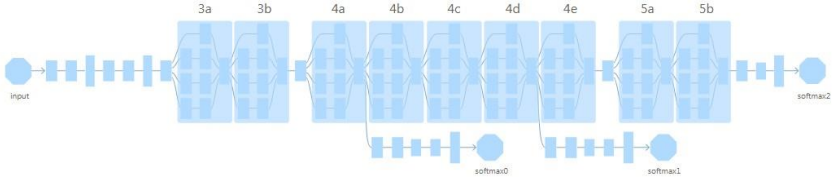
Google TensorFlow™

浙江大学城市学院 ZHEJIANG UNIVERSITY CITY COLLEGE



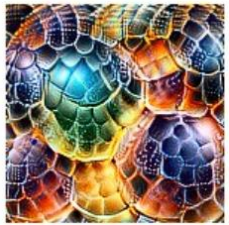
深度学习应用开发
基于TensorFlow的实践

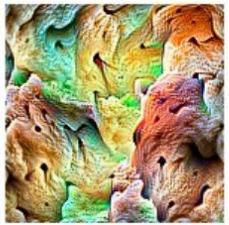
吴明晖 李卓蓉 金苍宏

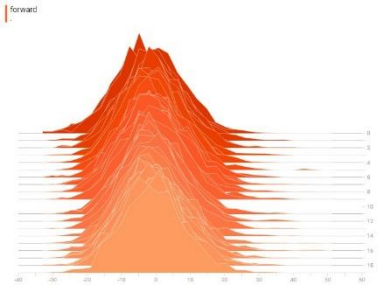
浙江大学城市学院

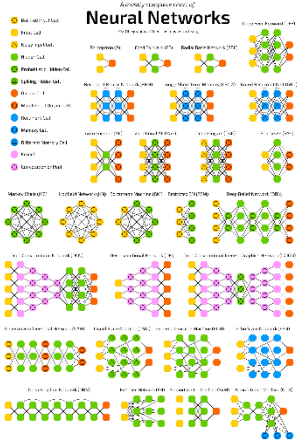
计算机与计算科学学院

Dept. of Computer Science
Zhejiang University City College

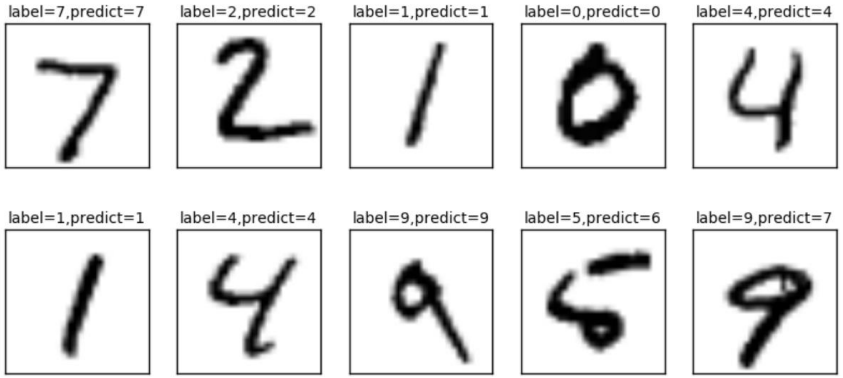


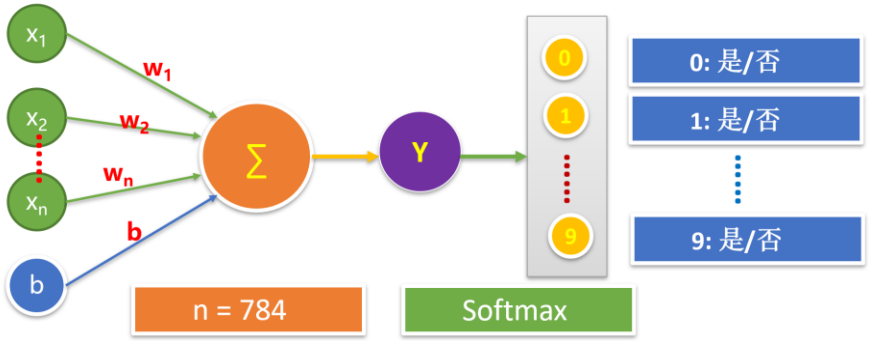






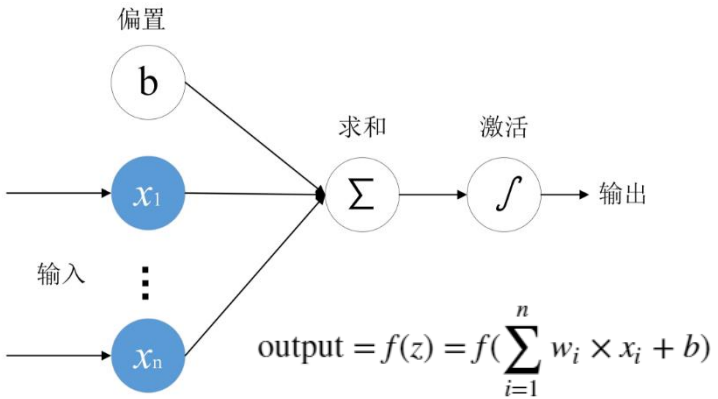
MNIST手写数字识别：分类应用入门






```
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
```

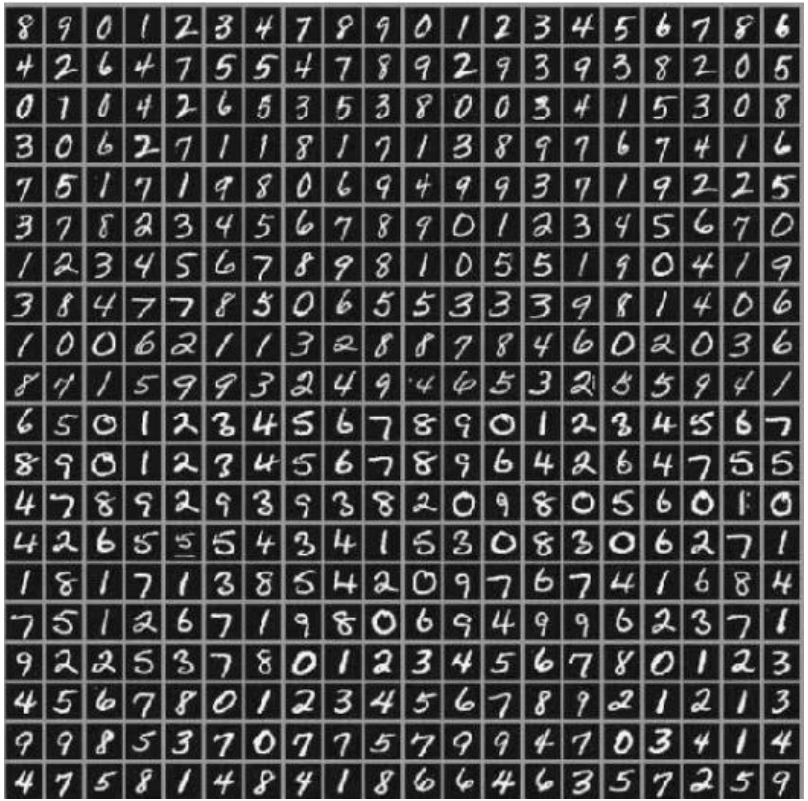
```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
```

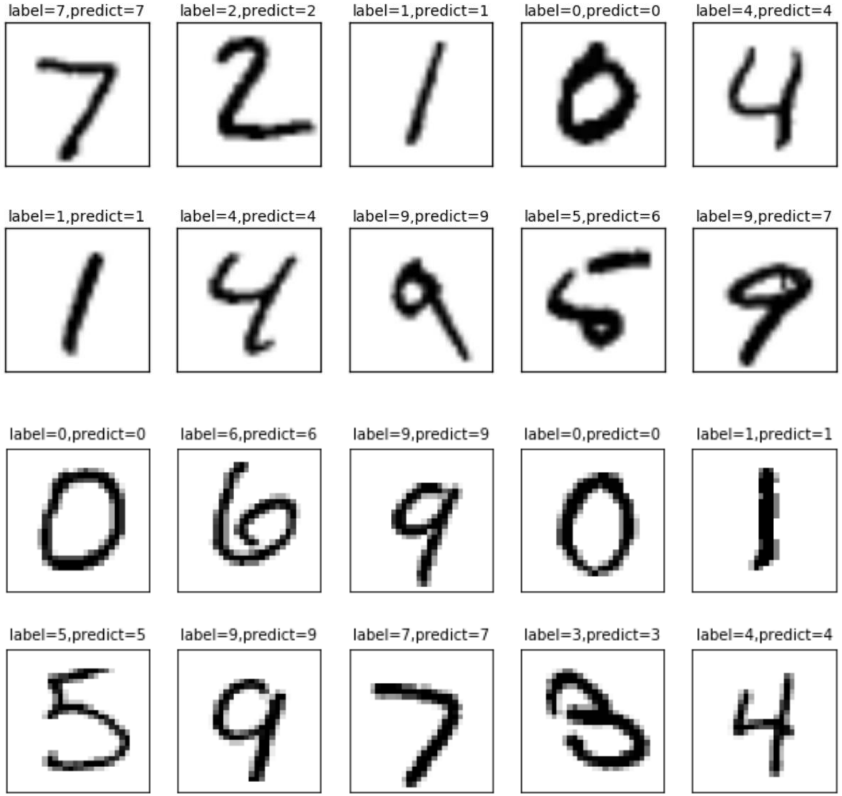


一个神经元处理分类问题

MNIST手写数字识别问题

MNIST手写数字识别：分类问题





MNIST手写数字识别数据集

MNIST 数据集来自美国国家标准与技术研究所，National Institute of Standards and Technology (NIST).

数据集由来自 250 个不同人手写的数字构成，其中 50% 是高中学生，50% 来自人口普查局（the Census Bureau）的工作人员

训练集 55000　验证集 5000　测试集 10000

MNIST手写数字识别数据集

MNIST 数据集可在 http://yann.lecun.com/exdb/mnist/ 获取

TensorFlow提供了数据集读取方法

```
import tensorflow as tf
```

```
import tensorflow.examples.tutorials.mnist.input_data as input_data
```

```
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
```

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
```

MNIST手写数字识别数据集

MNIST数据集文件在读取时如果指定目录下不存在，则会自动去下载，需等待一定时间

如果已经存在了，则直接读取

des › TF_ZUCC_5_MNIST_Single_Neuron › MNIST_data

名称	修改日期	类型	大小
t10k-images-idx3-ubyte.gz	2018/3/7 20:55	WinRAR 压缩文件	1,611 KB
t10k-labels-idx1-ubyte.gz	2018/3/7 20:55	WinRAR 压缩文件	5 KB
train-images-idx3-ubyte.gz	2018/3/7 20:55	WinRAR 压缩文件	9,681 KB
train-labels-idx1-ubyte.gz	2018/3/7 20:55	WinRAR 压缩文件	29 KB

了解MNIST手写数字识别数据集


```
print('训练集 train 数量: ',mnist.train.num_examples,
      ',验证集 validation 数量: ',mnist.validation.num_examples,
      ',测试集 test 数量: ',mnist.test.num_examples)
```

```
训练集 train 数量:  55000 ,验证集 validation 数量:  5000 ,测试集 test 数量:  10000
```

```
print('train images shape:', mnist.train.images.shape,
      'labels shaple:', mnist.train.labels.shape)
```

```
train images shape: (55000, 784) labels shaple: (55000, 10)
```

为什么是 10 ？

为什么是 784 ？

28 * 28 = 784

10 分类 One Hot编码

具体看一幅image的数据

```
len(mnist.train.images[0])
```

```
784
```

```
mnist.train.images[0].shape
```

```
(784,)
```

```
mnist.train.images[0]
```

```
array([ 0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.35294119,  0.5411765 ,  0.92156869,
        0.92156869,  0.92156869,  0.92156869,  0.92156869,  0.92156869,
        0.98431379,  0.98431379,  0.97254908,  0.99607849,  0.96078438,
        0.92156869,  0.74509805,  0.08235294,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
```

Image数据再塑形reshape

```
mnist.train.images[0].reshape(28,28)
```

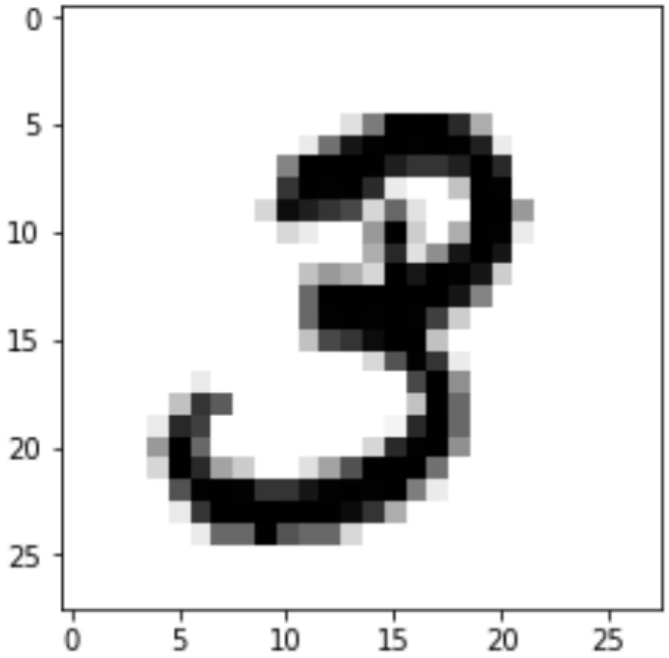
```
array([[ 0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ],
       [ 0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.38039219,  0.37647063,  0.3019608 ,  0.46274513,
         0.2392157 ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ],
       [ 0.        ,  0.        ,  0.        ,  0.35294119,  0.5411765 ,
         0.92156869,  0.92156869,  0.92156869,  0.92156869,  0.92156869,
         0.92156869,  0.98431379,  0.98431379,  0.97254908,  0.99607849,
         0.96078438,  0.92156869,  0.74509805,  0.08235294,  0.
```

可视化 image

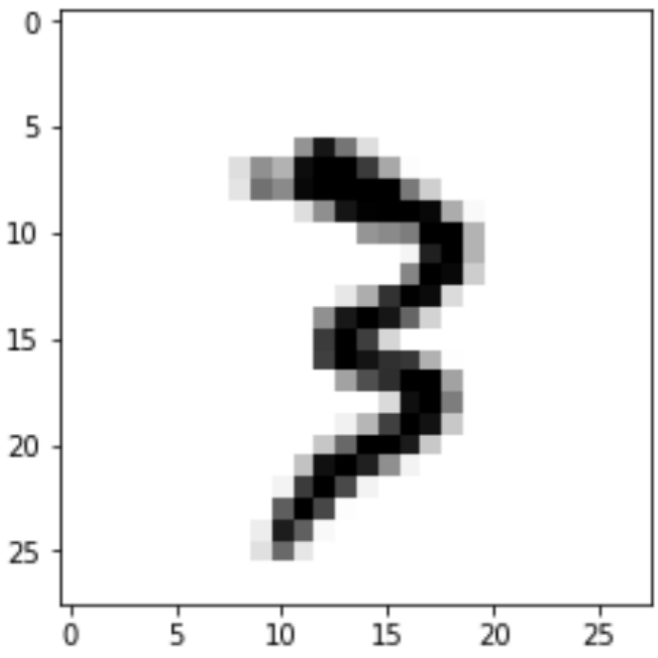
```
import matplotlib.pyplot as plt

def plot_image(image):
    plt.imshow(image.reshape(28,28),cmap='binary')
    plt.show()
```

可视化 image

```
plot_image(mnist.train.images[1])
```




```
plot_image(mnist.train.images[20000])
```



进一步了解 reshape()

```
import numpy as np
int_array = np.array([i for i in range(64)])
print (int_array)
```

```
[ 0  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 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
 50 51 52 53 54 55 56 57 58 59 60 61 62 63]
```

```
int_array.reshape(8,8)
```

```
array([[ 0,  1,  2,  3,  4,  5,  6,  7],
       [ 8,  9, 10, 11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20
       [24, 25, 26, 27, 28
       [32, 33, 34, 35, 36
       [40, 41, 42, 43, 44
       [48, 49, 50, 51, 52
       [56, 57, 58, 59, 60
```

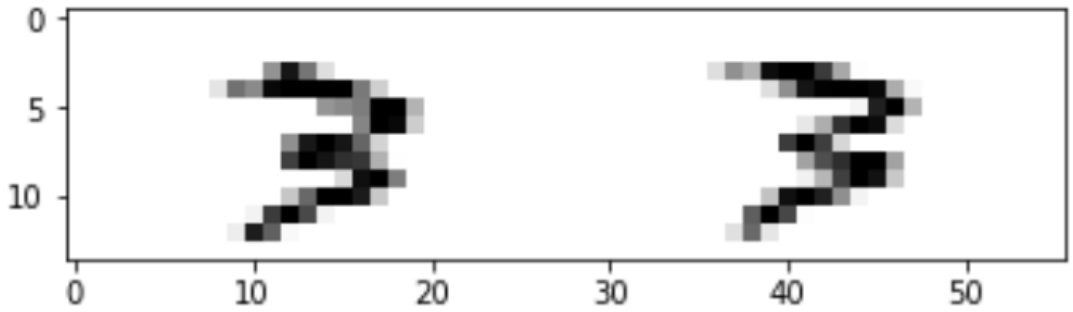
行优先，逐行排列

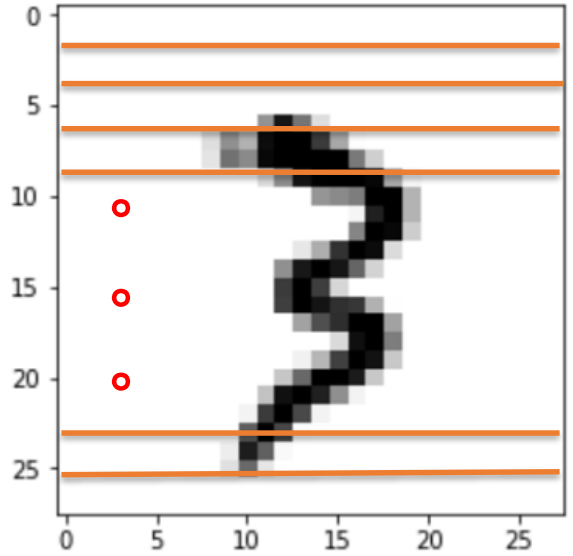
```
int_array.reshape(4,16)
```

```
array([[ 0,  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],
       [32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47],
       [48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63]])
```

思考：以下代码会输出什么图像？

```
plt.imshow(mnist.train.images[20000].reshape(14,56),cmap='binary')
plt.show()
```





标签数据与独热编码

认识标签 label

```
mnist.train.labels[1]
```

```
array([ 0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.])
```

内容并不是直接 3

one hot 独热编码

```
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
```

独热编码（one hot encoding）

一种稀疏向量，其中：

- 一个元素设为 1
- 所有其他元素均设为 0

独热编码常用于表示拥有有限个可能值的字符串或标识符

例如：假设某个植物学数据集记录了 15000 个不同的物种，其中每个物种都用独一无二的字符串标识符来表示。在特征工程过程中，可能需要将这些字符串标识符编码为独热向量，向量的大小为 15000

```
mnist.train.labels[1]
```

```
array([ 0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.])
```

为什么要采用one hot编码

1 将离散特征的取值扩展到了欧式空间，离散特征的某个取值就对应欧式空间的某个点

2 机器学习算法中，特征之间距离的计算或相似度的常用计算方法都是基于欧式空间的

能说 1 比 8 更相似于 3 吗?

3 将离散型特征使用one-hot编码，会让特征之间的距离计算更加合理

数字	0	1	2	3	4	5	6	7	8	9
编码	0	1	2	3	4	5	6	7	8	9

```
array([ 0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.])
```

独热编码如何取值？

```
mnist.train.labels[1]
```

```
array([ 0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.])
```

内容并不是直接 3

one hot 独热编码

```
import numpy as np
np.argmax(mnist.train.labels[1])
```

```
3
```

argmax返回的是最大数的索引

非one hot编码的标签值

```
mnist_no_one_hot = input_data.read_data_sets("MNIST_data/", one_hot=False)
```

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
```

```
print(mnist_no_one_hot.train.labels[0:10])
```

```
[7 3 4 6 1 8 1 0 9 8]
```

one_hot = False，直接返回数值

数据集的划分

数据集划分

构建和训练机器学习模型是希望对新的数据做出良好预测

如何去保证训练的实效，可以应对以前未见过的数据呢？

一种方法是将数据集分成两个子集：

训练集 – 用于训练模型的子集

测试集 – 用于测试模型的子集

通常，在测试集上表现是否良好是衡量能否在新数据上表现良好的有用指标，前提是：

测试集足够大

不会反复使用相同的测试集来作假

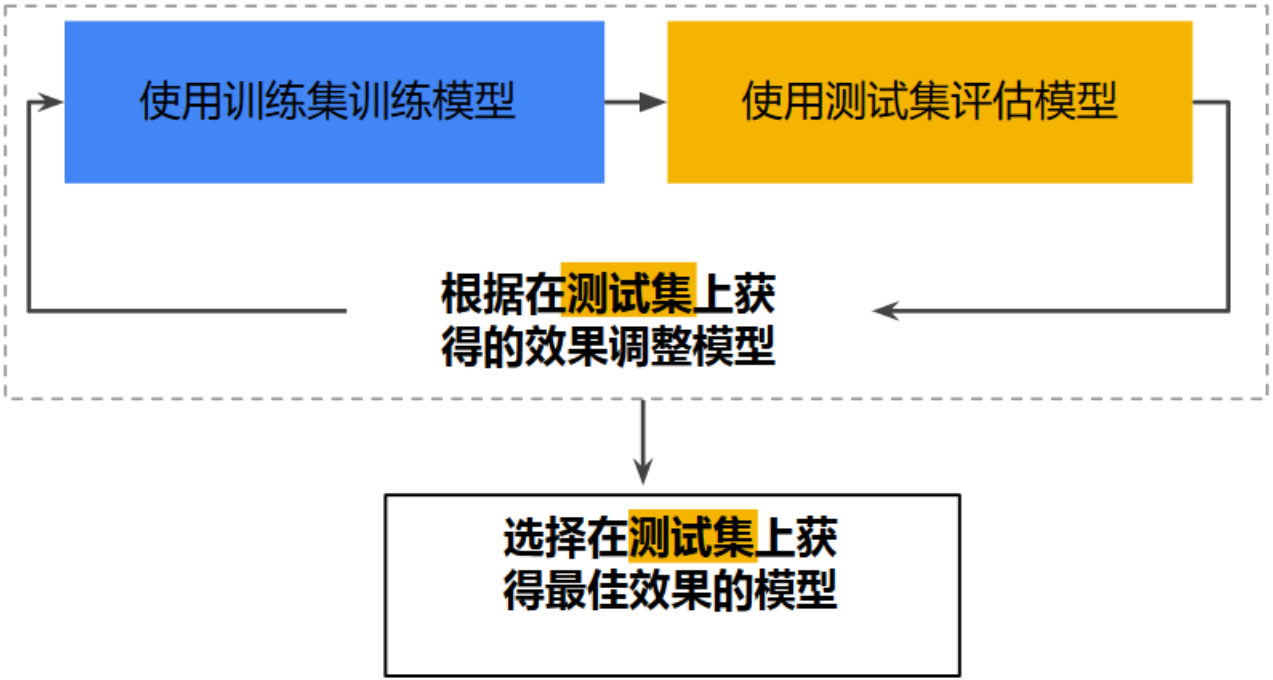
拆分数据

将单个数据集拆分为一个**训练集**和一个**测试集**

确保测试集满足以下两个条件：

- **规模足够大**，可产生具有统计意义的结果
- **能代表整个数据集**，测试集的特征应该与训练集的特征相同

工作流程


使用训练集训练模型
使用测试集评估模型
根据在测试集上获得的效果调整模型
选择在测试集上获得最佳效果的模型

有没有什么问题？

思考：

使用测试集和训练集来推动模型开发**迭代**的流程。

在每次迭代时，都会对训练数据进行训练并评估测试数据，并以基于测试数据的评估结果为指导来选择和更改各种**模型超参数**，例如学习速率和特征。这种方法是否存在问题？

多次重复执行该流程可能导致模型不知不觉地拟合了特定测试集的特性

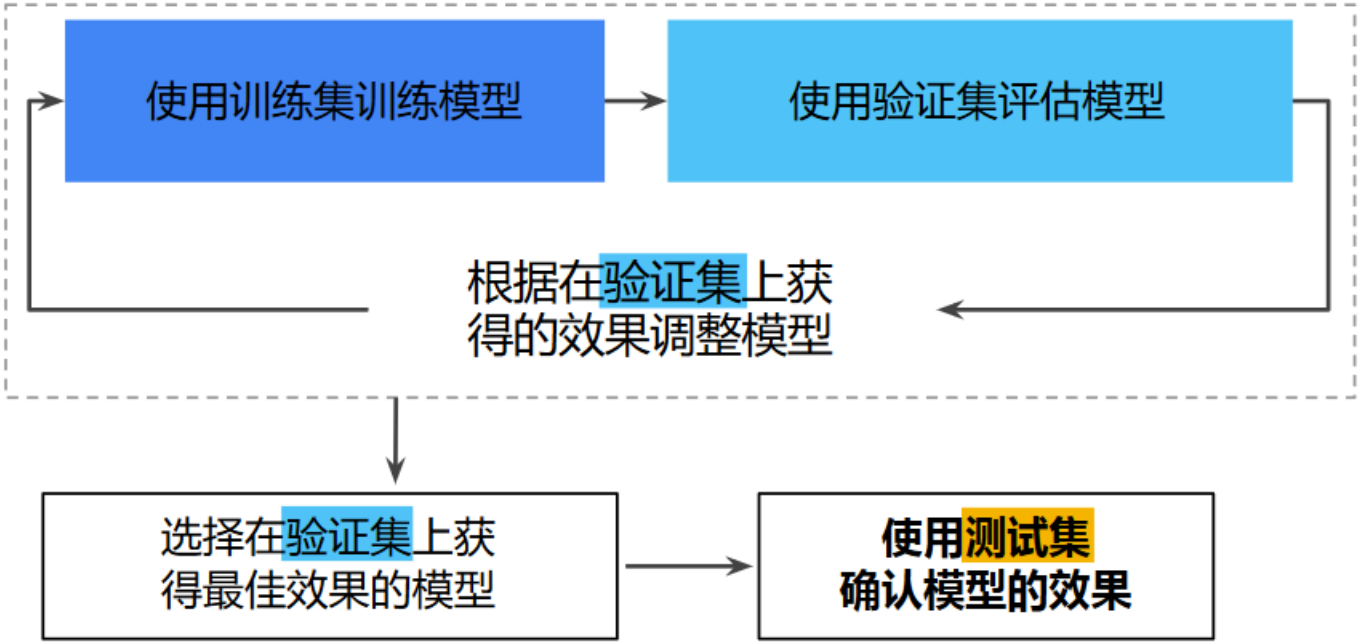
新的数据划分

通过将数据集划分为三个子集，可以大幅降低过拟合的发生几率：

使用**验证集**评估训练集的效果。

在模型“通过”验证集之后，使用测试集再次检查评估结果

新的工作流程


使用训练集训练模型
使用验证集评估模型
根据在验证集上获得的效果调整模型
选择在验证集上获得最佳效果的模型
使用测试集确认模型的效果

读取验证数据

```
print('validation images:', mnist.validation.images.shape,
      'labels:', mnist.validation.labels.shape)
```

```
validation images: (5000, 784) labels: (5000, 10)
```

读取测试数据

```
print('test images:', mnist.test.images.shape,
      'labels:', mnist.test.labels.shape)
```

```
test images: (10000, 784) labels: (10000, 10)
```

数据的批量读取

一次批量读取多条数据

```
batch_images_xs, batch_labels_ys = \
    mnist.train.next_batch(batch_size=10)
```

```
print(mnist.train.labels[0:10])
```

```
[[ 0.  0.  0.  0.  0.  0.  0.  1.  0.  0.]
 [ 0.  0.  0.  1.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  1.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  1.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  1.  0.]
 [ 0.  1.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  1.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  1.  0.]]
```

```
print(batch_labels_ys)
```

```
[[ 0.  1.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  1.  0.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  1.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  1.  0.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  1.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  1.  0.]
 [ 0.  0.  0.  0.  0.  1.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  1.  0.]]
```

next_batch（）实现内部会对数据集先做shuffle

数据读取

MNIST手写数字识别数据集读取

```
import tensorflow as tf
```

```
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
```

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
```