    throw new RuntimeException("Predictions don't match!");
}
```

用于判断结果预测，和所给的标签是否相同

使用不同的API用于预测 input_b 的值

```
//Now to run with the samediff executor, with input_b array expecting to get p
rediction_b
val graphSD = TFGraphMapper.getInstance().importGraph(new File(FROZEN_MLP)); //
Reimport graph here, necessary for the 1.0 alpha release
graphSD.associateArrayWithVariable(inputsPredictions.get("input_b"),
graph.variableMap().get("input"));
INDArray samediffPred = graphSD.execAndEndResult();
System.out.println("SameDiff exec prediction for input_b:\n" + samediffPred);
if (samediffPred.equals(inputsPredictions.get("prediction_b"))) {
    //this is true and therefore predictions are equal
    System.out.println("Predictions are equal to tensorflow");
}
```

对模型进行新增op

```
//add to graph to demonstrate pytorch like capability
System.out.println("Adding new op to graph..");
SDVariable linspaceConstant = graphSD.var("linspace", Nd4j.linspace(1, 10, 10))
;
SDVariable totalOutput = graphSD.getVariable("output").add(linspaceConstant);
INDArray totalOutputArr = totalOutput.eval();
System.out.println(totalOutputArr);
```

这个代码的意思就是对原有模型添加新的操作。

1. 首先使用`graphSD.var("linspace", Nd4j.linspace(1, 10, 10))`获取[1,2,3 ... 10]，10个整数的向量
2. `graphSD.getVariable("output").add(linspaceConstant);`将这个向量加入到模型的输出中。

整体输出

```
LIBND4J exec prediction for input_a:
[[         0,         0,         0,         0,         0,         0,         0,
1.0000,         0,         0]]
Predictions are equal to tensorflow
22:39:16,498 WARN  ~ No input found for Add and op name mmul
22:39:16,498 WARN  ~ No input found for Add_1 and op name mmul
SameDiff exec prediction for input_b:
[[         0,         0,    1.0000,         0,         0,         0,         0,
0,         0,         0]]
Predictions are equal to tensorflow
Adding new op to graph..
[[    1.0000,    2.0000,    4.0000,    4.0000,    5.0000,    6.0000,    7.0000,
8.0000,    9.0000,   10.0000]]
```

在最后我们可以看到，因为增加了新的op操作， 模型的原本输出`[[ 0, 0, 1.0000, 0, 0, 0, 0, 0, 0, 0]]` 加上了`[1,2,3 ... 10]`就会变成对应的`[[ 1.0000, 2.0000, 4.0000, 4.0000, 5.0000, 6.0000, 7.0000, 8.0000, 9.0000, 10.0000]]`。

更多文档可以查看 https://github.com/sjsdfg/deeplearning4j-issues。
你的star是我持续分享的动力

代码地址已经放在github上面，自行下载即可： https://github.com/sjsdfg/dl4j-tutorials