全国高校云计算大赛线下课程 TensorFlow 实验手册

实验二 持久化训练手写数字识别程序

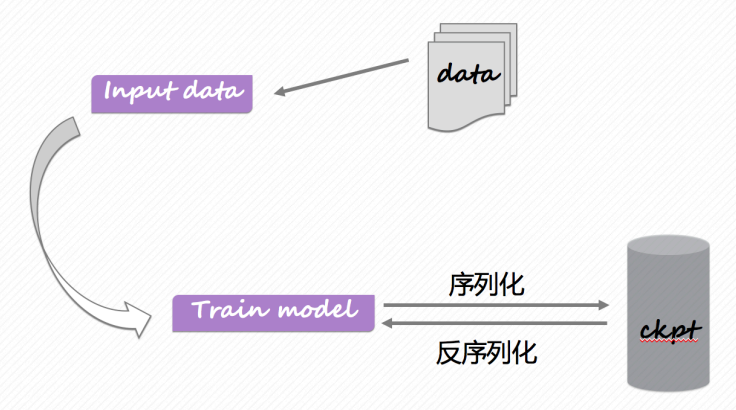
实验目的

1. 掌握持久化训练神经网络原理

实验要求

1. 编写持久化训练手写数字识别程序

实验原理



实验步骤

1. Run a docker contrainer to build our lab environment.

```
# docker run -it --name lab1 slic/tensorflow:1.1
```

2. Coding
   - datasets.py

```
# !/bash/bin/env python
# -*- coding: utf-8 -*-

from tensorflow.examples.tutorials.mnist import input_data

path_to_mnist_data = '/root/data'
mnist = input_data.read_data_sets(path_to_mnist_data, one_hot=True)
```

- graph.py

```
...

# ---------- MODEL checkpoint ----------
MODEL_PATH = './model'
MODEL_NAME = 'model.ckpt'

...
```

- train.py

```
# !/bash/bin/env python
# -*- coding: utf-8 -*-
import os

import tensorflow as tf

from datasets import mnist
from graph import x, y_, TRAINING_STEPS, BATCH_SIZE, train_step, global_step, \
    MODEL_PATH, MODEL_NAME


with tf.Session() as sess:
    # --------- 初始化 tf.train.Saver 类实例 saver 用于保存模型 ----------
    saver = tf.train.Saver()

    # --------- 初始化变量 ---------
    init_op = tf.global_variables_initializer()
    sess.run(init_op)

    validate_feed = {
        x: mnist.validation.images,
        y_: mnist.validation.labels
    }

    test_feed = {
        x: mnist.test.images,
        y_: mnist.test.labels
    }

    for i in range(TRAINING_STEPS+1):
        if i % 1000 == 0:
            # 保存模型

        xs, ys = mnist.train.next_batch(BATCH_SIZE)
        sess.run(train_step,
                 feed_dict={x: xs, y_: ys})
```

- train_eval.py (restore graph from model checkpoint file and calculate validate success)

```
# !/bash/bin/env python
# -*- coding: utf-8 -*-
import time

import tensorflow as tf

from datasets import mnist
from graph import x, y_, accuracy, MODEL_PATH

EVAL_INTERVAL = 5

with tf.Session() as sess:
    # --------- 初始化 tf.train.Saver 类实例 saver 用于读取模型 ----------
    saver = tf.train.Saver()

    validate_feed = {
        x: mnist.validation.images,
        y_: mnist.validation.labels
    }

    while True:
        # restore from checkpoint file

            validate_acc = sess.run(accuracy, feed_dict=validate_feed)
            print("After %s training step(s), validation accuracy "
                  "using average model is %g" % (global_step, validate_acc))

            time.sleep(EVAL_INTERVAL)
        else:
            print('No checkout point file found')
            break
```

3. Run & test

```
# docker exec -it [container id] /bin/bash
# python /path/to/train.py
# python /path/to/train_eval.py
```

全国高校云计算大赛线下课程 TensorFlow 实验手册

实验二 持久化训练手写数字识别程序

实验结果

```
root@2bd7d031d1f2:/tmp/lab2# python train_eval.py
Extracting /root/data/train-images-idx3-ubyte.gz
Extracting /root/data/train-labels-idx1-ubyte.gz
Extracting /root/data/t10k-images-idx3-ubyte.gz
Extracting /root/data/t10k-labels-idx1-ubyte.gz
After 0 training step(s), validation accuracy using average model is 0.087
After 0 training step(s), validation accuracy using average model is 0.087
After 0 training step(s), validation accuracy using average model is 0.087
After 0 training step(s), validation accuracy using average model is 0.087
After 1000 training step(s), validation accuracy using average model is 0.9752
After 1000 training step(s), validation accuracy using average model is 0.9752
After 1000 training step(s), validation accuracy using average model is 0.9752
After 1000 training step(s), validation accuracy using average model is 0.9752
After 2000 training step(s), validation accuracy using average model is 0.9774
After 2000 training step(s), validation accuracy using average model is 0.9774
After 2000 training step(s), validation accuracy using average model is 0.9774
After 3000 training step(s), validation accuracy using average model is 0.982
After 3000 training step(s), validation accuracy using average model is 0.982
After 3000 training step(s), validation accuracy using average model is 0.982
After 3000 training step(s), validation accuracy using average model is 0.982
After 4000 training step(s), validation accuracy using average model is 0.9828
After 4000 training step(s), validation accuracy using average model is 0.9828
After 4000 training step(s), validation accuracy using average model is 0.9828
After 4000 training step(s), validation accuracy using average model is 0.9828
After 5000 training step(s), validation accuracy using average model is 0.9834
After 5000 training step(s), validation accuracy using average model is 0.9834
After 5000 training step(s), validation accuracy using average model is 0.9834
```