Tech Summit 2018

Hands-on lab

从零开始用 AI

Microsoft

目录

概览

AI 人工智能不仅是近几年火热的技术话题，更是上升为国家层面的技术战略。 ML 机器学习作为这一波 AI 人工智能迅猛发展的引擎，也越来越多为人所熟知。

为了帮助人们方便快捷的使用 ML 机器学习来为现在的业务插上 AI 的翅膀，微软 Azure 云中不仅提供封装的认知服务，还提供了易用的、所见即所得的 Azure Machine Learning Studio 机器学习工作室。

最为重要的是，Azure ML Studio 还提供了免费的版本，而入门的教程如此简单，所以本次动手实验，让我们一起看看，如何借助 Azure ML Studio 实现零基础、零成本的 AI 入门。

目标

在这个动手实验里，您可以体验：

- 获取并使用免费的 Azure Machine Learning Studio 环境
- 使用免费的在线 Azure Jupyter Notebooks
- 尝试创建手写数字识别的机器学习模型
- 使用不同的神经网络来优化手写数字识别机器学习模型

系统需求

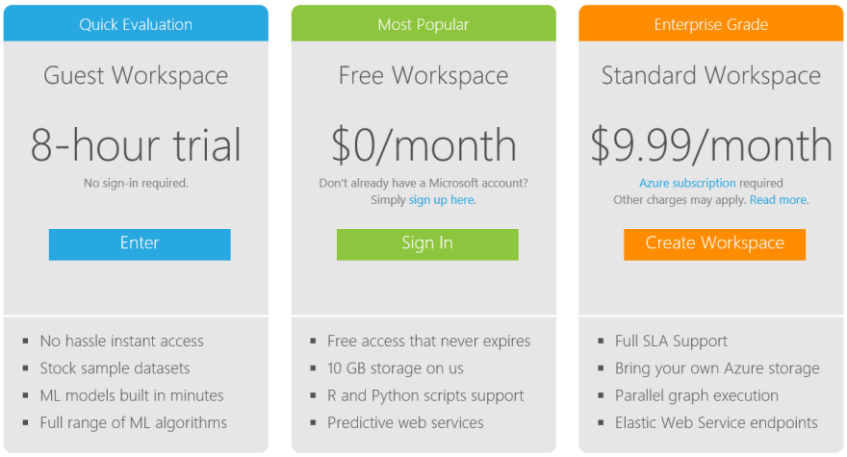
为完成本次动手实验，您需要准备：

- 浏览器，例如 Edge、Internet Explorer 或 Chrome
- 一个 Microsoft Account，例如 hotmail 账号
- Office Excel 软件

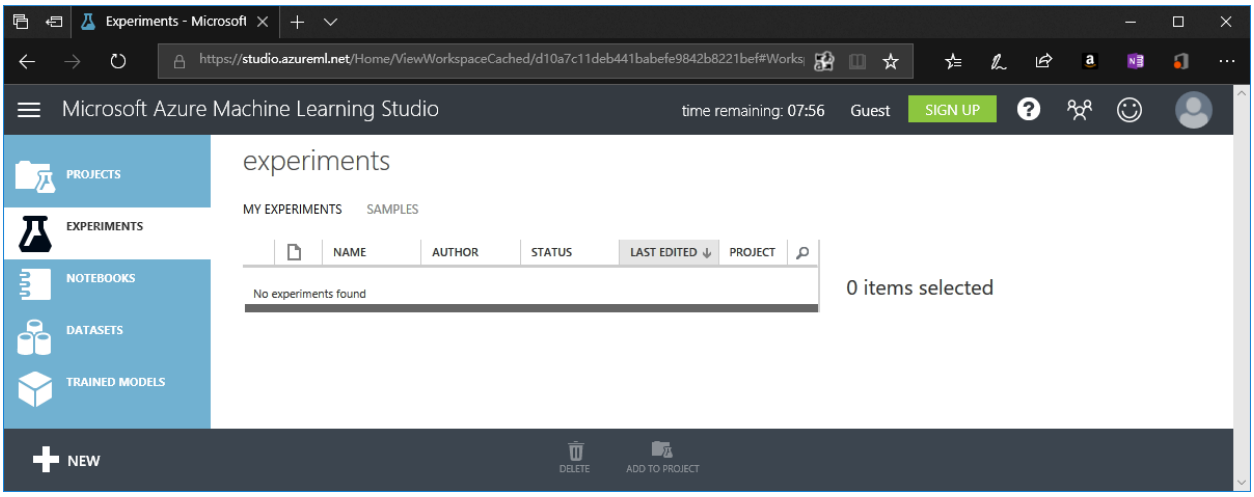
设置

您需要使用以下步骤来设置动手实验环境：

1. 打开浏览器，访问：https://studio.azureml.net/
2. 找到 “Sign up here” 链接，点击



3. 选择 “Free Workspace“ 选项。如果还没有准备好您的 Microsoft Account，可以点击 ” sign up here“ 链接来注册。点击 “Sign In” 按钮，使用您的 Microsoft Account 登录
4. 您应该能够看到类似如下的机器学习工作室界面：



练习

本动手实验包含以下练习：

1. 理解以及处理数据

   查看 MNIST 数据集并理解手写数字图片的数据处理

2. 创建训练学习模型

   创建用于训练机器学习的模型并使用模型进行训练

3. 优化模型并且发布

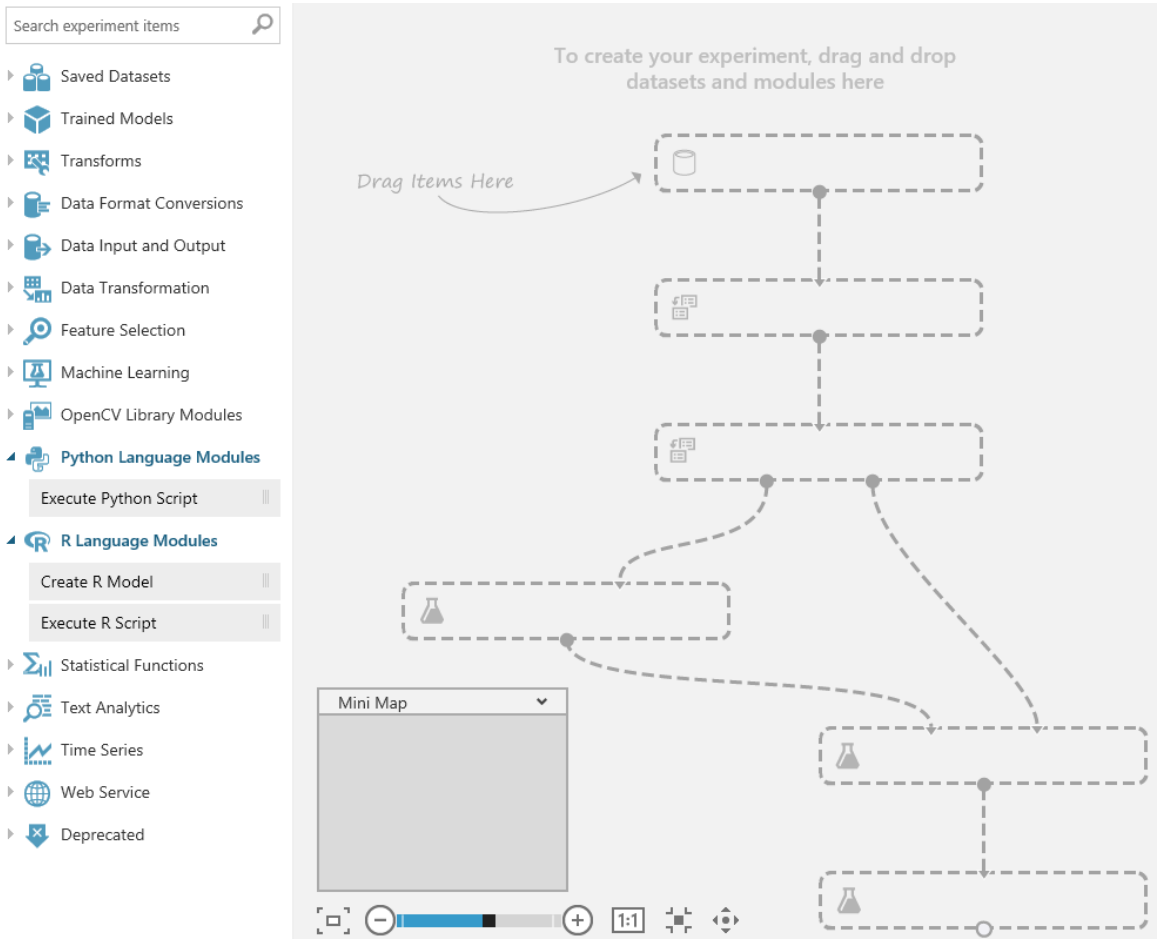
   使用不同架构的神经网络来优化手写数字识别模型，并发布训练好的模型

预计完成动手实验时间： 30 到 45 分钟.

练习 1：理解以及处理数据

任务 1 – 放置并查看数据集

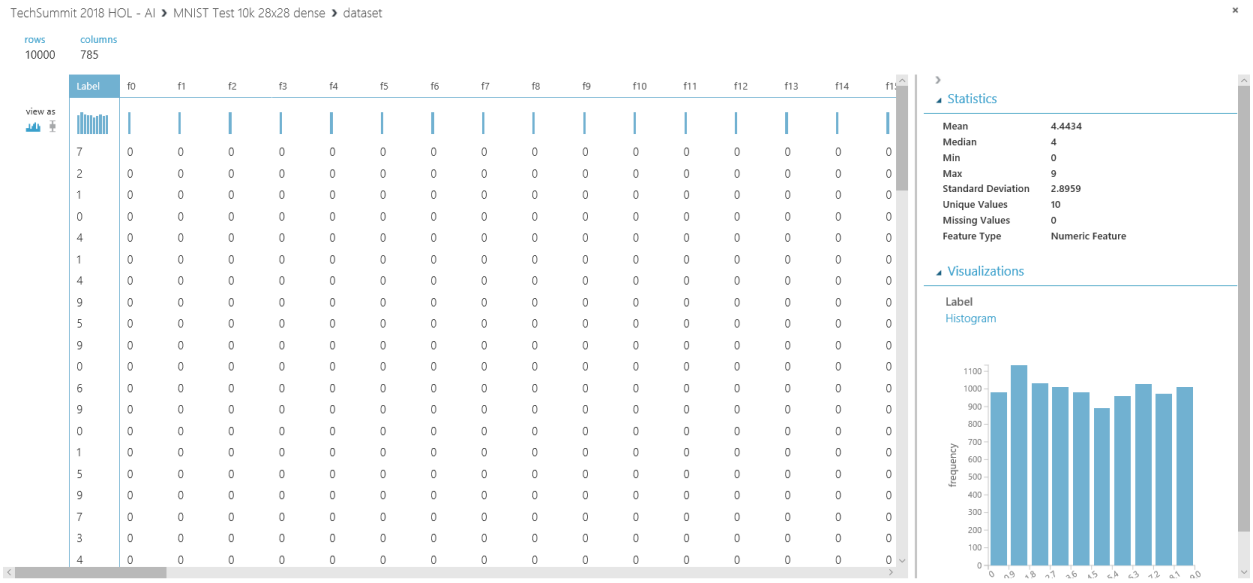
0. 在机器学习工作室下侧工具栏点击“NEW”，新建一个实验，应能看到如下工作区：



1. 在机器学习工作室左侧面板，找到“Saved Dataset”并展开，找到并展开 “Samples” 分支，点击然后找到“MNIST Train 60k 28x28 dense”数据集，拖拽并放置到编排区域

2. 相同位置找到“MNIST Test 10k 28x28 dense”数据集，拖拽并放置到编排区域

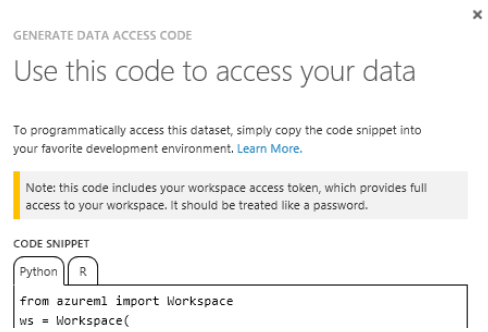
3. 右键点击“MNIST Test 10k 28x28 dense”数据集的输出节点，选择“Visualize”显示数据集内容，应形如下图：



请结合讲师的讲述，了解各字段代表的数据含义。

任务 2 – 将数据转化为图片

1. 右键点击“MNIST Test 10k 28x28 dense”数据集的输出节点，选择“Generate Data Access Code...”，显示形如下图，复制”Python“框内的代码



2. 右键点击“MNIST Test 10k 28x28 dense”数据集的输出节点，选择“Open in a new Notebook”，然后选择“Python 3”，打开用于执行 Python 程序的笔记本

3. 将第一个代码块中的代码，替换为刚才复制的代码

```
In [ ]: from azureml import Workspace

        ws = Workspace()
        ds = ws.datasets['MNIST Test 10k 28x28 dense']
        frame = ds.to_dataframe()
```

4. 点击上方菜单的"Run"按键，运行代码块，然后运行第二个代码块。如果代码运行成功，应看到如下显示，注意观察最后显示的行列数

Out[2]:

	Label	f0	f1	f2	f3	f4	f5	f6	f7	f8	...	f774	f775	f776	f777	f778	f779	f780	f781	f782	f783
0	7	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
⋮																					
9999	6	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

10000 rows × 785 columns

5. 在最下方的代码块中输入以下内容并运行：

```
import numpy as np
data_np_array=frame.values
print(data_np_array)
print(data_np_array.shape)
```

若代码成功运行，将看到如下图所示，注意观察最后显示的数组 Array 维度

In [3]:
```
import numpy as np
data_np_array=frame.values
print(data_np_array)
print(data_np_array.shape)
```

```
[[7 0 0 ..., 0 0 0]
 [2 0 0 ..., 0 0 0]
 [1 0 0 ..., 0 0 0]
 ...,
 [4 0 0 ..., 0 0 0]
 [5 0 0 ..., 0 0 0]
 [6 0 0 ..., 0 0 0]]
(10000, 785)
```

6. 在最下方的代码块中输入以下内容并运行：

```
select_img_array=data_np_array[999,1:]
print(select_img_array)
print(select_img_array.shape)
```

999 为选择提取的图片，可自行更改。如代码运行正常，应能看到如下显示，注意观察最后的数组 Array 维度，思考去掉了哪个数字，为什么去掉？

```
  0   0   0   0   0   0   0   0   0   0   0   0  71 244  33   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0]
(784,)
```

In [ ]:

7. 在最下方的代码块中输入以下内容并运行：

```
img_data=select_img_array.reshape((28,28))
print(img_data)
print(img_data.shape)
```

如代码正确运行，应看到如下显示， 注意观察输出数组维度，思考为何要进行转换？

```
[  0   0   0   0   0   0   0   0   0   0   0   0   0   0  71 244  33   0
   0   0   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0]]
(28, 28)
```

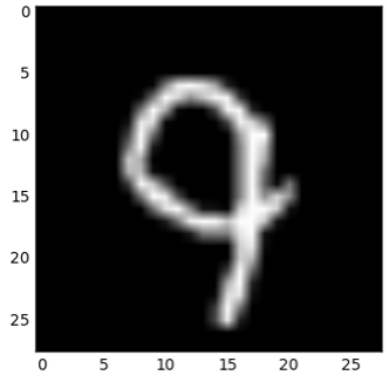
```
In [ ]:
```

8. 在最下方的代码块中输入以下内容并运行：

```
plt.gray()
plt.imshow(img_data)
```

如代码正确运行，应能看到如下显示：

```
Out[8]: <matplotlib.image.AxesImage at 0x7fcde09b8b38>
```



练习 2：创建训练学习模型

任务 1 – 选择机器学习算法

1. 在机器学习工作室左侧面板，找到“Machine Learning”并展开，找到并展开 ”Initialize Model” 分支，点击然后展开 “Classification” 分支，然后找到“Multiclass Neural Network”

2. 拖拽并放置该组件到编排区域，然后按照如下参数进行配置：

Create trainer mode	Single Parameter
Hidden layer specification	Customer definition script
The learning rate	0.01
Number of learning iterations	30
The initial learning weights diameter	0.3
The momentum	0
The type of normalizer	Min-Max normalizer
Random number seed	1

3. 在 “Neural network definition” 编辑框中，输入如下内容：

```
input Picture [28,28];
hidden H1 [200] from Picture all;
hidden H2 [200] from H1 all;
output Result [10] softmax from H2 all;
```

任务 2 – 链接训练模型

1. 在机器学习工作室左侧面板，找到 “Machine Learning” 并展开，找到并展开 ”Train” 分支，然后找到 “Train Model”

2. 拖拽并放置该组件到编排区域，链接上一步配置好的算法组件“Multiclass Neural Network “到该组件的左边输入节点，链接 “MNIST Train 60k 28x28 dense” 数据集到该组件的右边输入节点

3. 点击右侧 “Launch column selector” ，选择 “Label” 列

任务 3 – 链接评分模型

1. 在机器学习工作室左侧面板，找到 “Machine Learning” 并展开，找到并展开 ”Score” 分支，然后找到 “Score Model”

2. 拖拽并放置该组件到编排区域，链接上一步配置好的训练模型组件”Train Model”到该组件的左边输入节点，链接 “MNIST Train 10k 28x28 dense” 数据集到该组件的右边输入节点

3. 确认右侧”Append score column to output”选择框已勾选

任务 4 – 链接评估模型并执行 R 语言脚本

1. 在机器学习工作室左侧面板，找到”Machine Learning”并展开，找到并展开”Evaluate”分支，然后找到”Evaluate Model”

2. 拖拽并放置该组件到编排区域，链接上一步配置好的模型组件”Score Model”到该组件的左边输入节点，空置右边输入节点

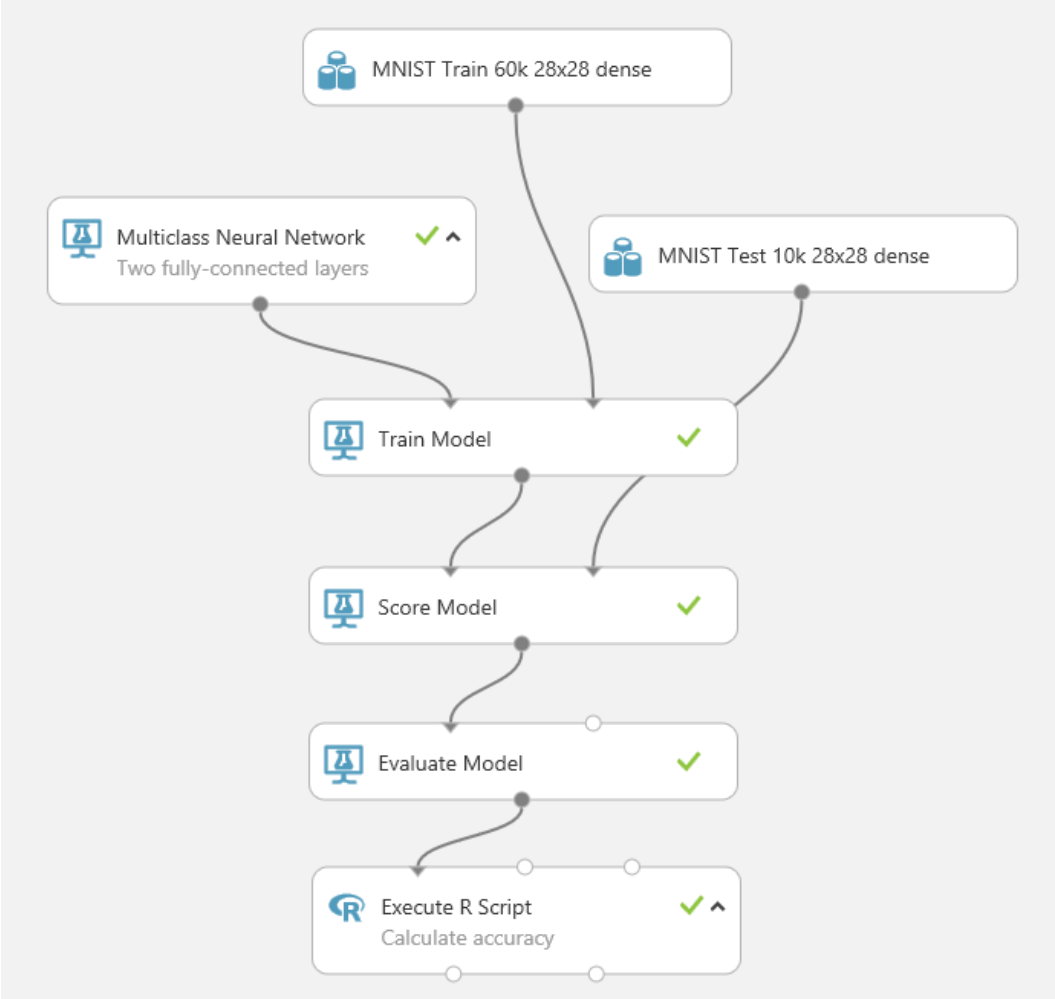
3. 在机器学习工作室左侧面板，找到”R Language Modules”展开，找到”Execute R Script”

4. 拖拽并放置该组件到编排区域，链接上一步配置好的模型组件”Evaluatue Model”到该组件的左边输入节点，空置中间、右边输入节点

5. 在右侧”R Script”输入框中输入如下内容：

```
dataset1 <- maml.mapInputPort(1)
feat <- data.matrix(dataset1[,2:11])
total = sum(sum(feat))
correct = sum(diag(feat))
accuracy = correct / total
acc = as.data.frame(accuracy)
maml.mapOutputPort("acc");
```

6. 查看链接好的所有组件，应如下图所示：



7. 点击机器学习工作室下侧的“RUN”，运行训练过程，并等待其结束

8. 右键点击“Execute R Script”组件的左边输出节点，选择“Visualize”，查看并记录显示的识别准确度 accuracy。

练习 3：优化模型并且发布

任务 1 – 更换神经网络

1. 选中“Multiclass Neural Network”组件，然后按照如下参数修改配置：

Create trainer mode	Single Parameter
Hidden layer specification	Customer definition script
The learning rate	0.01
Number of learning iterations	20
The initial learning weights diameter	0.9
The momentum	0.7
The type of normalizer	Min-Max normalizer
Random number seed	1

2. 在 “Neural network definition” 编辑框中，输入如下内容：

```
const { T = true; F = false; }

const {
// input image size
ImgW = 28;
ImgH = 28;

// first convolutional layer parameters
C1Maps = 5;
C1KernW = 5;
C1KernH = 5;
C1StrideW = 1;
C1StrideH = 1;
// The following formula computes dimensions with padding enabled.
C1OutW = (ImgW - 1) / C1StrideW + 1;
C1OutH = (ImgH - 1) / C1StrideH + 1;

// first pooling layer parameters
P1KernW = 2;
P1KernH = 2;
P1StrideW = 2;
P1StrideH = 2;
// The following formula computes dimensions with no padding.
P1OutW = (C1OutW - P1KernW) / P1StrideW + 1;
P1OutH = (C1OutH - P1KernH) / P1StrideH + 1;
```

```
// second convolutional layer parameters
C2Maps = 10;
C2KernW = 5;
C2KernH = 5;
C2StrideW = 1;
C2StrideH = 1;
// The following formula computes dimensions with padding enabled.
C2OutW = (P1OutW - 1) / C2StrideW + 1;
C2OutH = (P1OutH - 1) / C2StrideH + 1;
// Since Z dimension of the kernel is 1 and sharing is disabled in Z dimension
// total number of maps is a product of input maps and layer maps.
C2OutZ = C2Maps * C1Maps;

// second pooling layer parameters
P2KernW = 2;
P2KernH = 2;
P2StrideW = 2;
P2StrideH = 2;
// The following formula computes dimensions with no padding.
P2OutW = (C2OutW - P2KernW) / P2StrideW + 1;
P2OutH = (C2OutH - P2KernH) / P2StrideH + 1;
}

input Picture [ImgH, ImgW];

hidden C1 [C1Maps, C1OutH, C1OutW]
  from Picture convolve {
    InputShape  = [ImgH, ImgW];
    KernelShape = [C1KernH, C1KernW];
    Stride      = [C1StrideH, C1StrideW];
    Padding     = [T, T];
    MapCount = C1Maps;
  }

hidden P1 [C1Maps, P1OutH, P1OutW]
  from C1 max pool {
    InputShape  = [C1Maps, C1OutH, C1OutW];
    KernelShape = [1, P1KernH, P1KernW];
    Stride      = [1, P1StrideH, P1StrideW];
  }

hidden C2 [C2OutZ, C2OutH, C2OutW]
  from P1 convolve {
    InputShape  = [C1Maps, P1OutH, P1OutW];
    KernelShape = [1, C2KernH, C2KernW];
    Stride      = [1, C2StrideH, C2StrideW];
    Sharing     = [F, T, T];
    Padding     = [F, T, T];
    MapCount = C2Maps;
  }

hidden P2 [C2OutZ, P2OutH, P2OutW]
```

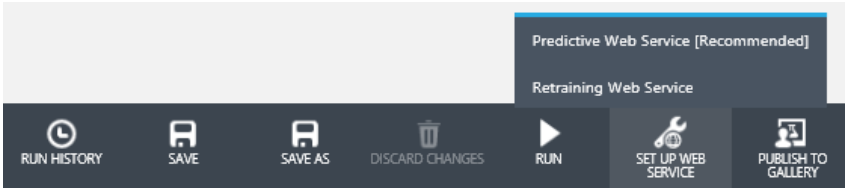
```
  from C2 max pool {
    InputShape  = [C2OutZ, C2OutH, C2OutW];
    KernelShape = [1,  P2KernH, P2KernW];
    Stride      = [1,  P2StrideH, P2StrideW];
  }

hidden H3 [100]
  from P2 all;

output Result [10] softmax
  from H3 all;
```

3. 点击机器学习工作室下侧的“RUN”，运行训练过程，并等待其结束

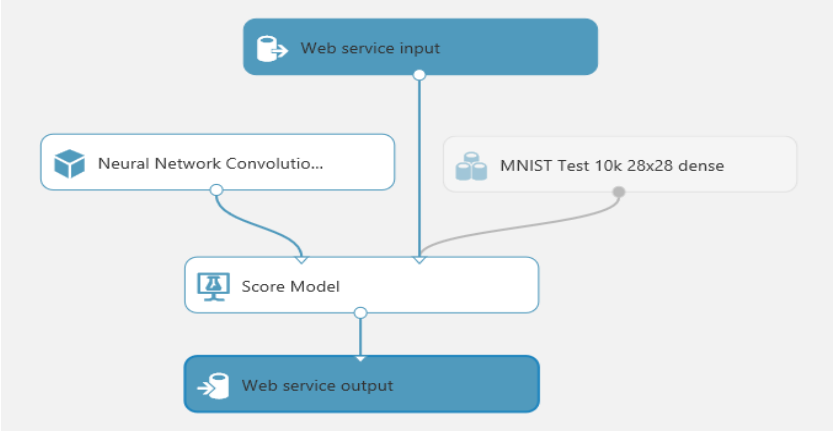
4. 右键点击“Execute R Script”组件的左边输出节点，选择“Visualize”，查看并记录显示的识别准确度 accuracy。

任务 2 – 发布训练模型

5. 点击机器学习工作室下侧的“SET UP WEB SERVICE”，选择“Predictive Web Service”，等待发布过程结束



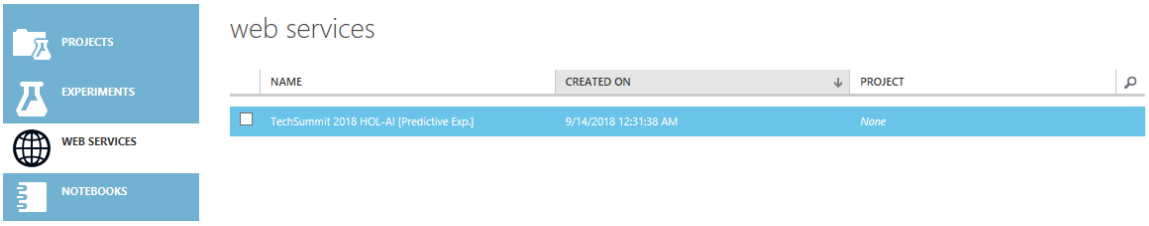
6. 机器学习工作会自动生成预测服务并发布



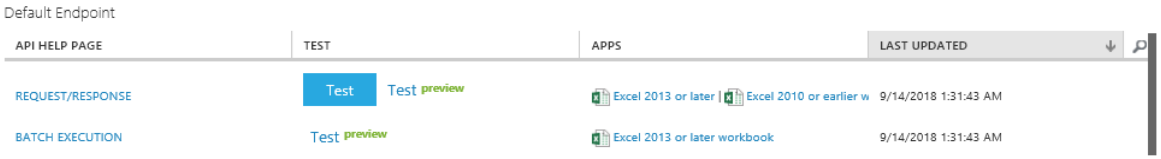
7. 点击机器学习工作室下侧的”RUN”，待模型运行结束后点击“DEPLY WEB SERVICE“

任务 3 – 使用预测服务

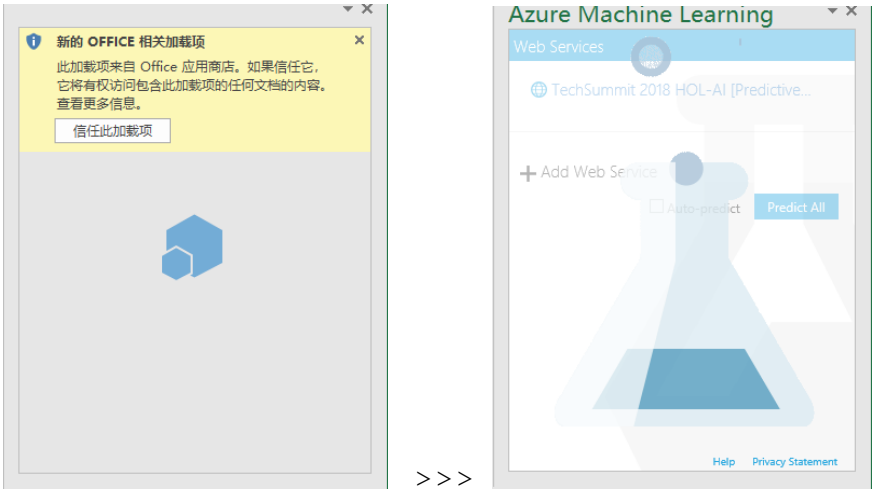
1. 点击“WEB SERVICES”，然后点击已经部署的预测 Web 服务



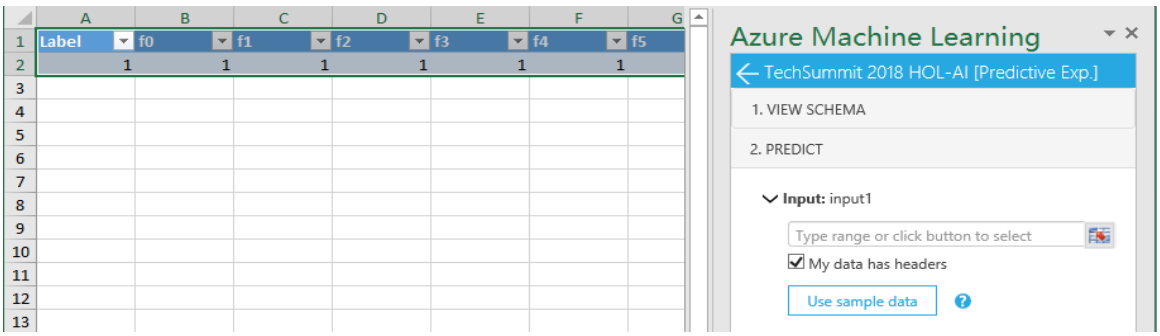
2. 点击“BATCH EXEUTION”后面的“Excel 2013 or later workbook”下载并打开 Excel 文件



3. 如提示新的加载项，点击“信任此加载项”，然后等待 Excel 加载项连接 Azure ML 服务

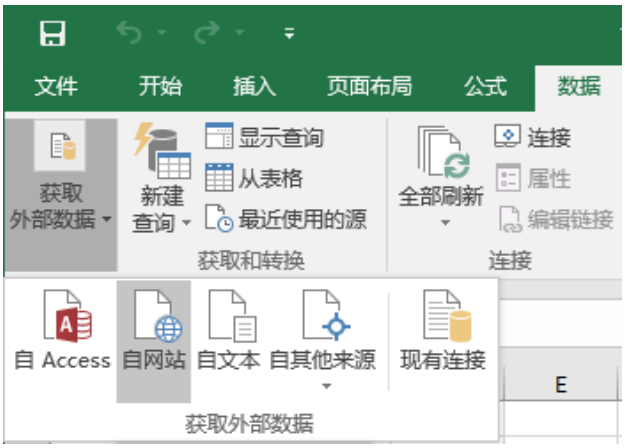


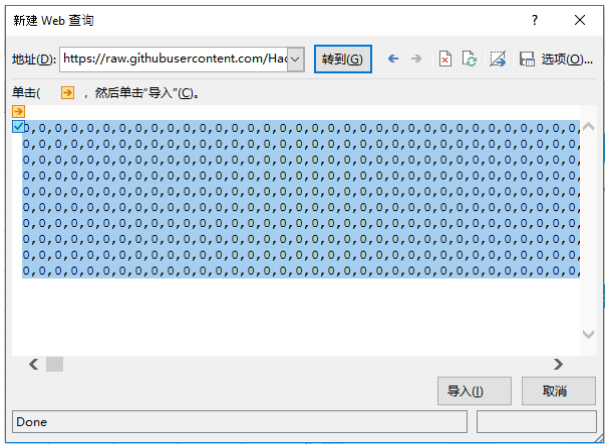
4. 为了匹配数据格式，可点击“Use sample data”显示示例数据作为参考



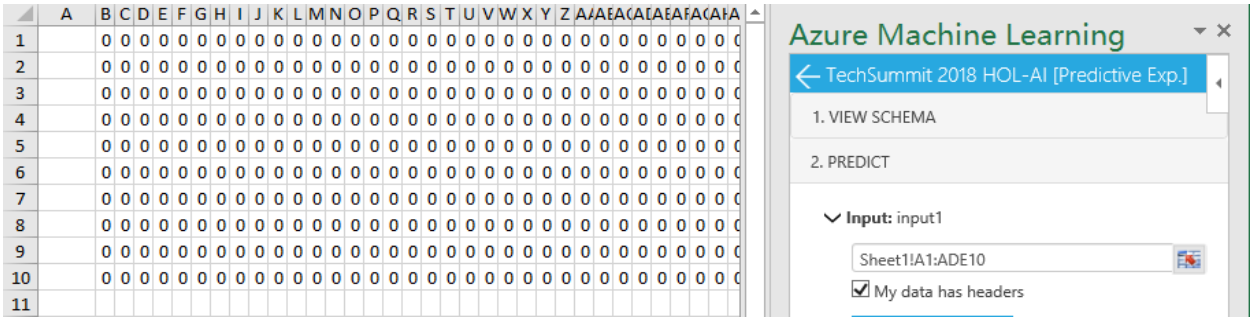
5. 删除示例数据，选中 B1 单元格，然后点击“数据”页，选择”获取外部数据”-”自网站”，在地址栏输入以下地址：

   https://raw.githubusercontent.com/HaoHoo/HOL-AI/master/data/HOL_MNIST_test.csv





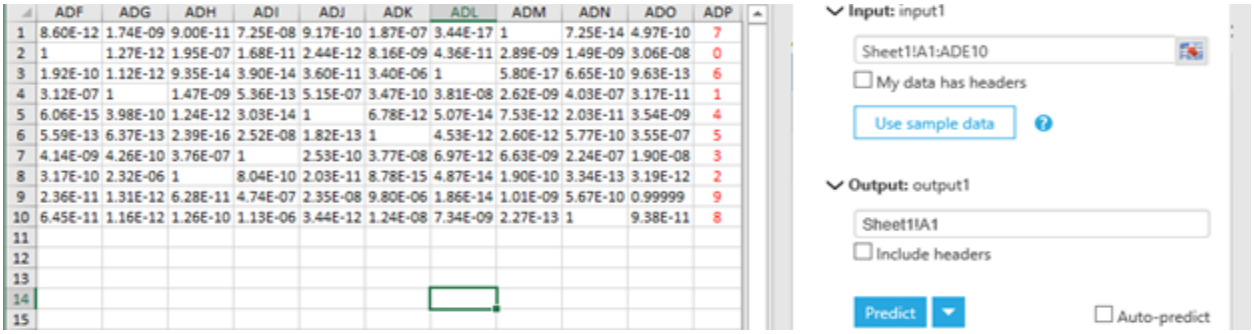
6. 点击“数据”页中的”分列”，选择逗号作为分隔符，导入数据完成后，应如下图所示：



   请思考为什么选择在 B1 单元格导入数据，空出 A 列？

7. 在“Input”处输入 *Sheet1!A1:ADE10* 或手动选择该单元格区域。在“Output”处输入 *Sheet1!1:A1*，清除“My data has headers”和“Include headers”选择框，然后点击“Predict”开始识别测试数据集。

   如果没有错误，在数据的最后 11 列应看到类似如下数据：



	ADF	ADG	ADH	ADI	ADJ	ADK	ADL	ADM	ADN	ADO	ADP
1	8.60E-12	1.74E-09	9.00E-11	7.25E-08	9.17E-10	1.87E-07	3.44E-17	1	7.25E-14	4.97E-10	7
2	1	1.27E-12	1.95E-07	1.68E-11	2.44E-12	8.16E-09	4.36E-11	2.89E-09	1.49E-09	3.06E-08	0
3	1.92E-10	1.12E-12	9.35E-14	3.90E-14	3.60E-11	3.40E-06	1	5.80E-17	6.65E-10	9.63E-13	6
4	3.12E-07	1	1.47E-09	5.36E-13	5.15E-07	3.47E-10	3.81E-08	2.62E-09	4.03E-07	3.17E-11	1
5	6.06E-15	3.98E-10	1.24E-12	3.03E-14	1	6.78E-12	5.07E-14	7.53E-12	2.03E-11	3.54E-09	4
6	5.59E-13	6.37E-13	2.39E-16	2.52E-08	1.82E-13	1	4.53E-12	2.60E-12	5.77E-10	3.55E-07	5
7	4.14E-09	4.26E-10	3.76E-07	1	2.53E-10	3.77E-08	6.97E-12	6.63E-09	2.24E-07	1.90E-08	3
8	3.17E-10	2.32E-06	1	8.04E-10	2.03E-11	8.78E-15	4.87E-14	1.90E-10	3.34E-13	3.19E-12	2
9	2.36E-11	1.31E-12	6.28E-11	4.74E-07	2.35E-08	9.80E-06	1.86E-14	1.01E-09	5.67E-10	0.99999	9
10	6.45E-11	1.16E-12	1.26E-10	1.13E-06	3.44E-12	1.24E-08	7.34E-09	2.27E-13	1	9.38E-11	8
11											
12											
13											
14											
15											

前 10 列为对应图片为 0~9 数字的可能性，第 11 列为机器学习识别出的数字。

8. 回到练习 1 中任务 2 的 notebook，在新代码块中输入如下代码：

```
import numpy as np
import matplotlib.pyplot as plt
arrdat=np.loadtxt("https://raw.githubusercontent.com/HaoHoo/HOL-AI/master/data/HOL_MNIST_test.csv", 'i2', delimiter=",")
imgdat=arrdat.reshape(10,28,28)
fig, axs = plt.subplots(2, 5)
fig.subplots_adjust(left=0.06, right=0.75, top=0.43, bottom=0.06, hspace=0, wspace=0.07)
images = []
k=0
for i in range(2):
    for j in range(5):
        images.append(axs[i,j].imshow(imgdat[k,]))
        axs[i,j].set_xticklabels('')
        axs[i,j].set_yticklabels('')
        axs[i,j].set_xticks([0,13,27])
        axs[i,j].set_yticks([0,13,27])
        k=k+1
axs[1,0].set_xticklabels((0,13,27))
axs[1,0].set_yticklabels((27,13,0))
plt.show()
```

如代码正常执行，应能看到显示测试数据的图片如下：