修能

TensorFlow框架(5)之机器学习实践

1. Iris data set

Iris数据集是常用的分类实验数据集，由Fisher, 1936收集整理。Iris也称鸢尾花卉数据集，是一类多重变量分析的数据集。数据集包含150个数据集，分为3类，每类50个数据，每个数据包含4个属性。可通过花萼长度，花萼宽度，花瓣长度，花瓣宽度4个属性预测鸢尾花卉属于（Setosa，Versicolour，Virginica）三个种类中的哪一类。

该数据集包含了5个属性：

- Sepal.Length（花萼长度），单位是cm;
- Sepal.Width（花萼宽度），单位是cm;
- Petal.Length（花瓣长度），单位是cm;
- Petal.Width（花瓣宽度），单位是cm;
- species (种类)：Iris Setosa（山鸢尾）、Iris Versicolour（杂色鸢尾），以及Iris Virginica（维吉尼亚鸢尾）。

< 2017年12月 >

日	一	二	三	四	五	六
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

如表 11所示的iris部分数据集。

表 11

6.4	2.8	5.6	2.2	2
5	2.3	3.3	1	1
4.9	2.5	4.5	1.7	2
4.9	3.1	1.5	0.1	0
5.7	3.8	1.7	0.3	0
4.4	3.2	1.3	0.2	0
5.4	3.4	1.5	0.4	0
6.9	3.1	5.1	2.3	2
6.7	3.1	4.4	1.4	1
5.1	3.7	1.5	0.4	0
5.2	2.7	3.9	1.4	1
6.9	3.1	4.9	1.5	1
5.8	4	1.2	0.2	0
5.4	3.9	1.7	0.4	0

我的标签

iOS(31)

AI(10)

C++(10)

CUDA(9)

Objective-C(8)

Spark(8)

TensorFlow(7)

Linux(5)

scikit-learn(2)

Python(1)

更多

随笔分类

Artificial Intelligence(5)

7.7	3.8	6.7	2.2	2
6.3	3.3	4.7	1.6	1

2. Neural Network

2.1 Perform

TensorFlow提供一个高水平的机器学习 API (tf.contrib.learn)，使得容易配置(configure)、训练(train)和评估(evaluate)各种机器学习模型。tf.contrib.learn库的使用可以概括为五个步骤，如下所示：

1) Load CSVs containing Iris training/test data into a TensorFlow Dataset

2) Construct a neural network classifier

3) Fit the model using the training data

4) Evaluate the accuracy of the model

5)Classify new samples

2.2 Code

本节以对 Iris 数据集进行分类为例进行介绍，如下所示是完整的TensorFlow程序：

```
from __future__ import absolute_import

from __future__ import division

from __future__ import print_function


import os

import urllib
```

C++(6)

CUDA(1)

iOS(39)

Linux(2)

Spark(8)

TensorFlow(6)

阅读排行榜

1. CUDA与VS2013安装(2785)

2. Linux平台的boost安装全解(2692)

3. iOS UIKit：TableView之编辑模式（3）(1637)

4. iOS 网络编程：socket(1066)

5. iOS 并行编程：GCD Dispatch Sources (912)

```
import numpy as np
import tensorflow as tf

# Data sets
IRIS_TRAINING = "iris_training.csv"
IRIS_TRAINING_URL = "http://download.tensorflow.org/data/iris_training.csv"

IRIS_TEST = "iris_test.csv"
IRIS_TEST_URL = "http://download.tensorflow.org/data/iris_test.csv"

def main():
# If the training and test sets aren't stored locally, download them.
if not os.path.exists(IRIS_TRAINING):
raw = urllib.urlopen(IRIS_TRAINING_URL).read()
with open(IRIS_TRAINING, "w") as f:
f.write(raw)

if not os.path.exists(IRIS_TEST):
raw = urllib.urlopen(IRIS_TEST_URL).read()
with open(IRIS_TEST, "w") as f:
f.write(raw)
```

```
# Load datasets.
training_set = tf.contrib.learn.datasets.base.load_csv_with_header(
filename=IRIS_TRAINING,
target_dtype=np.int,
features_dtype=np.float32)
test_set = tf.contrib.learn.datasets.base.load_csv_with_header(
filename=IRIS_TEST,
target_dtype=np.int,
features_dtype=np.float32)

# Specify that all features have real-value data
feature_columns = [tf.contrib.layers.real_valued_column("", dimension=4)]

# Build 3 layer DNN with 10, 20, 10 units respectively.
classifier = tf.contrib.learn.DNNClassifier(feature_columns=feature_columns,
hidden_units=[10, 20, 10],
n_classes=3,
model_dir="/tmp/iris_model")
# Define the training inputs
def get_train_inputs():
x = tf.constant(training_set.data)
y = tf.constant(training_set.target)
return x, y
```

```
# Fit model.
classifier.fit(input_fn=get_train_inputs, steps=2000)

# Define the test inputs
def get_test_inputs():
x = tf.constant(test_set.data)
y = tf.constant(test_set.target)
return x, y

# Evaluate accuracy.
accuracy_score = classifier.evaluate(input_fn=get_test_inputs,
steps=1)["accuracy"]

print("\nTest Accuracy: {0:f}\n".format(accuracy_score))

# Classify two new flower samples.
def new_samples():
return np.array(
[[6.4, 3.2, 4.5, 1.5],
[5.8, 3.1, 5.0, 1.7]], dtype=np.float32)

predictions = list(classifier.predict(input_fn=new_samples))
```

```
print(
"New Samples, Class Predictions: {}\n"
.format(predictions))


if __name__ == "__main__":
main()
```

3. Analysis

3.1 Load data

对于本文的程序，Iris数据集被分为两部分：

- 训练集：有120个样例，保存在iris_training.csv文件中；
- 测试集：有30个样例，保存在iris_test.csv文件中。

1) import module

首先程序引入必要module，然后定义了数据集的本地路径和网络路径；

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function


import os
import urllib
```

```
import tensorflow as tf

import numpy as np



IRIS_TRAINING = "iris_training.csv"

IRIS_TRAINING_URL = "http://download.tensorflow.org/data/iris_training.csv"



IRIS_TEST = "iris_test.csv"

IRIS_TEST_URL = http://download.tensorflow.org/data/iris_test.csv
```

2) Open File

若本地路径上不存在数据集指定的文件，则通过网上下载。

```
if not os.path.exists(IRIS_TRAINING):

raw = urllib.urlopen(IRIS_TRAINING_URL).read()

with open(IRIS_TRAINING,'w') as f:

f.write(raw)



if not os.path.exists(IRIS_TEST):

raw = urllib.urlopen(IRIS_TEST_URL).read()

with open(IRIS_TEST,'w') as f:

f.write(raw)
```

3) load Dataset

接着将Iris数据集加载到TensorFlow框架中，使其TensorFlow能够直接使用。这其中使用了learn.datasets.base模块的load_csv_with_header()函数。该方法有三个参数:

- filename：指定了CSV文件的名字；

- target_dtype：指定了数据集中目标数据类型，其为numpy datatype类型；
- features_dtype：指定了数据集中特征向量的数据类型，其为numpy datatype类型。

如表 11所示，Iris数据中的目标值为：0~2，所以可以定义为整型数据就可以了，即np.int，如下所示：

```
# Load datasets.
training_set = tf.contrib.learn.datasets.base.load_csv_with_header(
filename=IRIS_TRAINING,
target_dtype=np.int,
features_dtype=np.float32)
test_set = tf.contrib.learn.datasets.base.load_csv_with_header(
filename=IRIS_TEST,
target_dtype=np.int,
features_dtype=np.float32)
```

由于tf.contrib.learn中的数据类型（Datasets）是以元祖类型定义的，所以用户可以通过data 和 target两个域属性访问特征向量数据和目标数据。即training_set.data 和 training_set.target为训练数据集中的特征向量和目标数据。

3.2 Construct Estimator

tf.contrib.learn预定义了许多模型，称为：**Estimators**。用户以黑箱模型使用Estimator来训练和评估数据。本节使用tf.contrib.learn.DNNClassifier来训练数据，如下所示：

```
# Specify that all features have real-value data
feature_columns = [tf.contrib.layers.real_valued_column("", dimension=4)]


# Build 3 layer DNN with 10, 20, 10 units respectively.
```

```
classifier = tf.contrib.learn.DNNClassifier(feature_columns=feature_columns,

hidden_units=[10, 20, 10],

n_classes=3,

model_dir="/tmp/iris_model")
```

首先程序定义了模型的feature columns，其指定了数据集中特征向量的数据类型。每种类型都有一个名字，由于本节的数据是实数型，所以这里使用.real_valued_column类型。该类型第一个参数指定了列名字，第二个参数指定了列的数量。其中所有的特征类型都定义在：tensorflow/contrib/layers/python/layers/feature_column.py.

然后程序创建了DNNClassifier模型，

- feature_columns=feature_columns：指定所创建的特征向量类型；
- hidden_units=[10, 20, 10]：设置隐藏层的层数，并指定每层神经元的数据量；
- n_classes=3：指定目标类型的数量，Iris数据有三类，所以这里为3；
- model_dir=/tmp/iris_model：指定模型在训练期间保存的路径。

3.3 Describe pipeline

TensorFlow框架的数据都是以Tensor对象存在，即要么是constant、placeholder或Variable类型。通常训练数据是以placeholder类型定义，然后用户训练时，传递所有的数据。本节则将训练数据存储在constant类型中。如下所示：

```
# Define the training inputs

def get_train_inputs():

x = tf.constant(training_set.data)

y = tf.constant(training_set.target)



return x, y
```

3.4 Fit DNNClassifier

创建分类器后，就可以调用神经网络中DNNClassifier模型的**fit()**函数来训练模型了，如下所示：

```
# Fit model.

classifier.fit(input_fn=get_train_inputs, steps=2000)
```

通过向fit传递get_train_inputs函数返回的训练数据，并指定训练的步数为2000步。

3.5 Evaluate Model

训练模型后，就可以通过**evaluate()**函数来评估模型的泛化能力了。与fit函数类似，evaluate函数的输入数据也需为Tensor类型，所以定义了get_test_inputs()函数来转换数据。

```
# Define the test inputs

def get_test_inputs():

x = tf.constant(test_set.data)

y = tf.constant(test_set.target)

return x, y



# Evaluate accuracy.

accuracy_score = classifier.evaluate(input_fn=get_test_inputs, steps=1)["accuracy"]



print("\nTest Accuracy: {0:f}\n".format(accuracy_score))
```

注意：

由于evaluate函数的返回值是一个Map类型（即dict类型），所以直接根据"accuracy"键获取值：accuracy_score。

3.6 Classify Samples

在训练模型后，就可以使用estimator模型的**predict()函数**来预测样例。如表 31有所示的两个样例，希望预测其为什么类型。

表 31

Sepal Length	Sepal Width	Petal Length	Petal Width
6.4	3.2	4.5	1.5
5.8	3.1	5	1.7

如下所示的程序：

```
# Classify two new flower samples.

def new_samples():

return np.array(

[[6.4, 3.2, 4.5, 1.5],

[5.8, 3.1, 5.0, 1.7]], dtype=np.float32)


predictions = list(classifier.predict(input_fn=new_samples))


print(

"New Samples, Class Predictions: {}\n"

.format(predictions))
```

输出：

```
New Samples, Class Predictions: [1 2]
```

注意：

由于predict()函数执行的返回结果类型是generator。所以上述程序将其转换为一个list对象。

4. Logging and Monitoring

由于TensorFlow的机器学习Estimator是黑箱学习，用户无法了解模型执行发生了什么，以及模型什么时候收敛。所以tf.contrib.learn提供的一个Monitor API，可以帮助用户记录和评估模型。

4.1 Default ValidationMonitor

默认使用fit()函数训练Estimator模型时，TensorFlow会产生一些summary数据到fit()函数指定的路径中。用户可以使用Tensorborad来展示更详细的信息。如图 1所示，执行上述程序DNNClassifier的fit()和evaluate()函数后，默认在TensorBoard页面显示的常量信息。

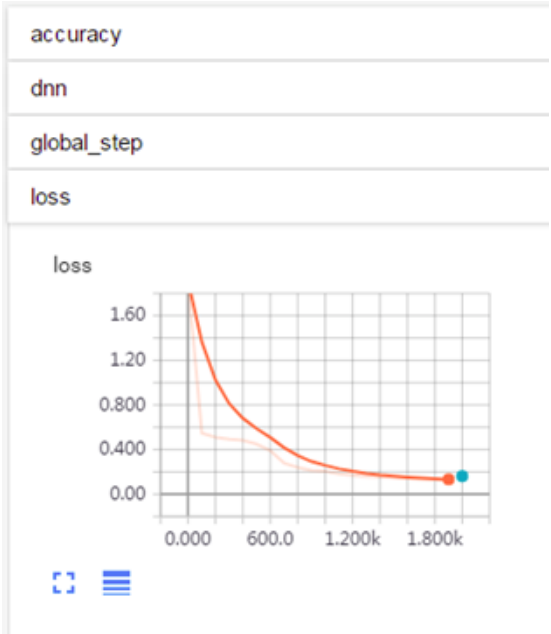

图 1

4.2 Monitors

为了让用户更直观地了解模型训练过程的细节，tf.contrib.learn提供了一些高级Monitors，使得用户在调用fit()函数时，可以使用Monitors来记录和跟踪模型的执行细节。如表 41所示是fitt()函数支持的Monitors类型：

表 41

Monitor	Description
CaptureVariable	每执行n步训练，就将保存指定的变量值到一个集合(collection)中
PrintTensor	每执行n步训练，记录指定的Tensor值
SummarySaver	每执行n步训练，使用tf.summary.FileWriter函数保存tf.Summary 缓存
ValidationMonitor	每执行n步训练，记录一批评估metrics，同时可设置停止条件

如\tensorflow\examples\tutorials\monitors\ iris_monitors.py所示的程序：

```
from __future__ import absolute_import

from __future__ import division

from __future__ import print_function


import os


import numpy as np

import tensorflow as tf
```

```
tf.logging.set_verbosity(tf.logging.INFO)


# Data sets
IRIS_TRAINING = os.path.join(os.path.dirname(__file__), "iris_training.csv")
IRIS_TEST = os.path.join(os.path.dirname(__file__), "iris_test.csv")



def main(unused_argv):
# Load datasets.
training_set = tf.contrib.learn.datasets.base.load_csv_with_header(
filename=IRIS_TRAINING, target_dtype=np.int, features_dtype=np.float)
test_set = tf.contrib.learn.datasets.base.load_csv_with_header(
filename=IRIS_TEST, target_dtype=np.int, features_dtype=np.float)

validation_metrics = {
"accuracy":
tf.contrib.learn.MetricSpec(
metric_fn=tf.contrib.metrics.streaming_accuracy,
prediction_key="classes"),
"precision":
tf.contrib.learn.MetricSpec(
metric_fn=tf.contrib.metrics.streaming_precision,
prediction_key="classes"),
```

```
"recall":
tf.contrib.learn.MetricSpec(
metric_fn=tf.contrib.metrics.streaming_recall,
prediction_key="classes"),
"mean":
tf.contrib.learn.MetricSpec(
metric_fn=tf.contrib.metrics.streaming_mean,
prediction_key="classes")
}
validation_monitor = tf.contrib.learn.monitors.ValidationMonitor(
test_set.data,
test_set.target,
every_n_steps=50,
metrics=validation_metrics,
early_stopping_metric="loss",
early_stopping_metric_minimize=True,
early_stopping_rounds=200)


# Specify that all features have real-value data
feature_columns = [tf.contrib.layers.real_valued_column("", dimension=4)]


# Build 3 layer DNN with 10, 20, 10 units respectively.
classifier = tf.contrib.learn.DNNClassifier(
```

```
feature_columns=feature_columns,
hidden_units=[10, 20, 10],
n_classes=3,
model_dir="/tmp/iris_model",
config=tf.contrib.learn.RunConfig(save_checkpoints_secs=1))

# Fit model.
classifier.fit(x=training_set.data,
y=training_set.target,
steps=2000,
monitors=[validation_monitor])

# Evaluate accuracy.
accuracy_score = classifier.evaluate(
x=test_set.data, y=test_set.target)["accuracy"]
print("Accuracy: {0:f}".format(accuracy_score))

# Classify two new flower samples.
new_samples = np.array(
[[6.4, 3.2, 4.5, 1.5], [5.8, 3.1, 5.0, 1.7]], dtype=float)
y = list(classifier.predict(new_samples))
print("Predictions: {}".format(str(y)))
```

```
if __name__ == "__main__":
tf.app.run()
```

4.3 Configuring ValidationMonitor

如图 1所示，如果没有指定任何evaluation metrics，那么ValidationMonitor默认会记录loss和accuracy信息。但用户可以通过创建ValidationMonitor对象来自定义metrics信息。

即通过向ValidationMonitor构造函数传递一个metrics参数，该参数是一个Map类型(dist)，其中的key是希望显示的名字，value是一个MetricSpec对象。

其中tf.contrib.learn.MetricSpec类的构造函数有如下四个参数：

1. metric_fn：是一个函数，TensorFlow在tf.contrib.metrics模块中预定义了一些函数，用户可以直接使用；
2. prediction_key：如果模型返回一个Tensor或与一个单一的入口，那么这个参数可以被忽略；
3. label_key：可选
4. weights_key：可选

如下所示创建一个dist类型的对象：

```
validation_metrics = {
"accuracy":
tf.contrib.learn.MetricSpec(
metric_fn=tf.contrib.metrics.streaming_accuracy,
prediction_key="classes"),
"precision":
tf.contrib.learn.MetricSpec(
```

```
metric_fn=tf.contrib.metrics.streaming_precision,
prediction_key="classes"),
"recall":
tf.contrib.learn.MetricSpec(
metric_fn=tf.contrib.metrics.streaming_recall,
prediction_key="classes"),
"mean":
tf.contrib.learn.MetricSpec(
metric_fn=tf.contrib.metrics.streaming_mean,
prediction_key="classes")
}
validation_monitor = tf.contrib.learn.monitors.ValidationMonitor(
test_set.data,
test_set.target,
every_n_steps=50,
metrics=validation_metrics,
early_stopping_metric="loss",
early_stopping_metric_minimize=True,
early_stopping_rounds=200)
```

注意：Python中的dist可以直接以一对"{}"初始化元素，如上validation_metrics对象创建所示。

5. 参考文献

[1].TensorFlowà Develop à Get Started àtf.contrib.learn Quickstart；

[2].TensorFlowà Develop à Get Started à Logging and Monitoring Basics with tf.contrib.learn；

分类: Artificial Intelligence, TensorFlow

标签: TensorFlow, AI

好文要顶 关注我 收藏该文

xiuneng
关注 - 0
粉丝 - 26
+加关注

0 0

« 上一篇：TensorFlow框架(4)之CNN卷积神经网络详解
» 下一篇：TensorFlow框架(6)之RNN循环神经网络详解

posted @ 2017-09-01 18:48 xiuneng 阅读(274) 评论(0) 编辑 收藏

刷新评论 刷新页面 返回顶部

最新IT新闻:
· 高铁为什么不用安全带？
· 卡巴斯基实验室创始人卡巴斯基：维护网络安全需要加强国际合作
· 马化腾：中国企业需成为新技术的驱动者和贡献者
· 蚂蚁会员积分将清零！69999积分可换iPhone X
· 微信支付商户平台一大波新功能：手机查看“经营数据”
» 更多新闻...

最新知识库文章:
· 以操作系统的角度述说线程与进程
· 软件测试转型之路
· 门内门外看招聘
· 大道至简，职场上做人做事做管理
· 关于编程，你的练习是不是有效的？
» 更多知识库文章...