Report of Remote Scientific Research Project

MNIST Project

Name：Yunfan Shen

Professor：Fan Zhang

Contents

1 Background

The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

2 Tools

2.1 Docker

Innovation in today's organizations comes from software, where all companies are becoming software companies and need to empower their developers to deliver new customer experiences quickly. Innovation can come in many different application formats - from traditional, monolithic applications to cloud-native and 12-factor applications.

These applications must also be able to run across hybrid/multi-cloud and out to the edge. Docker enables organizations to achieve these goals by providing the only end-to-end (desktop to the data center) experience for developing and scaling distributed applications while leveraging the processes, people and tools that they have in place today. In addition to building and running applications, the Docker Platform provides end-to-edge security at scale, without slowing down innovation with automated governance and compliance throughout the application lifecycle.

2.2 Flask

Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks.

Flask offers suggestions, but doesn't enforce any dependencies or project layout. It is up to the developer to choose the tools and libraries they want to use. There are many extensions provided by the community that make adding new functionality easy.

2.3 Redis

Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs, geospatial indexes with radius queries and streams. Redis has built-in replication, Lua scripting, LRU eviction, transactions and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster.

Redis is a high-performance key-value database, which is completely open source and free, and redis is a NOSQL database, which is a database solution for solving a series of problems such as high concurrency, high expansion, large data storage and so on. It is a non-relational database. However, it can not replace relational databases, but can only be extended in a specific environment.

2.4 Tensorflow

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

Build and train ML models easily using intuitive high-level APIs like Keras with eager execution, which makes for immediate model iteration and easy debugging. Easily train and deploy models in the cloud, on-prem, in the browser, or on-device no matter what language you use. A simple and flexible architecture to take new ideas from concept to code, to state-of-the-art models, and to publication faster.

3 project implementation

3.1 The data sets

The data is stored in a very simple file format designed for storing vectors and multidimensional matrices. General info on this format is given at the end of this page, but you don't need to read that to use the data files.

All the integers in the files are stored in the MSB first (high endian) format used by most non-Intel processors. Users of Intel processors and other low-endian machines must flip the bytes of the header.

There are 4 files:

train-images-idx3-ubyte: training set images

train-labels-idx1-ubyte: training set labels

t10k-images-idx3-ubyte: test set images

t10k-labels-idx1-ubyte: test set labels

The training set contains 60000 examples, and the test set 10000 examples. The first 5000 examples of the test set are taken from the original NIST training set. The last 5000 are taken from the original NIST test set. The first 5000 are cleaner and easier than the last 5000.

Here is structure of the set of label file:

[offset] [type] [value] [description]

0000 32 bit integer 0x00000801(2049) magic number (MSB first)

0004 32 bit integer 60000 number of items

0008 unsigned byte ?? label

0009 unsigned byte ?? label

.....

xxxx unsigned byte ?? label

The labels values are 0 to 9.

Here is the structure of the set of image file:

[offset] [type] [value] [description]

0000 32 bit integer 0x00000803(2051) magic number

0004 32 bit integer 60000 number of images

0008 32 bit integer 28 number of rows

0012 32 bit integer 28 number of columns

0016 unsigned byte ?? pixel

0017 unsigned byte ?? pixel

.....

xxxx unsigned byte ?? pixel

3.2 The codes

I used Multilayer Feed-Forward Neural Network to help train the data set.

```
# 定义前向传播过程，两个参数，一个是输入数据，一个是正则化参数
def forward(x, regularizer):
    # w1的维度就是[输入神经元大小，第一层隐含层神经元大小]
    w1 = get_weight([INPUT_NODE, LAYER_NODE], regularizer)
    # 偏置b参数，与w的后一个参数相同
    b1 = get_bias(LAYER_NODE)
    # 激活函数
    y1 = tf.nn.relu(tf.matmul(x, w1) + b1)

    w2 = get_weight([LAYER_NODE, OUTPUT_NODE], regularizer)
    b2 = get_bias(OUTPUT_NODE)
    y = tf.matmul(y1, w2) + b2

    return y
```

1.Forward propagation

```
def backward(mnist):
    # 一共有多少个特征，784行，一列
    x = tf.placeholder(tf.float32, [None, mnist_forward.INPUT_NODE])
    y_ = tf.placeholder(tf.float32, [None, mnist_forward.OUTPUT_NODE])
    # 给前向传播传入参数x和正则化参数计算出y的值
    y = mnist_forward.forward(x, REGULARIZER)
    # 初始化global－step，它会随着训练轮数增加
    global_step = tf.Variable(0, trainable=False)

    # softmax和交叉商一起运算的函数，logits传入是x*w，也就是y
    ce = tf.nn.sparse_softmax_cross_entropy_with_logits(logits=y, labels=tf.argmax(y_, 1))
    cem = tf.reduce_mean(ce)
    loss = cem + tf.add_n(tf.get_collection("losses"))

    learning_rate = tf.train.exponential_decay(LEARNING_RATE_BASE,
                                               global_step,
                                               mnist.train.num_examples / BATCH_SIZE,
                                               LEARNING_RATE_DECAY,
                                               staircase=True)

    train_step = tf.train.GradientDescentOptimizer(learning_rate).minimize(loss, global_step=global_step)

    # 滑动平均处理，可以提高泛华能力
    ema = tf.train.ExponentialMovingAverage(MOVING_AVERAGE_DECAY, global_step)
    ema_op = ema.apply(tf.trainable_variables())
    # 将train_step和滑动平均计算ema_op放在同一个节点
    with tf.control_dependencies([train_step, ema_op]):
        train_op = tf.no_op(name="train")

    saver = tf.train.Saver()
```

2.Back propagation(1)

```
with tf.Session() as sess:

    init_op = tf.global_variables_initializer()
    sess.run(init_op)

    for i in range(STEPS):
        # mnist.train.next_batch()函数包含一个参数BATCH_SIZE,表示随机从训练集中抽取BATCH_SIZE个样本输入到神经网络
        # next_batch函数返回的是image的像素和标签label
        xs, ys = mnist.train.next_batch(BATCH_SIZE)
        # _,表示后面不使用这个变量
        _, loss_value, step = sess.run([train_op, loss, global_step], feed_dict={x: xs, y_: ys})

        if i % 1000 == 0:
            print("After {} training step(s),loss on training batch is {} ".format(step, loss_value))
            saver.save(sess, os.path.join(MODEL_SAVE_PATH, MODEL_NAME), global_step=global_step)
```

3.Back propagation(2)

Picture preprocessing:

The uploaded image is resized, then transformed into gray image, and the noise is filtered through binary processing. Finally, only black and white pixels are left.

```
def pre_pic(picName):
    # 先打开传入的原始图片
    img = Image.open(picName)
    # 使用消除锯齿的方法resize图片
    reIm = img.resize((28, 28), Image.ANTIALIAS)
    # 变成灰度图，转换成矩阵
    im_arr = np.array(reIm.convert("L"))
    threshold = 50  # 对图像进行二值化处理，设置合理的阈值，可以过滤掉噪声，让他只有纯白色的点和纯黑色点
    for i in range(28):
        for j in range(28):
            im_arr[i][j] = 255 - im_arr[i][j]
            if (im_arr[i][j] < threshold):
                im_arr[i][j] = 0
            else:
                im_arr[i][j] = 255
    # 将图像矩阵拉成1行784列，并将值变成浮点型（像素要求的仕0-1的浮点型输入）
    nm_arr = im_arr.reshape([1, 784])
    nm_arr = nm_arr.astype(np.float32)
    img_ready = np.multiply(nm_arr, 1.0 / 255.0)

    return img_ready
```

4.Picture preprocessing

```python
def application(num,path):
    testNum = num
    re = 999
    # 使用循环来历遍需要测试的图片才结束
    for i in range(testNum):
        # input可以实现从控制台接收字符格式,图片存储路径
        #testPic = input("the path of test picture:")
        testPic = path
        print('test1')
        # 将图片路径传入图像预处理函数中
        testPicArr = pre_pic(testPic)
        print('test2')
        # 将处理后的结果输入到预测函数最后返回预测结果
        preValue = restore_model(testPicArr)
        print("The prediction number is :", preValue)
        re = preValue
    return re
```

5.Test function

After successful upload the file, call the test function, return the results to the front-end display, and upload the file name, test results and the time to redis database.

```python
UPLOAD_FOLDER = './imgs'
ALLOWED_EXTENSIONS = set(['png', 'jpg', 'jpeg'])

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1] in ALLOWED_EXTENSIONS
re = ''
@app.route('/', methods=['GET', 'POST'])
def upload_file():
    global re
    if request.method == 'POST':
        file = request.files['file']
        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
            re = application(1, UPLOAD_FOLDER+'/'+filename)
            redis.sadd(d.datetime.now().strftime("%Y.%m.%d-%H:%M:%S"), d.datetime.now().strftime("%Y.%m.%d-%H:%M:%S"), filename, str(re[0]))
            print(redis.smembers(d.datetime.now().strftime("%Y.%m.%d-%H:%M:%S")))
            return redirect(url_for('upload_file',
                                    filename=filename))
    return render_template('index.html', re=re)
```

6. Flask function

```
<!doctype html>
    <title>Upload pictures</title>
    <h1>Upload pictures</h1>
    <form action="" method=post enctype=multipart/form-data>
      <p><input type=file name=file>
         <input type=submit value=Upload>
    </form>
    <div> The prediction number is : {{re}} </div>
```

7. Front-end code

3.3 docker

Create a file called Dockerfile and a file called requirements. Add the content below to them.

```
# Use an official Python runtime as a parent image
FROM python:3.7

# Set the working directory to /app
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python", "mnist_app.py"]
```

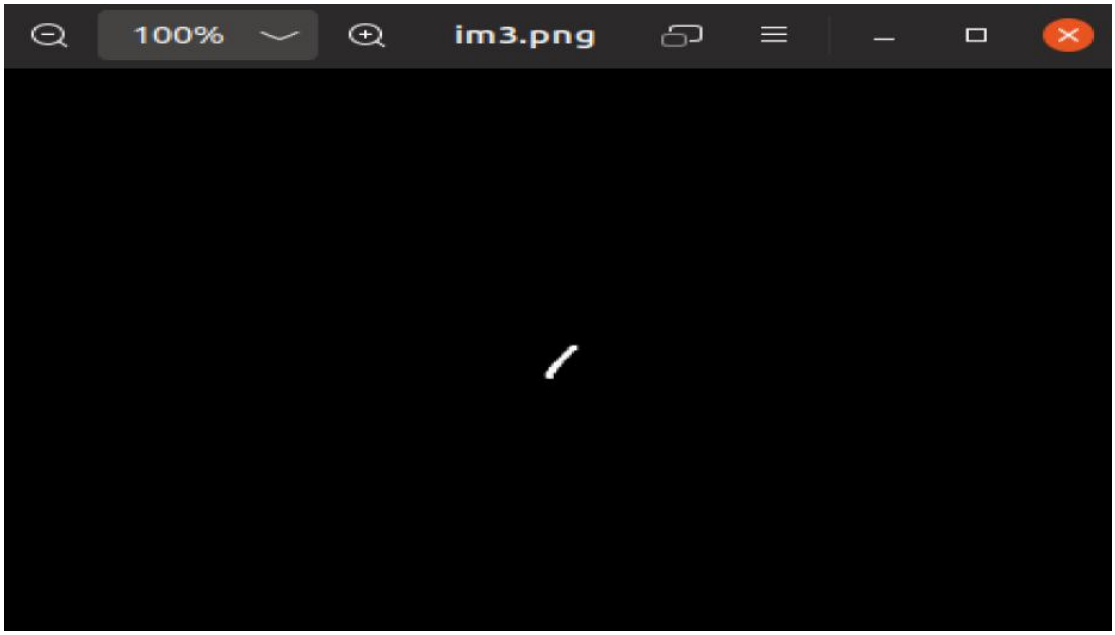
Flask
Redis
tensorflow
numpy
time
Pillow

Then, at the level of my project directory, run the build command:

Docker build --tag=mymnist .

3.4 Results

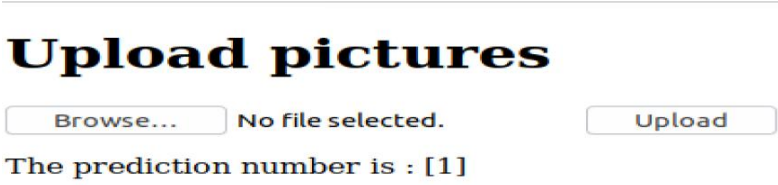
Upload the picture ‘im3.png’, Then the result will show on the website. Run redis in the background the result will be delivery to it at the same time.



8.img3.png

9.website

10.result(1)



11. result(2)

4 Thoughts and acquisition

I have learned some related courses of machine learning and participated in some relevant competitions before, but all I have done is the training data step. I did some work of applying algorithms and adjusting parameters. This project is the first time

that I have completed a large data related app completely. Although it is rough, I have experienced the whole process, including pre-processing, training data sets, writing corresponding front-end and background code, and docker packaging process. I have learned a lot.

First of all, I do not have sufficient time to do this project. Because the deadline of this project is almost the same as my GRE and TOEFL exams, I have to consider three tasks at the same time. The front-end of this project is very simple. I will try to improve this work in the future when I have time.

Then the biggest achievement of this course is that professor showed me the whole ecology of big data area, which gave me a general understanding of this area, rather than just knowing some related algorithms as before. Professor also sincerely recommend to us the future direction of learning. I am so grateful to professor ZhangFan for teaching me so much.

Of course, It also encountered a lot of problems during this time, most of which were solved through online search and teachers' help. But until the end there was a problem that bothered me, that is, docker can not download Python package when it is building. I tried to open VPN, or change the mirror image. However, both of them were in vain. Today, when I was running my program in pycharm, I found that my Python crashed. That's why I couldn't download the python package when I packed it on docker. It may be caused by the crash of my VMware. I again realized the shortcomings of the virtual machine. In contrast, docker is really a good thing.

Finally, I would like to thank Zhang Fan, who guided me in this project, and thank you for helping me solve the problems I met and teaching me so much.