# 炼丹入门——基于MNIST手写数字识别

——1951976 李林飞

# 目录

CONTENTS

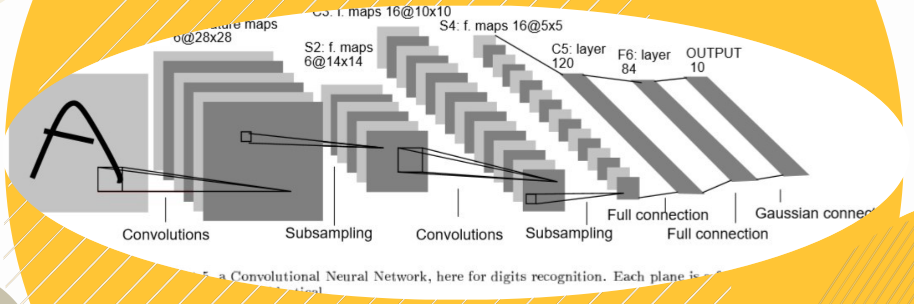
01

# MNIST数据集

数据历史——机器学习中的"hello world"

数据特性——"修仙人士"的真实工作

# MNIST数据集简介




LeNet-5 网络

- Yan LeCun

  1998, Gradient-Based Learning Applied to Document Recognition, LeNet-5 网络

- 数据来源

  NIST，训练集(training set)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。测试集(test set)也是同样比例的手写数字数据。

- URL

  http://yann.lecun.com/exdb/mnist/
  线性分类器（Linear Classifiers） K-近邻算法（K-Nearest Neighbors） 支持向量机（SVMs） 神经网络（Neural Nets卷积神经网络（Convolutional nets）

# 数据特性——导入库

```
# “Numeric Python” 一个由多维数组对象和用于处理数组的例程集合组成的库
import numpy as np

# 激活matplotlib
%matplotlib inline  import matplotlib
import matplotlib.pyplot as plt

# 随机数
from random import *

# 机器学习框架: Apache Singa | Amazon Machine Learning (AML) | Azure ML Studio
#       Caffe | H2O | Massive Online Analysis (MOA) | MLlib (Spark)
#       Mlpack | Pattern | Scikit-Learn | Shogu | TensorFlow
#       Theano | Torch | Veles
import tensorflow as tf
from tensorflow.contrib.learn.python.learn.datasets.mnist import read_data_sets
```

# 数据特性——数据文件

```
# 读入数据
warnings.filterwarnings('ignore')
mnist = read_data_sets('./Datasets')
```

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

train-images-idx3-ubyte.gz —— 训练图片集

train-labels-idx1-ubyte.gz —— 训练标签集

t10k-images-idx3-ubyte.gz —— 测试图片集

t10k-labels-idx1-ubyte.gz —— 测试标签集

# 数据特性——类型与分类

```
# 查看数据类型
type(mnist)
```

tensorflow.contrib.learn.python.learn.datasets.base.Datasets

```
# 训练集
train_images = mnist.train.images
train_labels = mnist.train.labels

# 验证集
validation_images = mnist.validation.Images
validation_labels = mnist.validation.labels

# 测试集
test_images = mnist.test.images
test_labels = mnist.test.labels
```

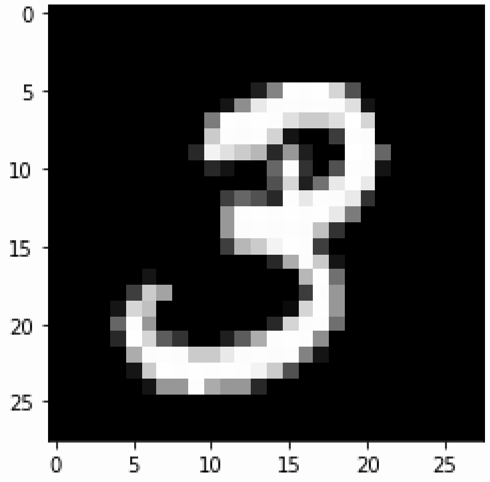
# 数据特性——shape

```
# 取出一条数据,打印数据的shape
im = mnist. train. images[1]
print(im. shape)
```

```
(784,)
```

```
# 查看图片
im = im. reshape(28, 28)
print(im. shape)
# plt. imshow(im)
# plt. imshow(im, cmap='Greys')
plt. imshow(im, cmap='gray')
plt. show()
# 查看标签
print(mnist. train. labels[1])
```

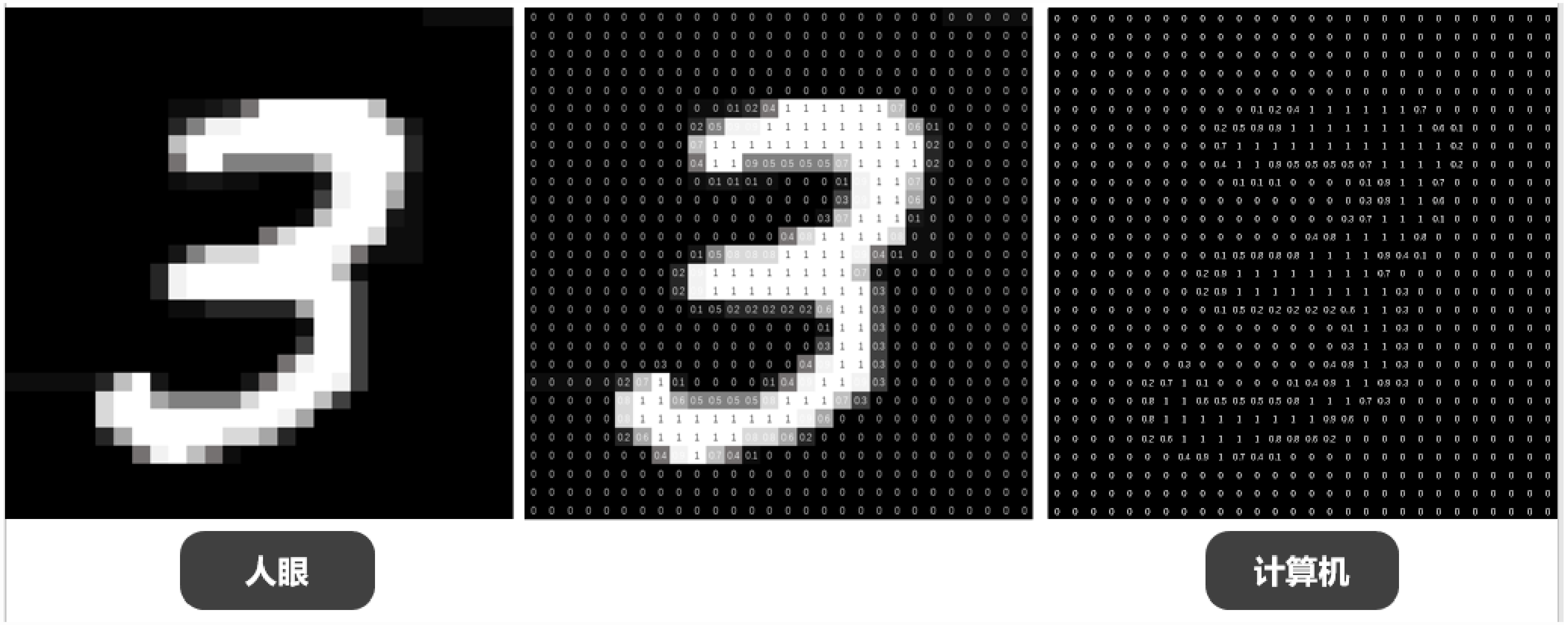
```
(28, 28)
```




```
3
```

# 数据特性——one-hot编码

28*28=784

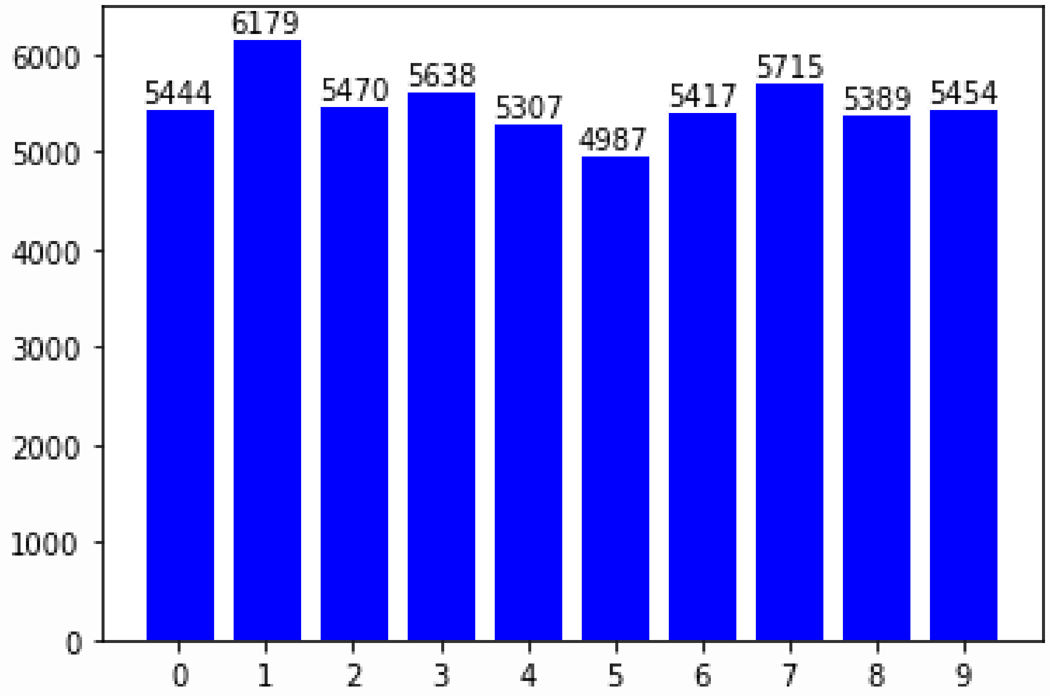



```
# 查看one-hot编码
from tensorflow.examples.tutorials.mnist import
input_data  mnist_test = input_data.
read_data_sets("./Datasets/", one_hot=True)
print(mnist_test. train. labels[1])
```

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

# 数据特性——统计数据

```
# 统计可视化
X = []
Y = []

for i in range(10):
    x = i
    y = np.sum(train_labels == i)
    X.append(x)
    Y.append(y)
    plt.text(x, y, '%s' % y, ha='center', va= 'bottom')

plt.bar(X, Y, facecolor='blue', edgecolor='white')
plt.xticks(X)
plt.show()
```



02

# 训练数据预处理

自选数据
二分类变量

# 自选数据

- 随机选取数据——next.batch()

```
x_train, y_train = mnist.train.next_batch(60000)
x_test, y_test = mnist.test.next_batch(10000)
```

- 查看数据形状

```
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

((60000, 784), (60000,), (10000, 784), (10000,))

- 转置——reshape()

```
Xinput_,yinput_ = x_train, y_train

Xinput_ = x_train.reshape(60000,784)
yinput_ = y_train.reshape(60000,)

X = Xinput_
y = yinput_
```

# 二分类变量——准备训练集测试集

- 筛选0和1

```
X_train = X[np.any([y == 1,y == 2], axis = 0)]
y_train = y[np.any([y == 1,y == 2], axis = 0)]
```

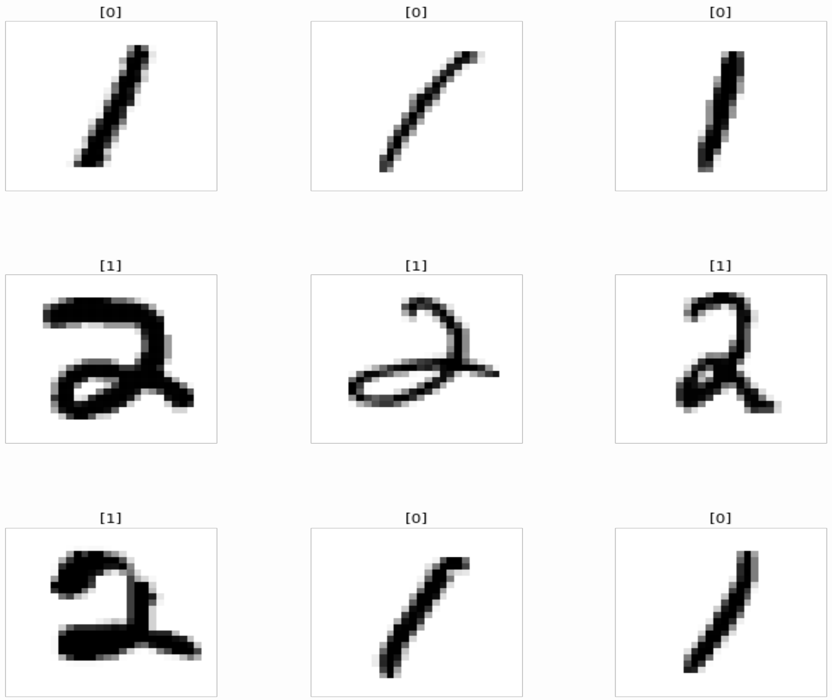
- 查看分布

```
print("number of 1:", np.count_nonzero(y_train == 1))
print("number of 2:", np.count_nonzero(y_train == 2))
```

```
number of 1: 6764
number of 2: 5968
```

- 二分类变量标签

```
# 设置分类变量
y_train_shifted = y_train_tr - 1
```

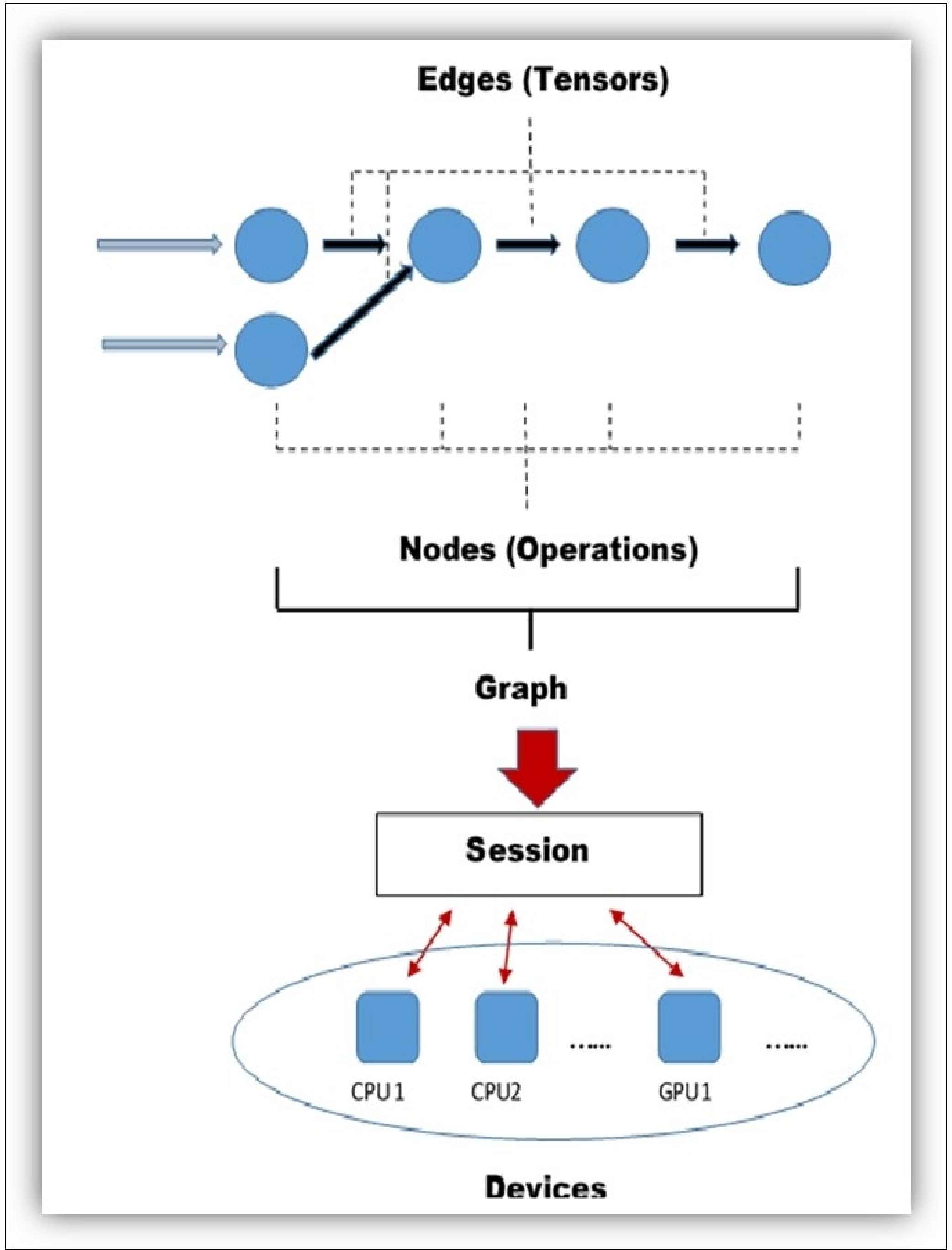


03

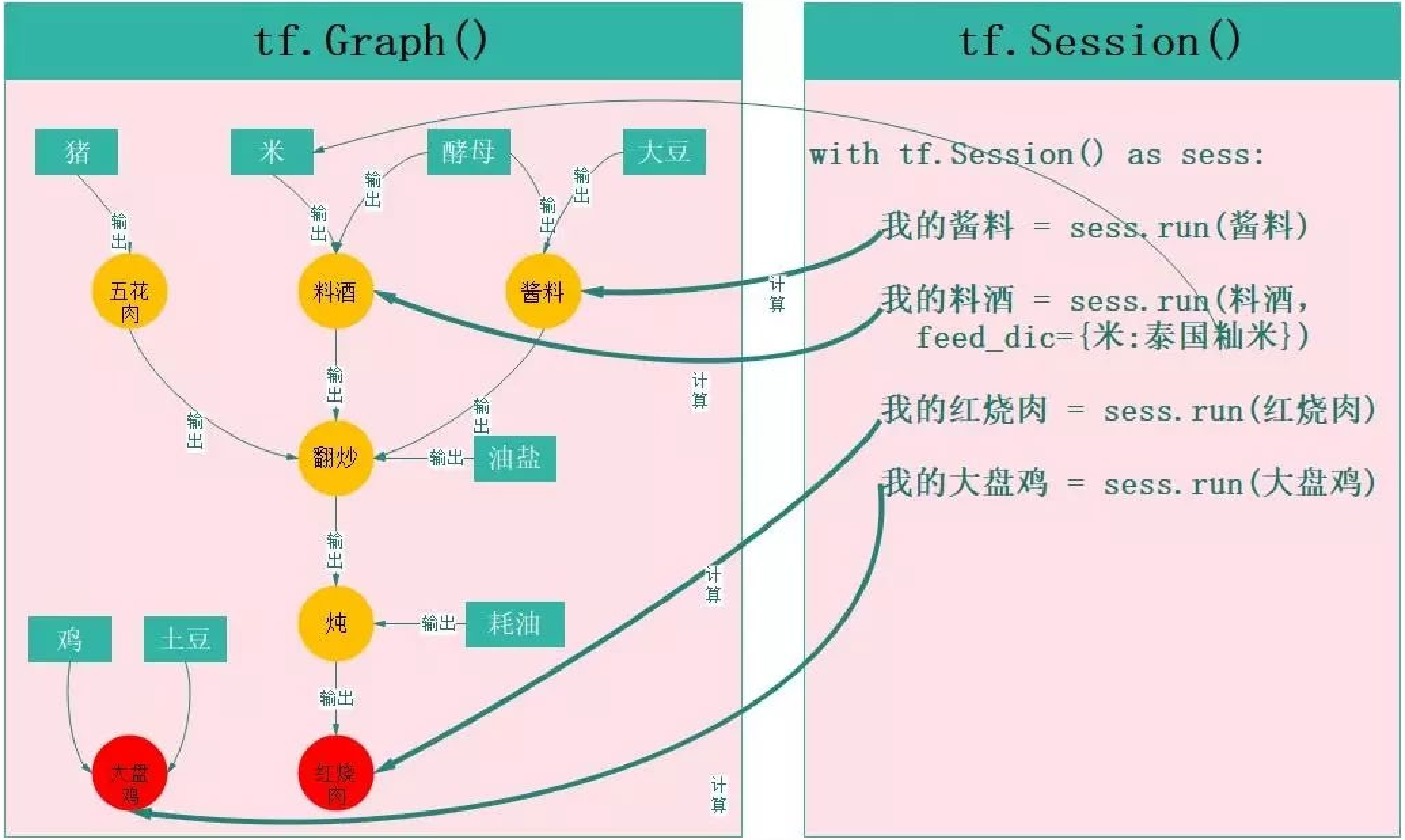
# 构建训练模型

Graph——构建Tensorflow计算图
Session——构建计算图的运行函数
Run——开始训练

# Graph & Session



# Graph & Session



# Graph —— placeholder

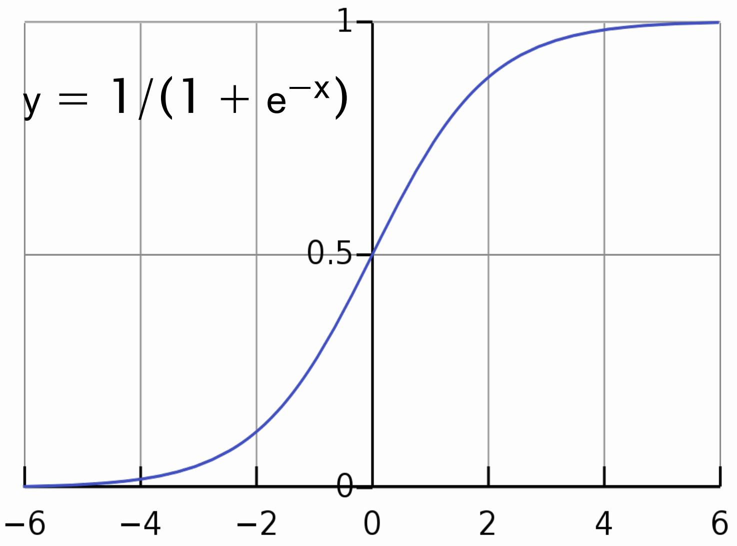
```
# 清除默认图形堆栈并重置全局默认图形
# 重置计算图，若无：每在jupyter notebook上运行一次上述程序，就会在图上新增一个节点。
tf. reset_default_graph()

# tf.placeholder( dtype,shape=None,name=None)
# dtype:数据类型；shape:数据形状；namme:数据名称
# placeholder()函数是在神经网络构建graph的时候在模型中的占位，    # 此时并
没有把要输入的数据传入模型，它只会分配必要的内存。
# 等建立session，在会话中，运行模型的时候通过feed_dict()函数向占位符喂入数据。
X = tf. placeholder(tf. float32, [n_dim, None])       # image 784
Y = tf. placeholder(tf. float32, [1, None])  # label 二分类
learning_rate = tf. placeholder(tf. float32, shape=())# 学习率

# tensorflow中进行优化的参数必须定义成tf.Variable类型
# tf.Variable(initializer,name),initializer是初始化参数 —— 计算图中一个值 #
初始化零向量
W = tf. Variable(tf. zeros([1, n_dim]))       # 权重
b = tf. Variable(tf. zeros(1)) # 偏移量

init = tf. global_variables_initializer() # 初始化模型参数
```

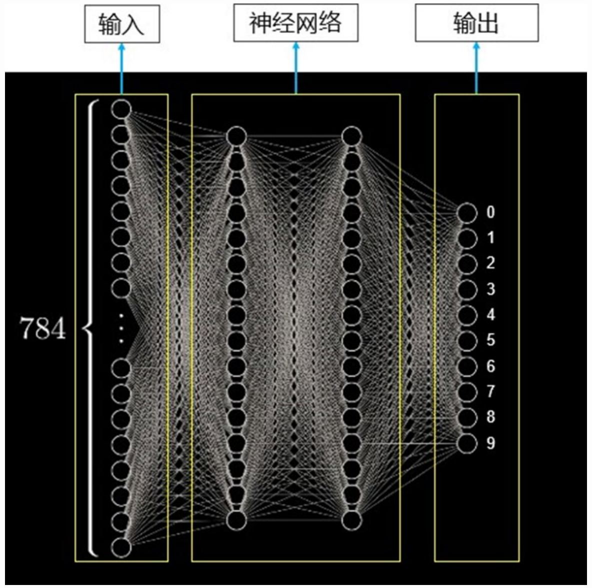
# Graph —— 逻辑回归

$y = \mathrm{sigmoid}(W^T * X + b)$ —— W：权重，b：偏移量

- 定义神经网络模型

```
#定义神经网络模型
y_ = tf.sigmoid(tf.matmul(W,X)+b)        # W*X+b

# y = tf.nn.softmax(tf.matmul(x,W) + b)
# cross_entropy = -tf.reduce_sum(y_*tf.log(y))      # 交叉熵
```



# Graph —— 损失函数

- 漂亮的损失函数：交叉熵(cross-entropy)

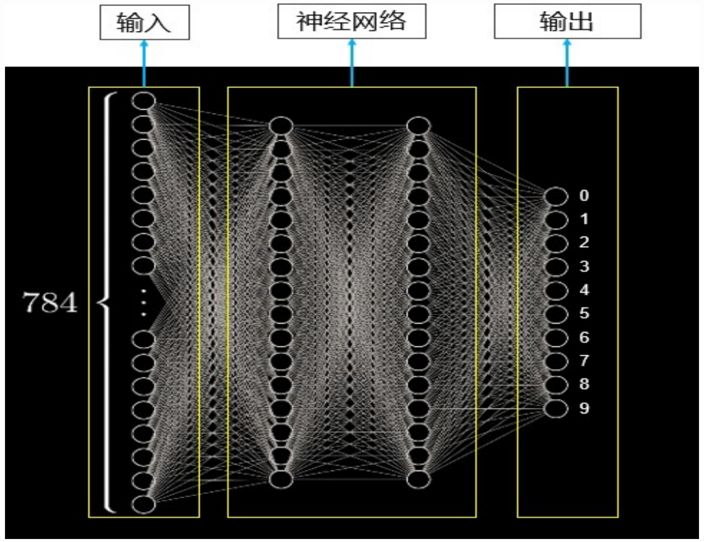
$$H_{y'}(y) = -\sum_i y_i' log(y_i)$$ ——y 是我们预测的概率分布, y' 是实际的分布

损失函数（熵）为 $J = -(ylog\hat{y} + (1-y)log(1-\hat{y})$

- 定义交叉熵

```
#定义损失函数计算节点，此处为熵
cost = - tf.reduce_mean(Y * tf.log(y_)+(1-Y) * tf.log(1-y_))

# 计算cost平均值
# tf.reduce_mean函数用于计算张量tensor沿着指定的数轴（tensor的
# 某一维度）上的的平均值，主要用作降维或者计算tensor（图像）的
# 平均值。
# reduce_mean(input_tensor,axis=None,keep_dims=False,name=None,
# reduction_indices=None)
```

# Graph —— 反向传播 & 梯度下降




## 反向传播算法(backpropagation algorithm)

输出cost小 → 反向传播函数 BP算法 BP神经网络 (BackPropagation) → 加强网络 → 增大网络参数

输出cost大 → 反向传播函数 BP算法 BP神经网络 (BackPropagation) → 减弱网络 → 减小网络参数

功能：修正神经网络参数
让输出值更加靠近答案

## 梯度下降算法

```
# 定义训练节点：梯度下降算法（gradient descent algorithm）最小化交叉熵
training_step = tf. train. GradientDescentOptimizer(learning_rate). minimize(cost)
# 优化算法
#  class tf.train.AdagradOptimizer # class
tf.train.MomentumOptimizer  # class
tf.train.AdamOptimizer
# class tf.train.FtrlOptimizer
# class tf.train.RMSPropOptimizer  # class
tf.AggregationMethod
# tf.clip_by_norm(t, clip_norm, name=None)
```

# Session —— 定义

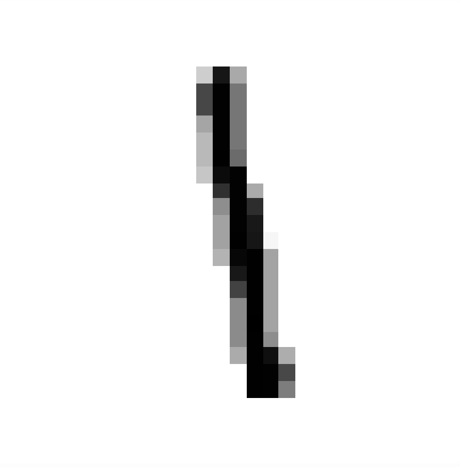
```
# 定义运行函数
def run_logistic_model(learning_r, training_epochs, train_obs, train_labels, debug
  = sess = tf. Session()
  sess. run(init)

  cost_history = np. empty(shape=[0], dtype = float)# 记录损失值

  for epoch in range(training_epochs+ 1):
    #运行训练节点，开始训练
    sess. run(training_step, feed_dict = {X: train_obs, Y: train_labels, learning_
    #运行损失函数计算节点，获得损失函数值
    cost_ = sess. run(cost, feed_dict={ X:train_obs, Y: train_labels, learning_rat

    cost_history = np. append(cost_history, cost_)  # 每50轮计算

    一个损失值

    if (epoch % 50 == 0) & debug:
      print("Reached epoch",epoch,"cost J =", str. format('{0:.6f}', cost_))

  return sess, cost_history
```

# Session —— 启动

```
# 在Session中启动模型
sess = tf.Session()
sess.run(init)
print(sess.run(y_, feed_dict={X:Xtrain, Y: ytrain, learning_rate: 2}))
print(sess.run(cost, feed_dict={X:Xtrain, Y: ytrain, learning_rate: 2}))
sess.close()
```

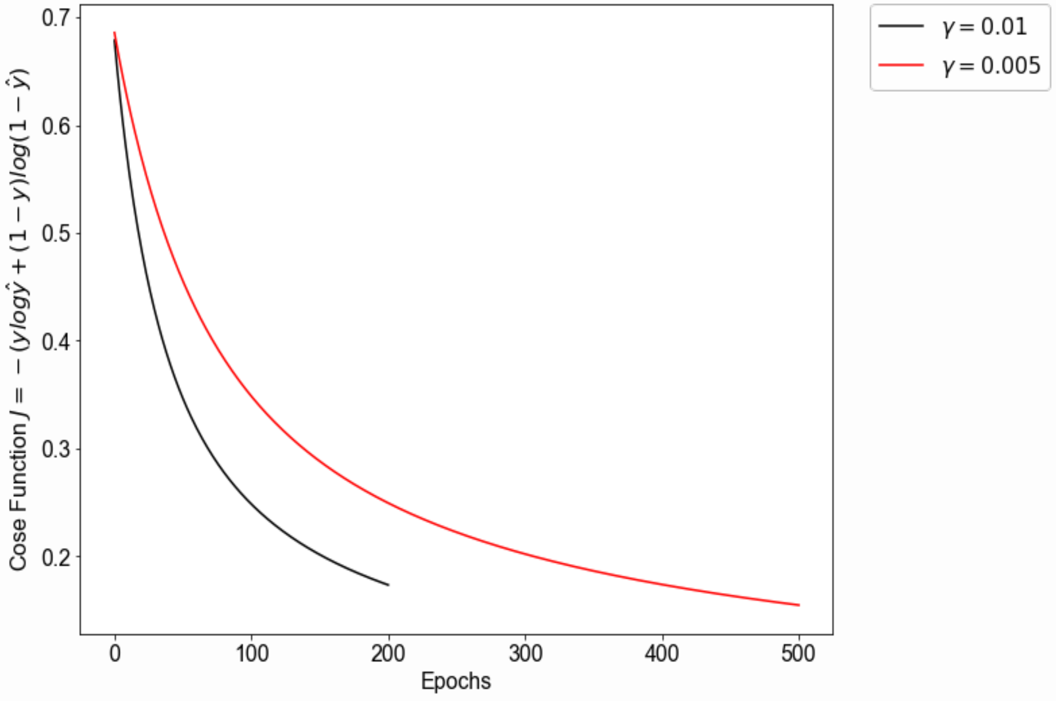
# Run —— 学习率0.05/0.01，迭代次数500/200

```
sess, cost_history2 = run_logistic_model(learning_r = 0.005,
                training_epochs = 500,
                train_obs = Xtrain,
                train_labels = ytrain,
                debug = True)
```



该图片真实值： [1]

该图片预测后的结果： [0.2044488]



04

# 模型评估与拓展

计算准确率

Scikit-Learn逻辑回归

# 预测准确率

定义用于精度计算的计算节点，查看训练针对训练集的手写数字识别精度

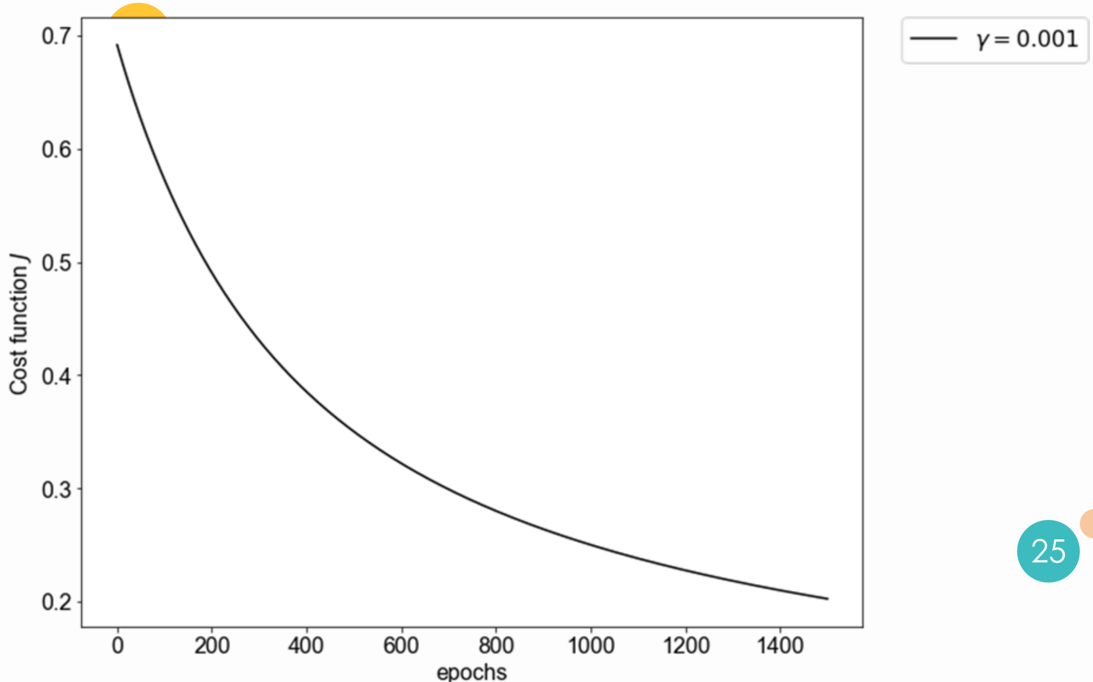
对样本$i$分类是这样计算的：

如果$P(y^{(i)}=1|x^{(i)})<0.5$，则样本属于类别0（即手写数字图像为1）

如果$P(y^{(i)}=1|x^{(i)})>0.5$，则样本属于类别1（即手写数字图像为2）

```
# 计算预测准确率
correct_prediction1=tf.equal(tf.greater(y_, 0.5), tf.equal(Y,1))

accuracy1 = tf. reduce_mean(tf. cast(correct_prediction1, tf. float32))

print(sess.run(accuracy1, feed_dict={X:Xtrain, Y: ytrain, learning_rate: 0.05}))
```

0.97031105



# 拓展 —— Scikit-Learn库函数

- URL

http://scikitlearn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.ht

- Logistic Regression

LogisticRegression： 手动指定一个正则化系数
LogisticRegressionCV： 使用了交叉验证来选择正则化系数C
logistic_regression_path： 拟合数据后，不能直接来做预测，只能为拟合数据选择合适逻辑 回归的系数和正则化系数

- 数字识别

```
# 导入库
from sklearn.linear_model import LogisticRegression
```

```
# 定义函数
logistic = LogisticRegression()
```

```
# 准备数据
XX = Xtrain.T
YY = ytrain.T.ravel()
```

```
#  训练模型
logistic.fit(XX,YY)
```

```
# 查看精度
sum(logistic.predict(XX) == YY) / len(XX)
```

0.9966226830034559

谢谢聆听，期末加油!

——1951976 李林飞