基于TensorFlow的MNIST数据集识别实践

武汉大学 谷歌 联合实验室

目录

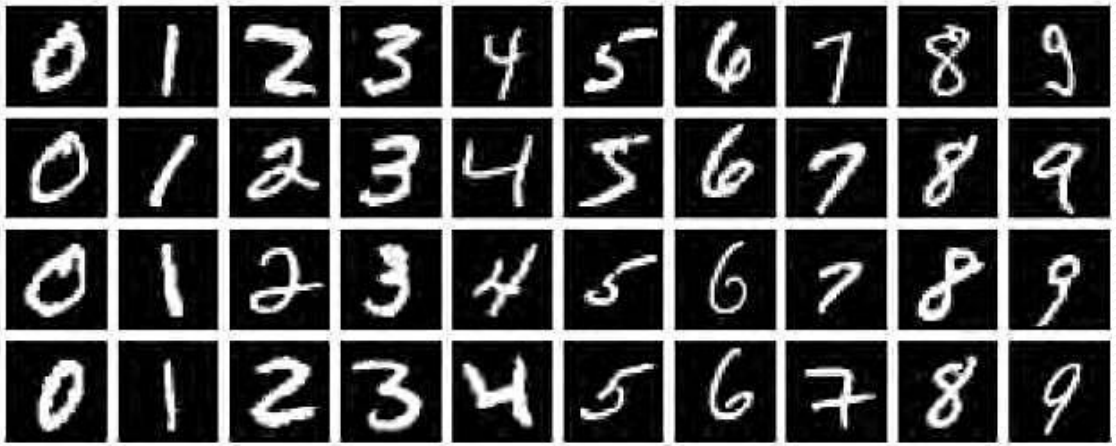
- MNIST数据集
- MNIST手写数字识别实践
  - 逻辑回归方法
  - 人工神经网络方法
  - 卷积神经网络CNN方法
- 调试技巧：TensorFlow可视化
- 小结

MNIST数据集

简介：手写数字数据库，它有60000个训练样本集和10000个测试样本集

官网：http://yann.lecun.com/exdb/mnist/

外观：

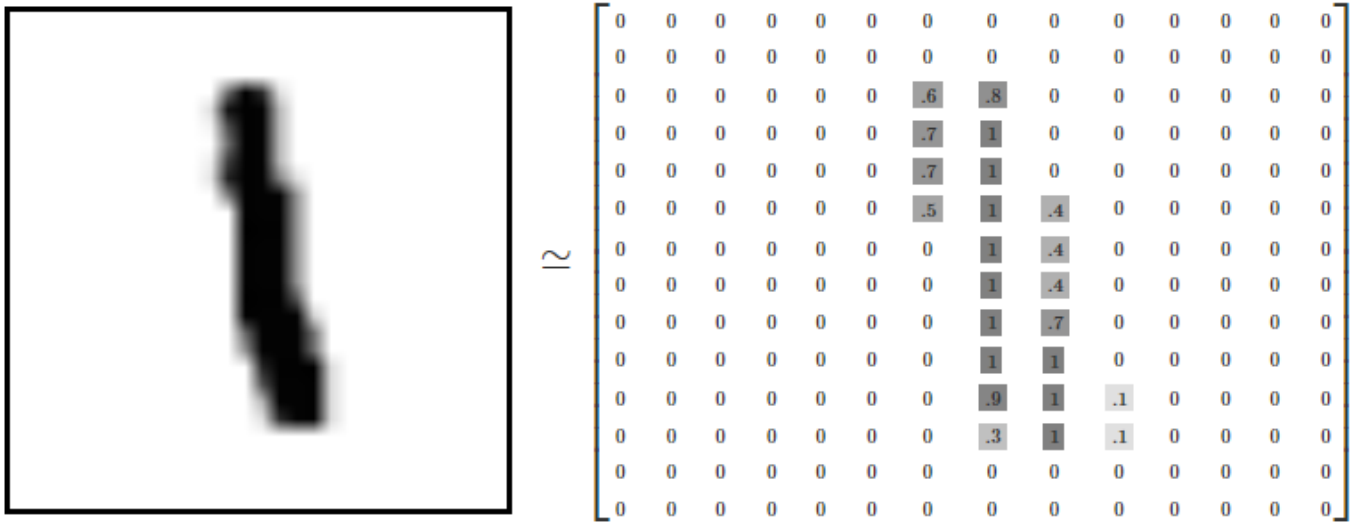


图片像素：28*28

MNIST数据集

数据集构成

数据集	目的
data_sets.train	55000个图像和标签（labels），作为主要训练集。
data_sets.validation	5000个图像和标签，用于迭代验证训练准确度。
data_sets.test	10000个图像和标签，用于最终测试训练准确度（trained accuracy）。



MNIST数据集下载

```
In [3]: import tensorflow as tf

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

```
Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
Extracting MNIST_data/train-images-idx3-ubyte.gz
Successfully downloaded train-labels-idx1-ubyte.gz 28881 bytes.
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Successfully downloaded t10k-images-idx3-ubyte.gz 1648877 bytes.
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Successfully downloaded t10k-labels-idx1-ubyte.gz 4542 bytes.
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
```

运行上面3行代码，会在当前路径下新建文件夹MNIST_data，并从Git上下载数据集并解压

课程代码：
Machine_Learning\03_MNIST\00_input_data.ipynb

MNIST数据集下载

哪里安装的？

```
In [3]: import tensorflow as tf

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

在源码目录
https://github.com/tensorflow/models/tree/master/research/slim/datasets 存放了ImageNet、MNIST等常用数据集的下载脚本

- dataset_factory.py
- dataset_utils.py
- download_and_convert_cifar10.py
- download_and_convert_flowers.py
- download_and_convert_imagenet.sh
- download_and_convert_mnist.py
- download_imagenet.sh

```
# The URLs where the MNIST data can be downloaded.
_DATA_URL = 'http://yann.lecun.com/exdb/mnist/'
_TRAIN_DATA_FILENAME = 'train-images-idx3-ubyte.gz'
_TRAIN_LABELS_FILENAME = 'train-labels-idx1-ubyte.gz'
_TEST_DATA_FILENAME = 't10k-images-idx3-ubyte.gz'
_TEST_LABELS_FILENAME = 't10k-labels-idx1-ubyte.gz'

_IMAGE_SIZE = 28
_NUM_CHANNELS = 1
```

MNIST数据集查看

```
import tensorflow as tf
#from tensorflow.examples.tutorials.mnist import input_data

#one_hot 独热编码，也叫一位有效编码。在任意时候只有一位为1，其他位都是0
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)

train_images = mnist.train.images
train_labels = mnist.train.labels
test_images = mnist.test.images
test_labels = mnist.test.labels

print("train_images_shape:", train_images.shape)
print("train_labels_shape:", train_labels.shape)
print("test_images_shape:", test_images.shape)
print("test_labels_shape:", test_labels.shape)
print("train_images:", train_images[0])
print("train_images_length:",len(train_images[0]))
print("train_labels:", train_labels[0])
```

运行上面代码，会打印训练集中起始位置图片及其标签

```
 0.         0.         0.         0.         0.         0.
 0.34901962 0.98431379 0.9450981  0.33725491 0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.01960784 0.80784321 0.96470594 0.6156863  0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.
 0.01568628 0.45882356 0.27058825 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.
train_images_length: 784
train_labels: [ 0.  0.  0.  0.  0.  0.  0.  1.  0.  0.]
```

课程代码：cell1
Machine_Learning\03_MNIST\
01_logistic_regression.ipynb

目录

- MNIST数据集
- MNIST手写数字识别实践
  - 逻辑回归方法
  - 人工神经网络方法
  - 卷积神经网络CNN方法
- 调试技巧： TensorFlow可视化
- 小结

MNIST手写数字识别实践

预备知识

课程代码：cell2 Machine_Learning\03_MNIST\01_logistic_regression.ipynb

tf.argmax tf.reduce_mean tf.equal tf.cast

```
import tensorflow as tf
import numpy as np

#占位符，适用于不知道具体参数的时候
x = tf.placeholder(tf.float32, shape=(4, 4))
y = tf.add(x, x)
# [1,  32, 44, 56]
# [89, 12, 90, 33]
# [35, 69, 1,  10]
argmax_paramter = tf.Variable([[1, 32, 44, 56], [89, 12, 90, 33], [35, 69, 1, 10]])

#最大列索引
argmax_0 = tf.argmax(argmax_paramter, 0)
#最大行索引
argmax_1 = tf.argmax(argmax_paramter, 1)

#平均数
reduce_0 = tf.reduce_mean(argmax_paramter, reduction_indices=0)
reduce_1 = tf.reduce_mean(argmax_paramter, reduction_indices=1)

#相等
equal_0 = tf.equal(1,2)
equal_1 = tf.equal(2,2)

#类型转换
cast_0 = tf.cast(equal_0,tf.int32)
cast_1 = tf.cast(equal_1,tf.float32)
```

```
with tf.Session() as sess:
    init = tf.global_variables_initializer();
    sess.run(init)

    rand_array = np.random.rand(4, 4)
    print(sess.run(y, feed_dict={x: rand_array}))

    print("argmax_0:", sess.run(argmax_0))
    print("argmax_1:", sess.run(argmax_1))
    print("reduce_0:", sess.run(reduce_0))
    print("reduce_1:", sess.run(reduce_1))
    print("equal_0:", sess.run(equal_0))
    print("equal_1:", sess.run(equal_1))
    print("cast_0:", sess.run(cast_0))
    print("cast_1:", sess.run(cast_1))
```

MNIST手写数字识别实践

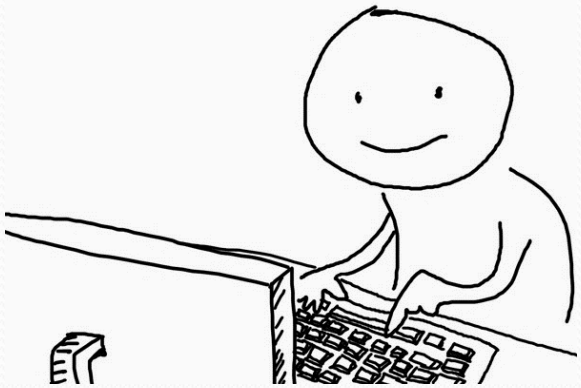
预备知识

tf.argmax　tf.reduce_mean　tf.equal　tf.cast

运行课程代码并分析结果：

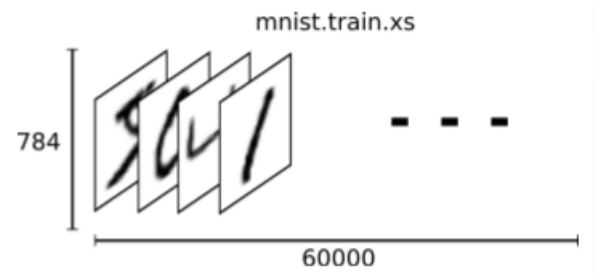
cell2

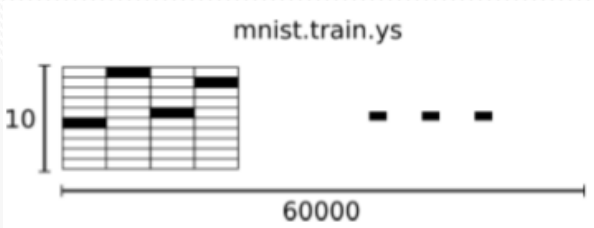
Machine_Learning\03_MNIST\01_logistic_regression.ipynb



MNIST手写数字识别实践

逻辑回归方法 Part1






```python
# 变量
batch_size = 100

#训练的x(image),y(label)
# x = tf.Variable()
# y = tf.Variable()
x = tf.placeholder(tf.float32, [None, 784])
y = tf.placeholder(tf.float32, [None, 10])

# 模型权重
#[55000,784] * W = [55000,10]
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))

# 用softmax构建逻辑回归模型
pred = tf.nn.softmax(tf.matmul(x, W) + b)

# 损失函数(交叉熵)
cost = tf.reduce_mean(-tf.reduce_sum(y*tf.log(pred), 1))

# 低度下降
optimizer = tf.train.GradientDescentOptimizer(0.01).minimize(cost)

# 初始化所有变量
init = tf.global_variables_initializer()
```

Softmax
Cross entropy？

课程代码：cell2　Machine_Learning\03_MNIST\01_logistic_regression.ipynb

MNIST手写数字识别实践

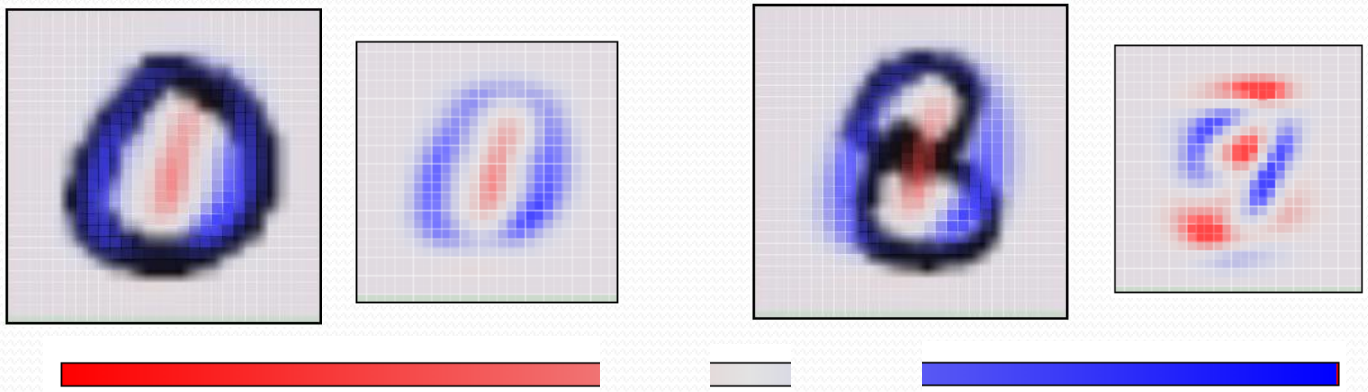
Softmax regression 模型可以用来给不同的对象分配概率

Step1：计算输入变量x属于某一分类的“证据”

$$\text{evidence}_i = \sum_j W_{i,j} x_j + b_i$$

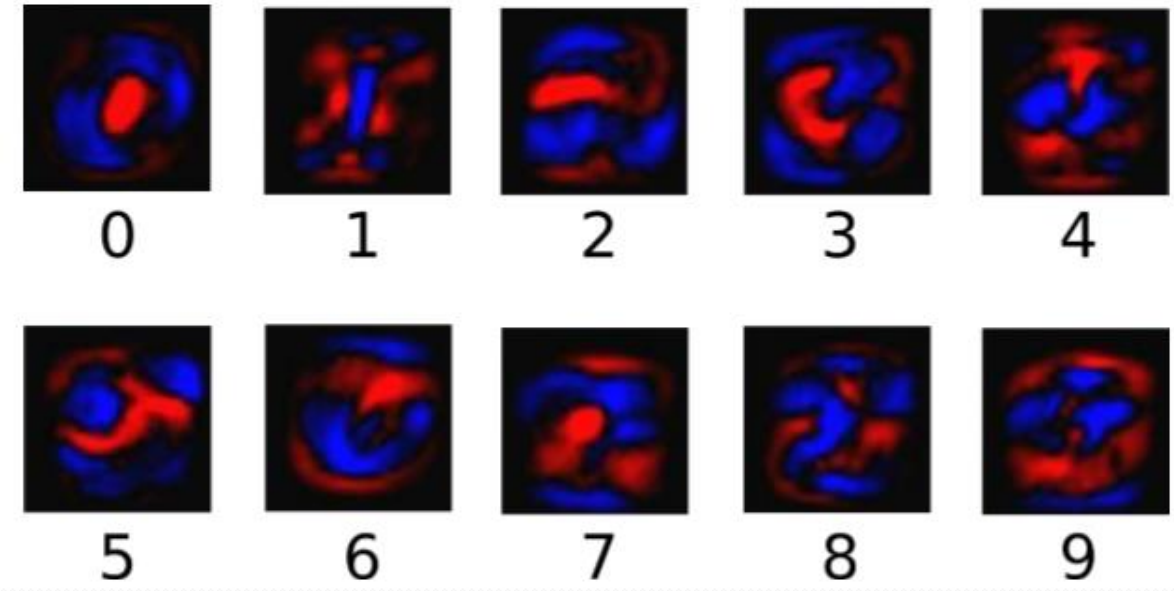
其中$W_{i,j}$代表该输入变量的权重，b_i代表数字i 类的偏置量或者干扰量，j代表累计求和，对MNIST图像而言，x_j为每一个像素点的灰度值

即：第一步是对图片像素值进行加权求和，如果这个像素具有很强的证据说明这张图片不属于该类，那么相应的权值为负数，相反如果这个像素拥有有利的证据支持这张图片属于这个类，那么权值是正数。



http://colah.github.io/posts/2014-10-Visualizing-MNIST/

MNIST手写数字识别实践



Step2：把“证据” 转换成概率 y

$$y = \text{softmax}(\text{evidence})$$

$$\text{softmax}(x) = \text{normalize}(\exp(x))$$

$$= \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

MNIST手写数字识别实践

1.逻辑回归

$$P(i) = \frac{1}{1 + exp(-\theta_i^T x)}$$

$\theta_i^T x$:代表线性回归函数

2.softmax

$$P(i) = \frac{exp(\theta_i^T x)}{\sum_{k=1}^{K} exp(\theta_k^T x)}$$

softmax是逻辑回归的泛化

当类别数k = 2 时，softmax 回归退化为逻辑回归

$$h_\theta(x) = \frac{1}{e^{\theta_1^T x} + e^{\theta_2^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x} \\ e^{\theta_2^T x} \end{bmatrix}$$

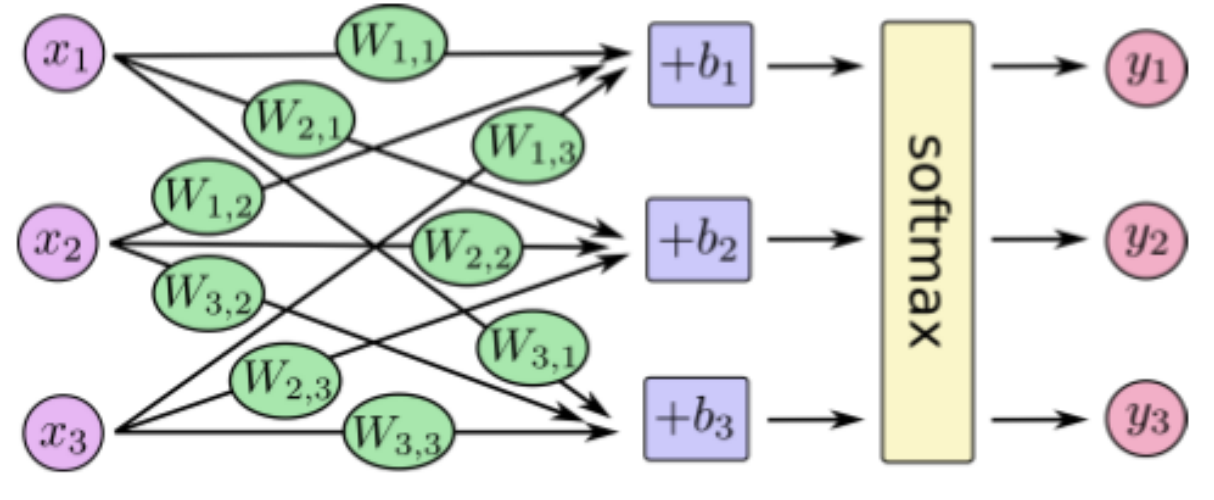
从两个参数向量中都减去向量 θ_1 ，得到:

$$h(x) = \frac{1}{e^{\vec{0}^T x} + e^{(\theta_2 - \theta_1)^T x^{(i)}}} \begin{bmatrix} e^{\vec{0}^T x} \\ e^{(\theta_2 - \theta_1)^T x} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{1+e^{(\theta_2-\theta_1)^T x^{(i)}}} \\ \frac{e^{(\theta_2-\theta_1)^T x}}{1+e^{(\theta_2-\theta_1)^T x^{(i)}}} \end{bmatrix} = \begin{bmatrix} \frac{1}{1+e^{(\theta_2-\theta_1)^T x^{(i)}}} \\ 1 - \frac{1}{1+e^{(\theta_2-\theta_1)^T x^{(i)}}} \end{bmatrix}$$

MNIST手写数字识别实践

1.逻辑回归 Part1代码 的构图形式



$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax}\left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

MNIST手写数字识别实践

逻辑回归方法 Part1

```
# 变量
batch_size = 100

#训练的x(image),y(label)
# x = tf.Variable()
# y = tf.Variable()
x = tf.placeholder(tf.float32, [None, 784])
y = tf.placeholder(tf.float32, [None, 10])

# 模型权重
#[55000,784] * W = [55000,10]
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))

# 用softmax构建逻辑回归模型
pred = tf.nn.softmax(tf.matmul(x, W) + b)

# 损失函数(交叉熵)
cost = tf.reduce_mean(-tf.reduce_sum(y*tf.log(pred), 1))

# 低度下降
optimizer = tf.train.GradientDescentOptimizer(0.01).minimize(cost)

# 初始化所有变量
init = tf.global_variables_initializer()
```

如果f（.）是非线性的激活函数（activation function），这就是一个分类模型；如果f（.）是恒等函数（identity），则是回归模型

$$y(\mathbf{x}, \mathbf{w}) = f\left(\sum_{j=1}^{M} w_j \phi_j(\mathbf{x})\right)$$

经验基函数一致

课程代码：cell2　Machine_Learning\03_MNIST\01_logistic_regression.ipynb

MNIST手写数字识别实践

逻辑回归方法 Part2

```
# 加载session图
with tf.Session() as sess:
    sess.run(init)

    # 开始训练
    for epoch in range(25):
        avg_cost = 0.

        total_batch = int(mnist.train.num_examples/batch_size)
        for i in range(total_batch):
            batch_xs, batch_ys = mnist.train.next_batch(batch_size)
            sess.run(optimizer, {x: batch_xs,y: batch_ys})
            #计算损失平均值
            avg_cost += sess.run(cost,{x: batch_xs,y: batch_ys}) / total_batch
        if (epoch+1) % 5 == 0:
            print("Epoch:", '%04d' % (epoch+1), "cost=", "{:.9f}".format(avg_cost))

    print("运行完成")

    # 测试求正确率
    correct = tf.equal(tf.argmax(pred, 1), tf.argmax(y, 1))
    accuracy = tf.reduce_mean(tf.cast(correct, tf.float32))
    print("正确率:", accuracy.eval({x: mnist.test.images, y: mnist.test.labels}))
```

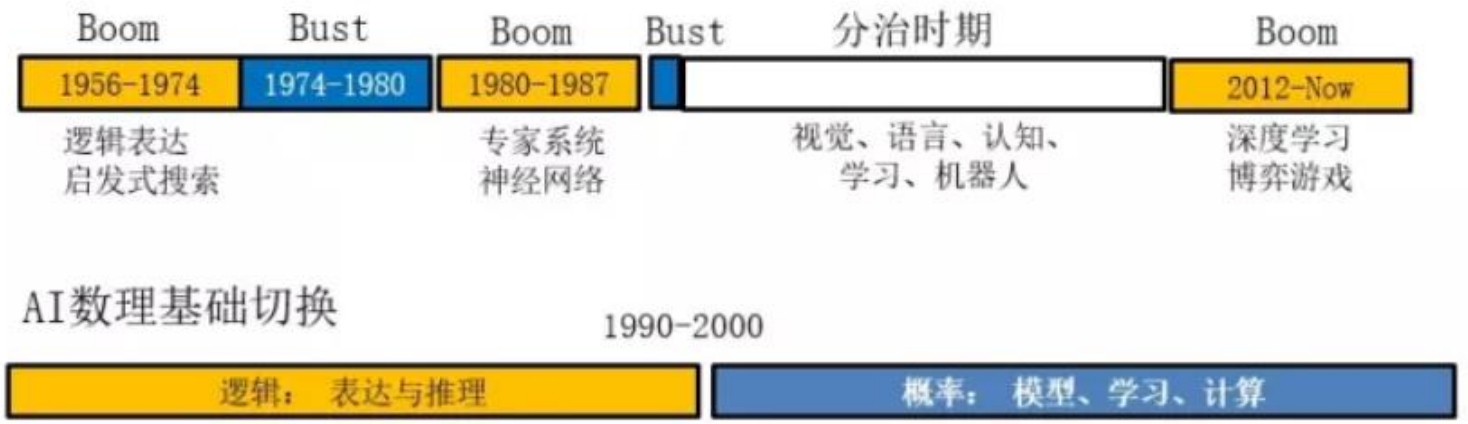
课程代码：cell2　Machine_Learning\03_MNIST\01_logistic_regression.ipynb

目录

- MNIST数据集
- MNIST手写数字识别实践
  - 逻辑回归方法
  - 人工神经网络方法
  - 卷积神经网络CNN方法
- 调试技巧： TensorFlow可视化
- 小结

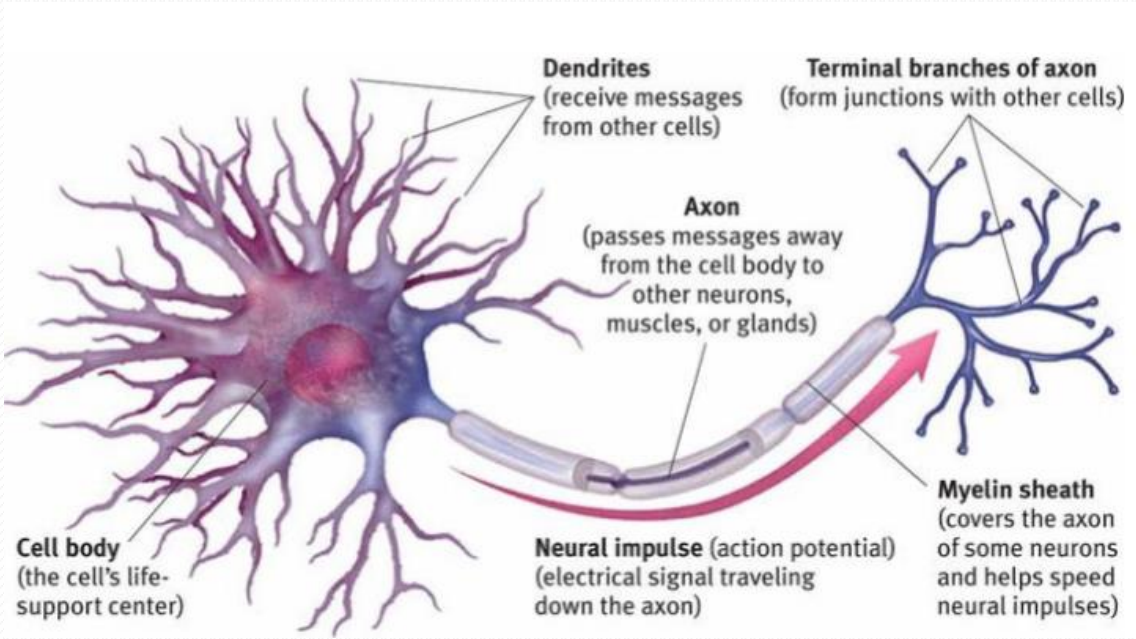
MNIST手写数字识别实践

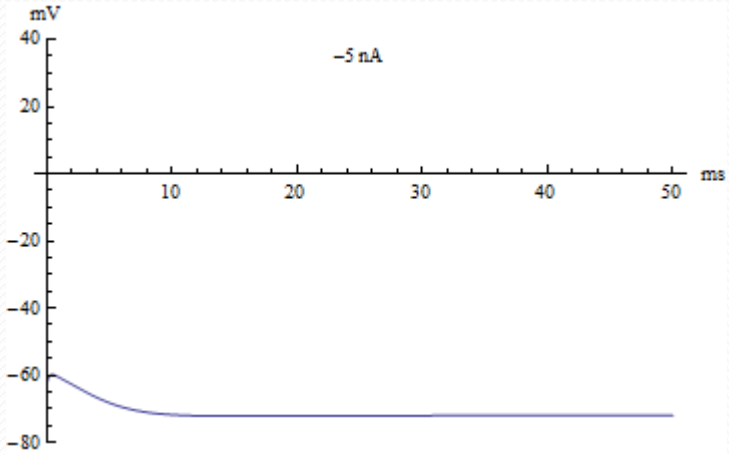
人工智能发展背景



MNIST手写数字识别实践

人工神经网络方法





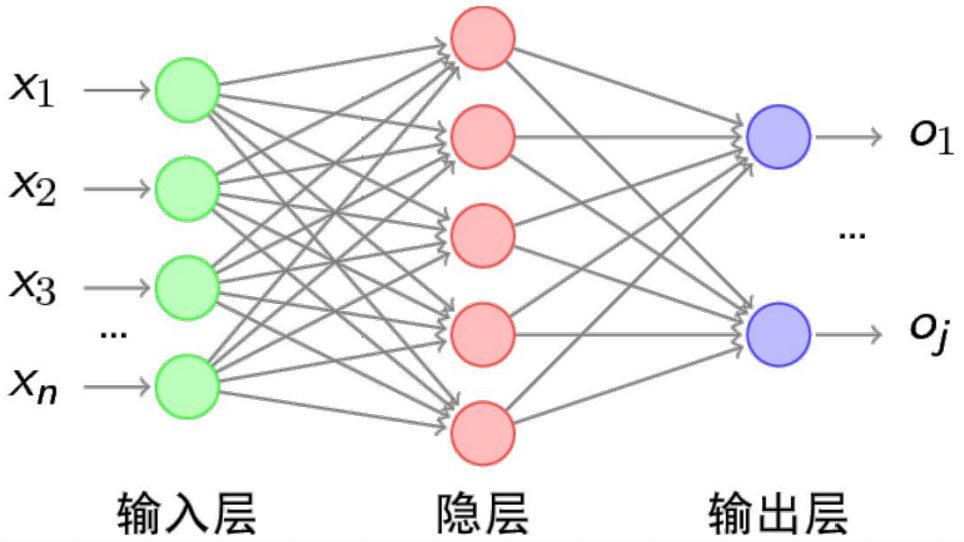
Hodgkin-Huxley model

MNIST手写数字识别实践

人工神经网络方法

$$E = \sum_{k} (h_k - y_k)^2$$

h is the output under current weights.
y is the labelled data of current input.

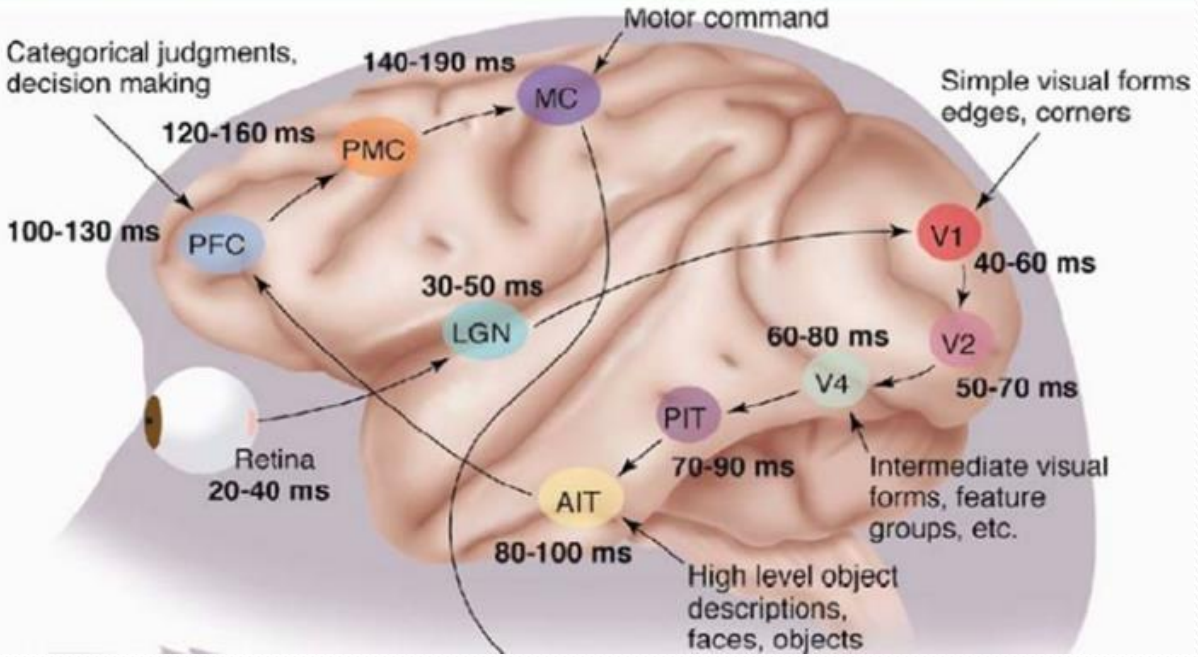


扩展——为什么采用层次网络结构

- **人脑视觉机理**

✓ 诺贝尔医学奖获得者 David Hubel和TorstenWiesel发现了视觉系统的信息处理机制

✓ 发现了一种被称为“方向选择性细胞的神经元细胞，当瞳孔发现了眼前的物体的边缘，而且这个边缘指向某个方向时，这种神经元细胞就会活跃



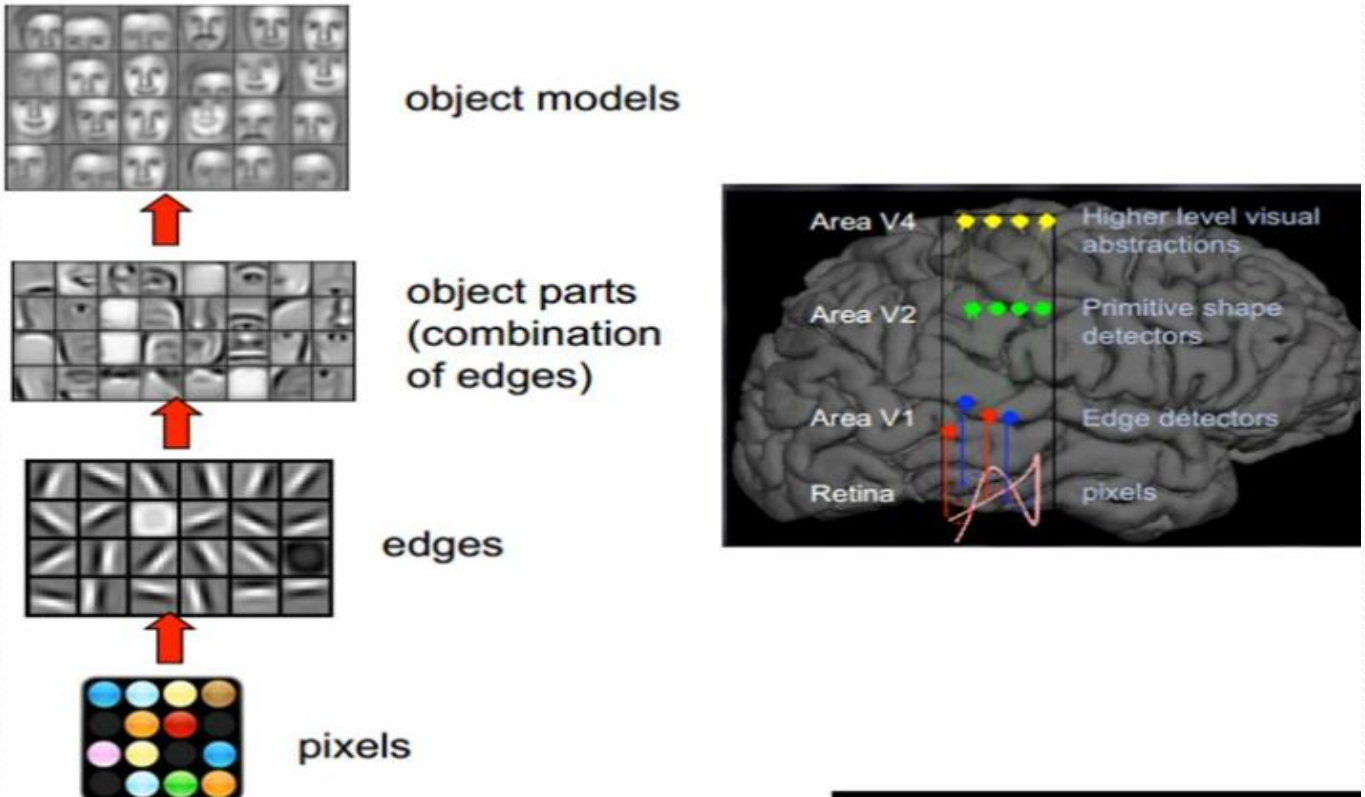
扩展——为什么采用层次网络结构

- **人脑视觉机理**

✓ 人的视觉系统的信息处理是分级的

✓ 高层的特征是低层特征的组合，从低层到高层的特征表示越来越抽象，越来越能表现语义或者意图

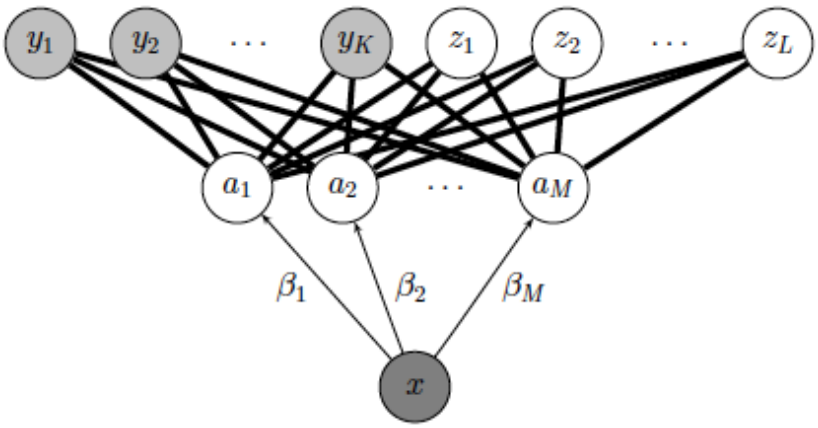
✓ 抽象层面越高，存在的可能猜测就越少，就越利于分类



扩展——为什么采用层次网络结构

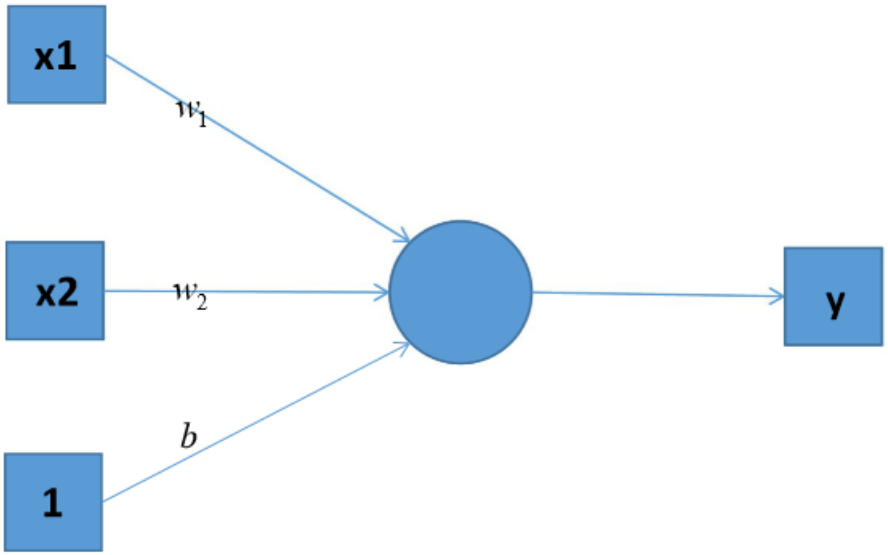
- 视觉的层次性
- ✓ 属性学习，类别作为属性的一种组合映射

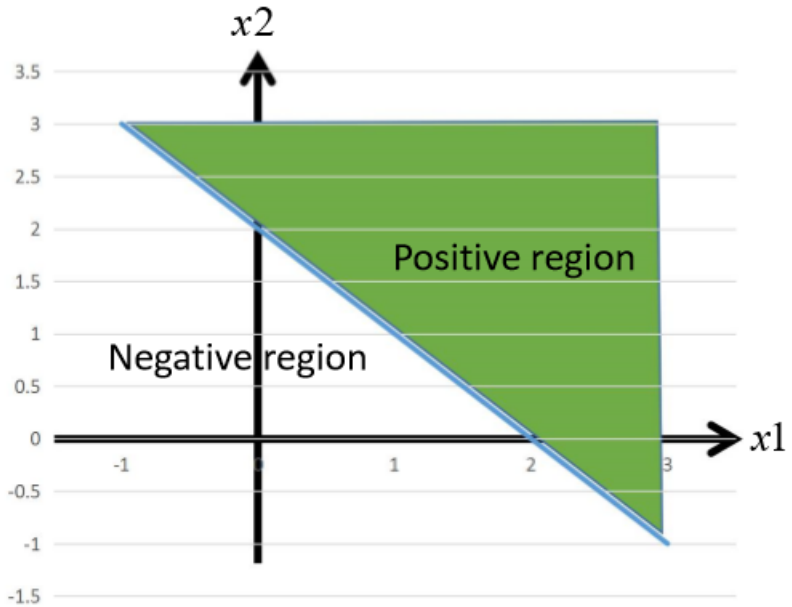
  Lampert et al. CVPR’09



MNIST手写数字识别实践

Perceptron





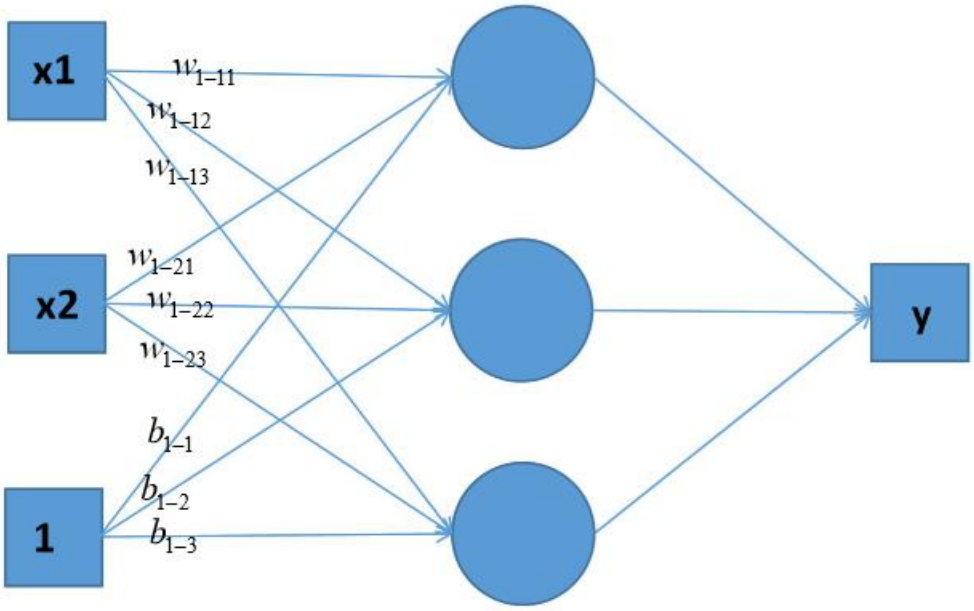
$$w_1 = 1, w_2 = 1, b = -2$$

$$y = w_1 x_1 + w_2 x_2 + b$$

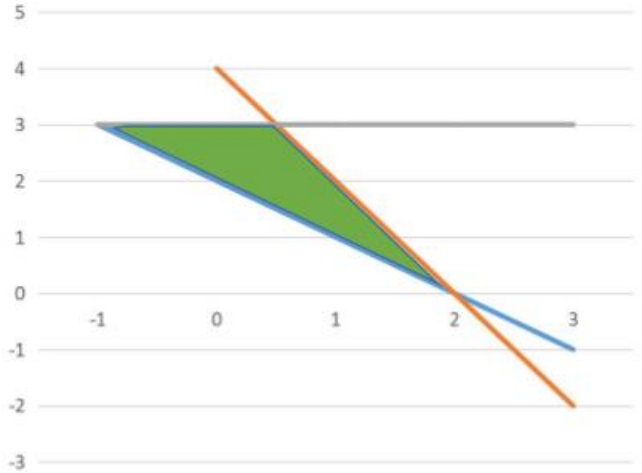
Image source: Course Note of Artificial Neural Network, KU Leuven, Johan Suykens

MNIST手写数字识别实践

Perceptron



linear combination of three decision lines



$$w_{1-11} = 1, w_{1-12} = 1, b_{1-1} = -2$$

$$w_{1-21} = 2, w_{1-22} = 1, b_{1-2} = 4$$

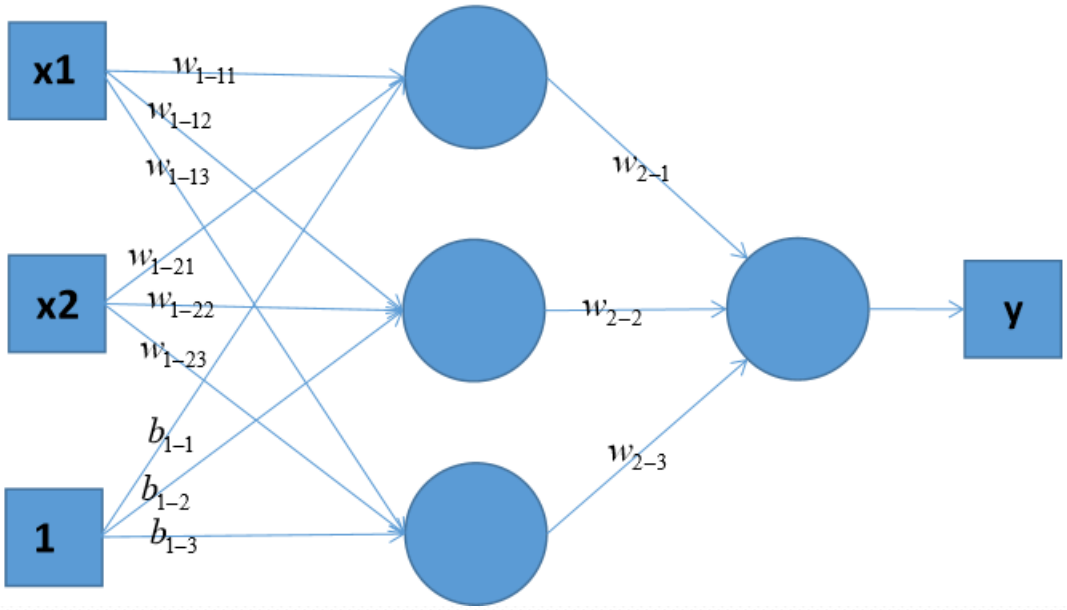
$$w_{1-31} = 0, w_{1-32} = 1, b_{1-3} = 3$$

Image source: Course Note of Artificial Neural Network, KU Leuven, Johan Suykens

MNIST手写数字识别实践

感知机串接1层隐含层

Perceptron with one hidden layer



$$y = w_{2\text{-}1}(w_{1\text{-}11}x_1 + w_{1-21}x_2 + b_{1\text{-}1}) + w_{2\text{-}2}(w_{1\text{-}12}x_1 + w_{1-22}x_2 + b_{1\text{-}2}) + w_{2\text{-}3}(w_{1\text{-}13}x_1 + w_{1-23}x_2 + b_{1\text{-}3})$$

Image source: Course Note of Artificial Neural Network, KU Leuven, Johan Suykens

MNIST手写数字识别实践

感知机串接1层隐含层 1个激活函数

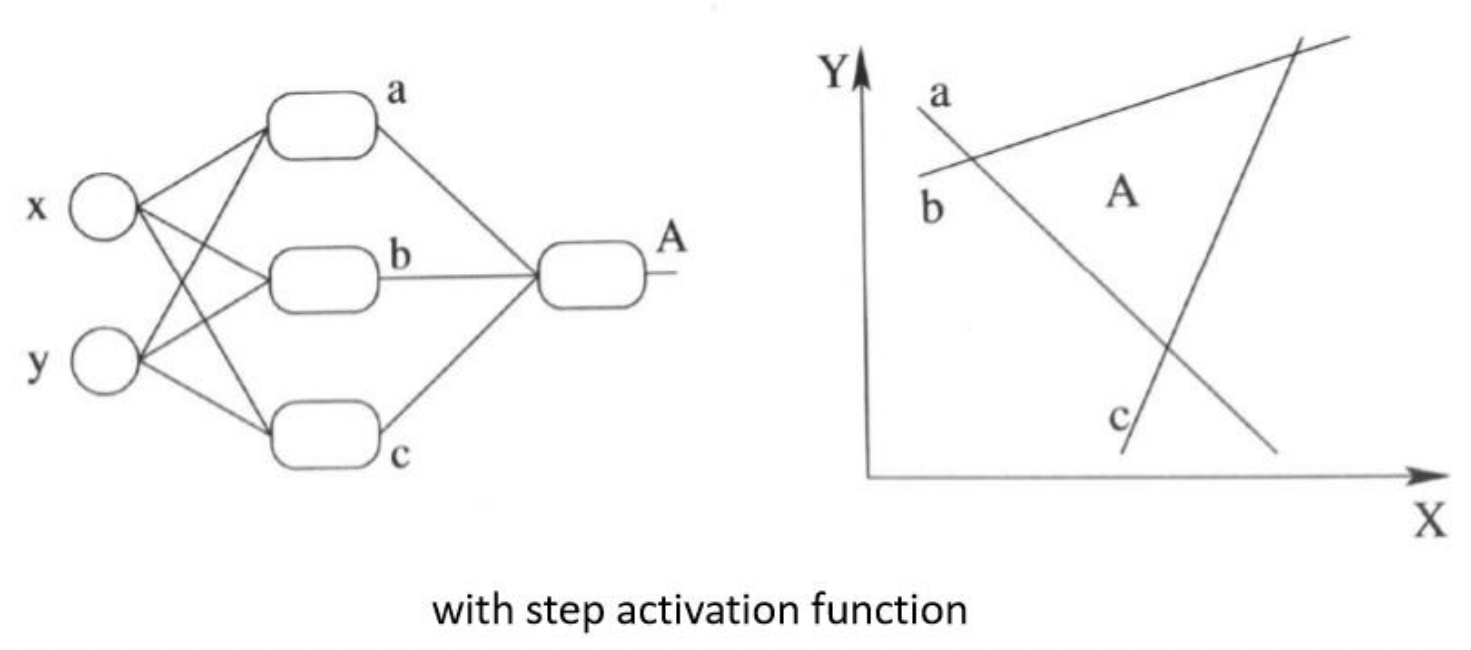
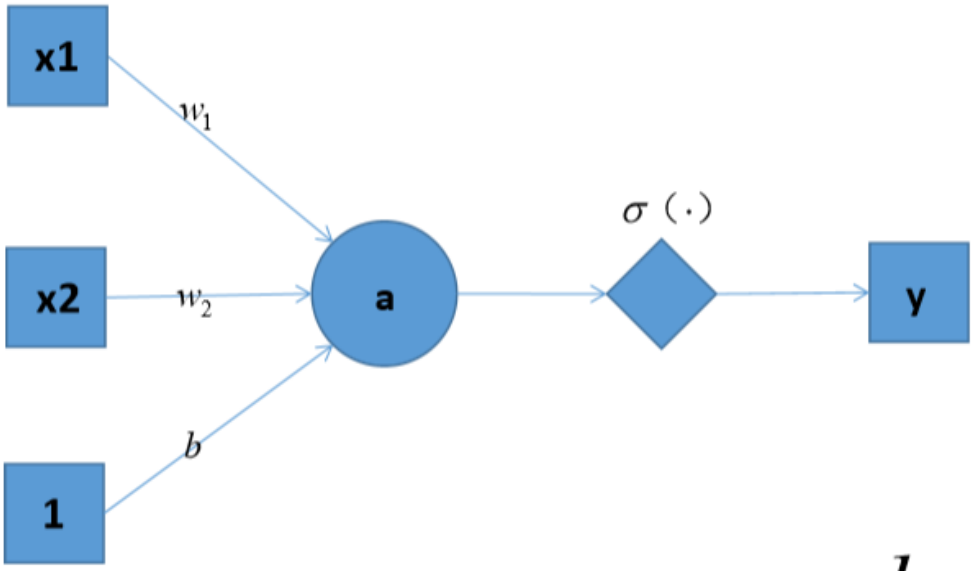
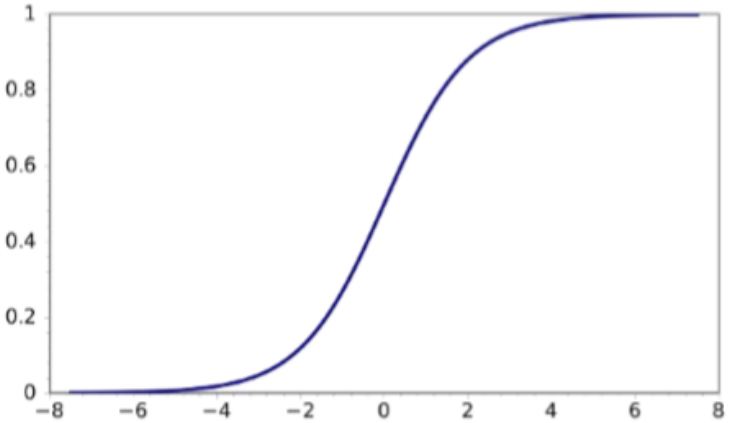


Image source: Course Note of Artificial Neural Network, KU Leuven, Johan Suykens

MNIST手写数字识别实践

感知机串接1层隐含层 1个非线性激活函数





$$a = w_1 x_1 + w_2 x_2 + b$$

$$y = \sigma(a)$$

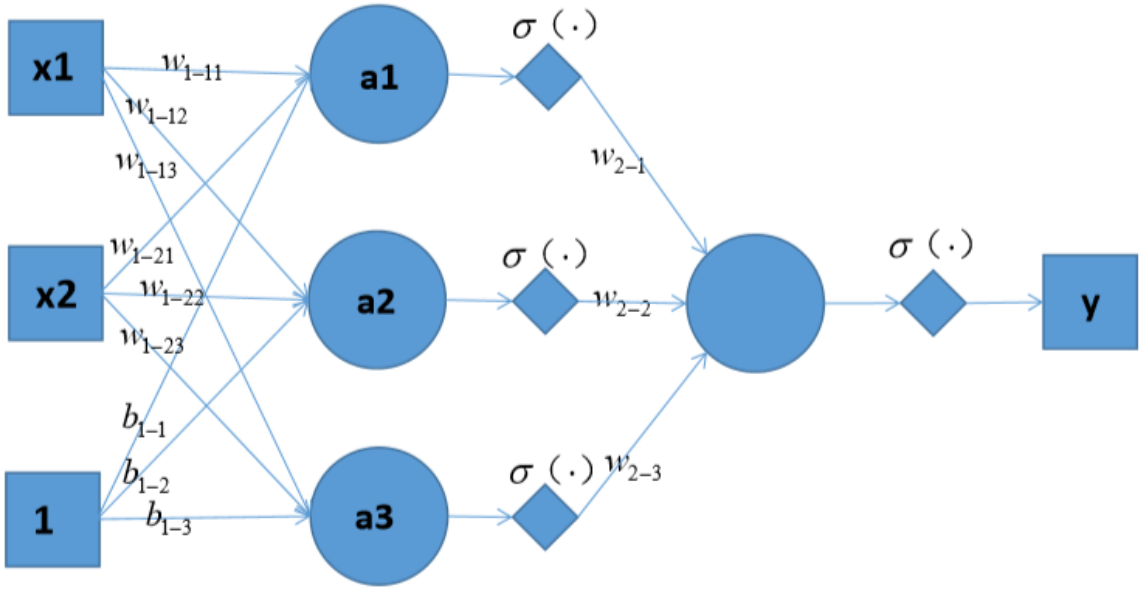
$\sigma(\cdot)$ is a non-linear activation function, sigmoid was the most popular one,

$$\sigma(y) = \frac{1}{1+e^{-y}}$$

Image source: Course Note of Artificial Neural Network, KU Leuven, Johan Suykens

MNIST手写数字识别实践

感知机串接1层隐含层 1个非线性激活函数



$$a1 = w_{1\text{-}11}x_1 + w_{1-21}x_2 + b_{1\text{-}1}$$

$$a2 = w_{1\text{-}12}x_1 + w_{1-22}x_2 + b_{1\text{-}2}$$

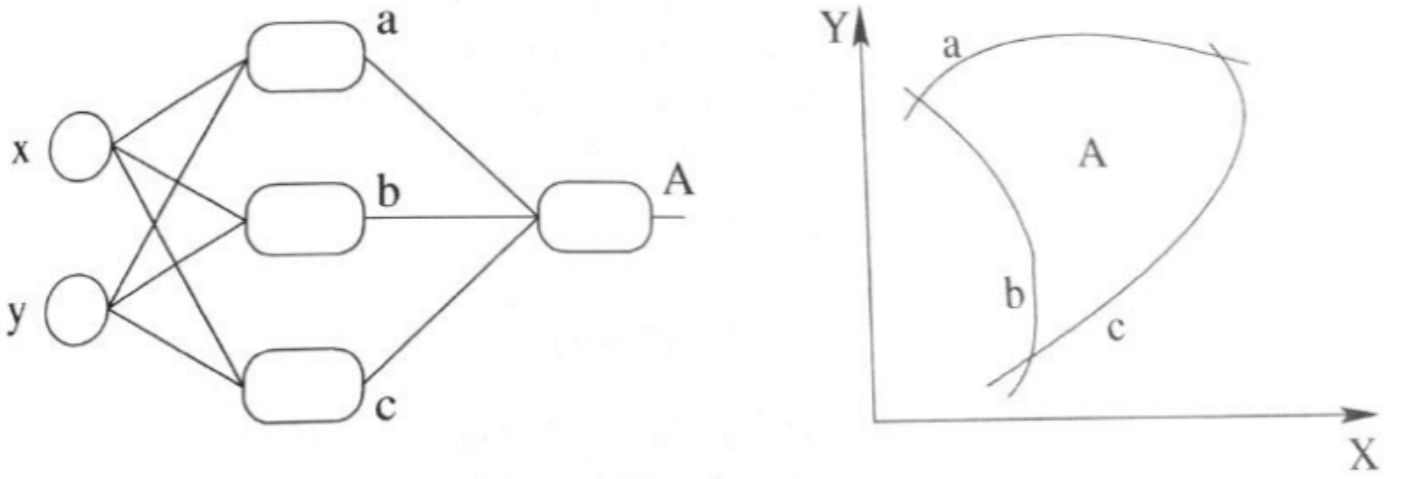
$$a3 = w_{1\text{-}13}x_1 + w_{1-23}x_2 + b_{1\text{-}3}$$

$$y = \sigma(w_{2-1}\sigma(a1) + w_{2-2}\sigma(a2) + w_{2-3}\sigma(a3))$$

Image source: Course Note of Artificial Neural Network, KU Leuven, Johan Suykens

MNIST手写数字识别实践

感知机串接1层隐含层 1个非线性激活函数



with sigmoid activation function

Image source: Course Note of Artificial Neural Network, KU Leuven, Johan Suykens

MNIST手写数字识别实践

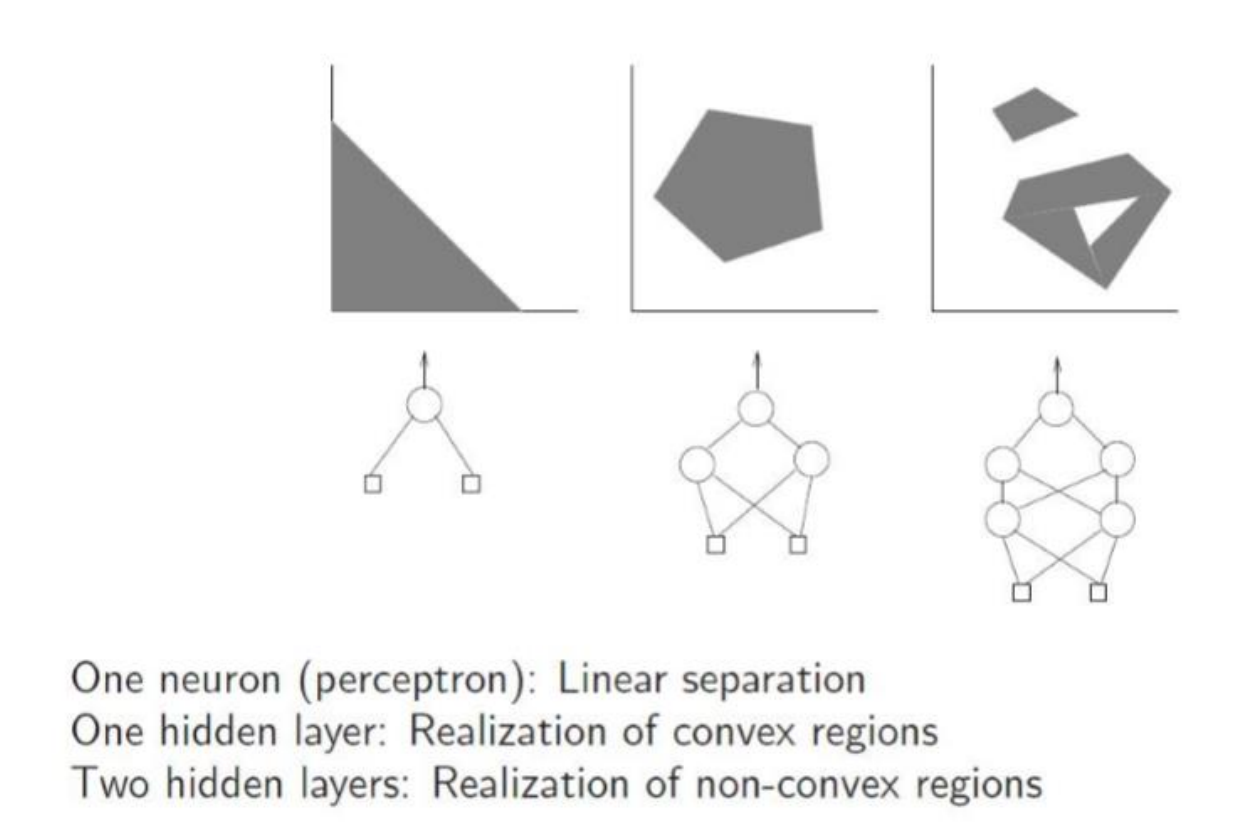
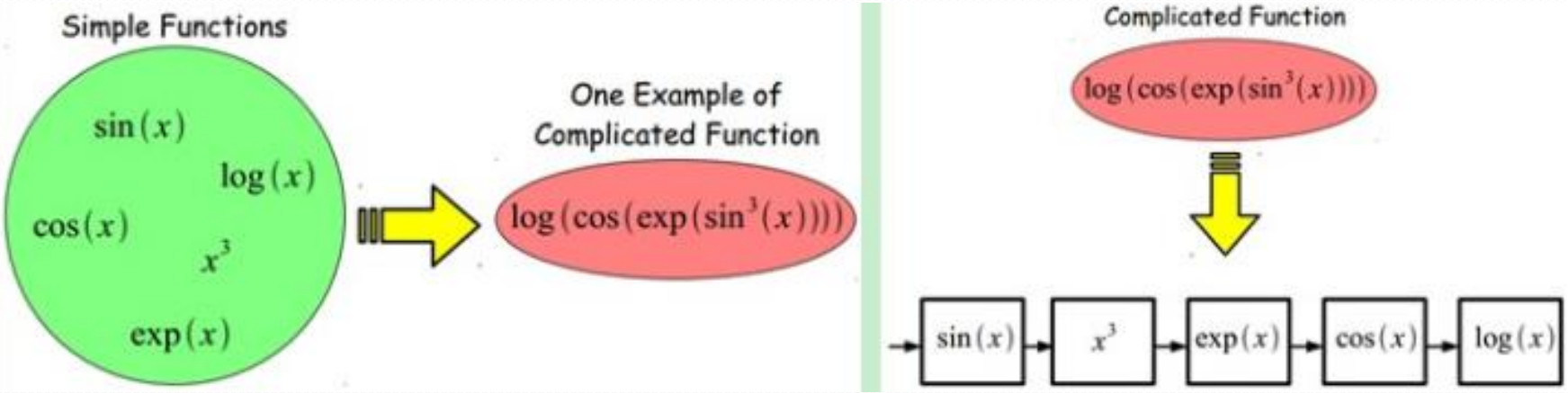


Image source: Course Note of Artificial Neural Network, KU Leuven, Johan Suykens

MNIST手写数字识别实践

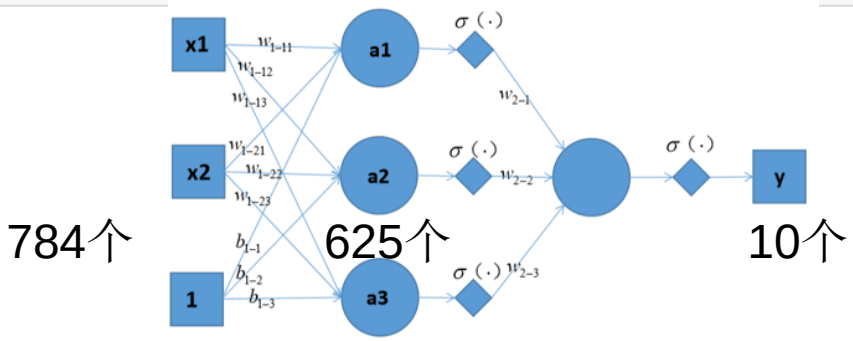
- 神经网络：可通过学习一种多层非线性网络结构，实现复杂函数逼近，表征输入数据分布式表示。



MNIST手写数字识别实践

```
X = tf.placeholder("float", [None, 784])
Y = tf.placeholder("float", [None, 10])

w_h = init_weights([784, 625]) # create symbolic variables
w_o = init_weights([625, 10])

py_x = model(X, w_h, w_o)

cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=py_x, labels=Y))
train_op = tf.train.GradientDescentOptimizer(0.05).minimize(cost) # construct an opti
predict_op = tf.argmax(py_x, 1)
# Launch the graph in a session
with tf.Session() as sess:
    # you need to initialize all variables
    tf.global_variables_initializer().run()

    for i in range(10):
        for start, end in zip(range(0, len(trX), 128), range(128, len(trX)+1, 128)):
            sess.run(train_op, feed_dict={X: trX[start:end], Y: trY[start:end]})
        print(i, np.mean(np.argmax(teY, axis=1) ==
                         sess.run(predict_op, feed_dict={X: teX})))
```

```
0 0.6923
1 0.8211
2 0.8621
3 0.8795
4 0.8887
5 0.8929
6 0.8976
7 0.9006
8 0.9034
9 0.905
```



课程代码：Machine_Learning\03_MNIST\03_net.ipynb

目录

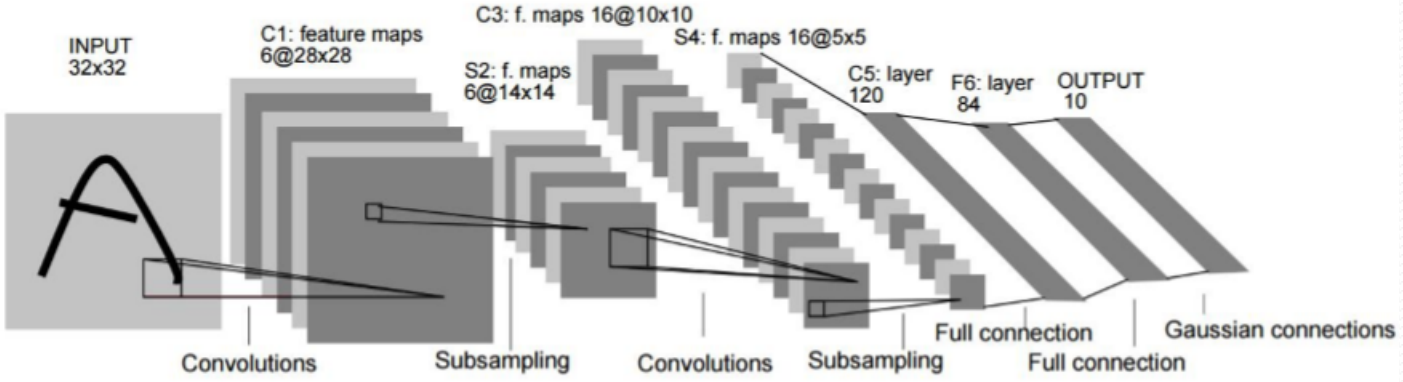
- MNIST数据集
- MNIST手写数字识别实践
  - 逻辑回归方法
  - 人工神经网络方法
  - 卷积神经网络CNN方法
- 调试技巧： TensorFlow可视化
- 小结

MNIST手写数字识别实践

卷积神经网络简介

- 来历：
  Hubel和Wiesel在研究猫脑皮层中用于局部敏感和方向选择的神经元时发现其独特的网络结构可以有效地降低反馈神经网络的复杂性，继而提出了卷积神经网络
- 经典CNN模型：LeNet

Designed for 2-dimensional object recognition, take the spatial information into account.



Basic types of layers:

1. convolution layer: for feature extraction
2. sub-sampling layer: for simplifying feature, prevent overfitting.
3. fully connected layer: for final classification.

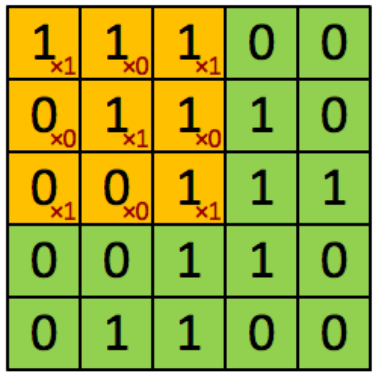
课程论文lecun-98_Gradient-Based Learning Applied to Document Recognition.pdf

MNIST手写数字识别实践

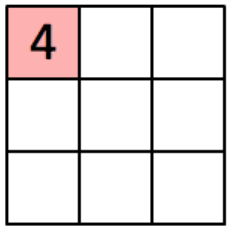
卷积神经网络-卷积层

卷积核大小、
数量？
层数？

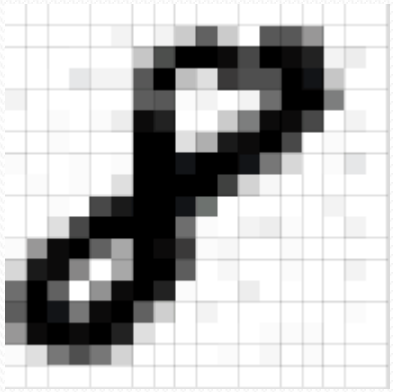
卷积的主要目的是为了从输入图像中提取特征。
卷积可以通过从输入的一小块数据中学到图像的特征，并可以保留像素间的空间关系。



Image



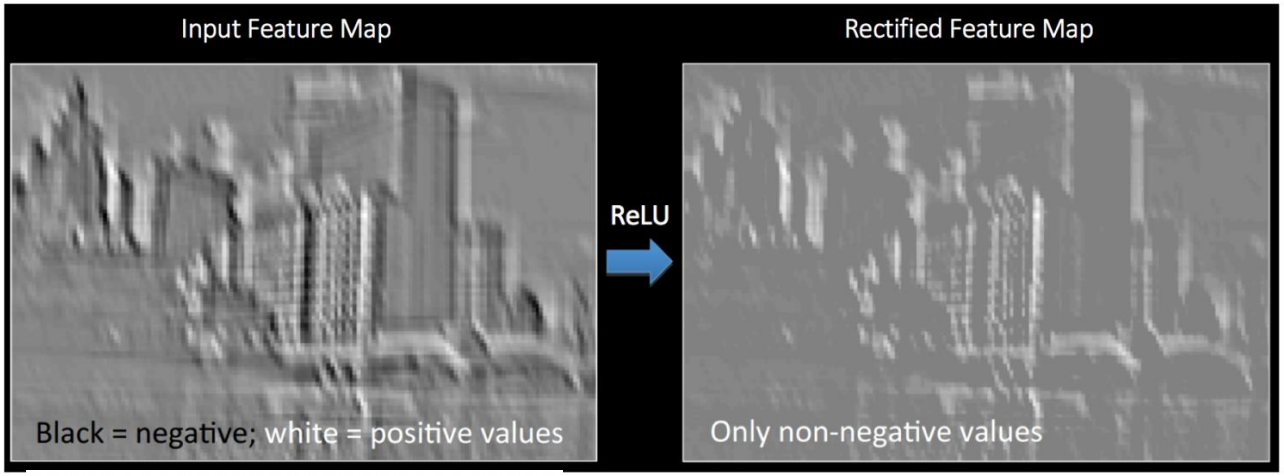
Convolved
Feature

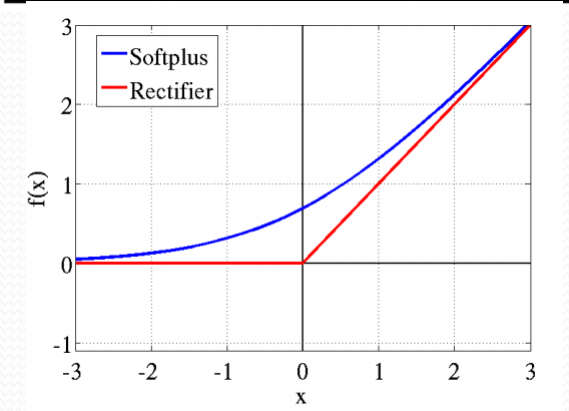


MNIST手写数字识别实践

卷积神经网络-激活函数

ReLU 是一个元素级别的操作（应用到各个像素），并将特征图中的所有小于 0 的像素值设置为零，ReLU 的目的是在 ConvNet 中引入非线性。





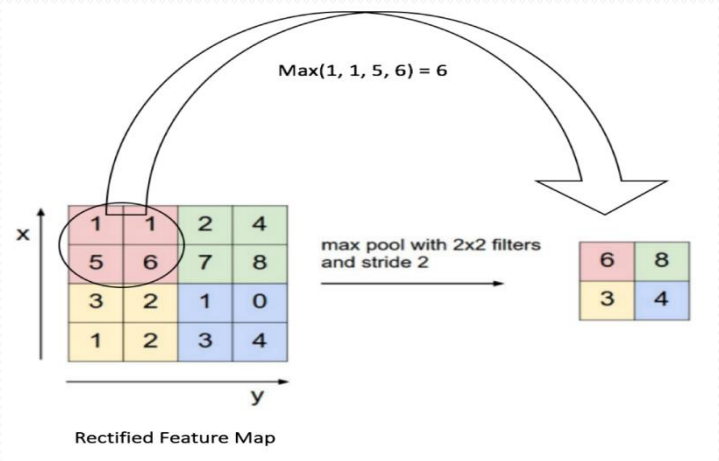
Y = Max(0，X)

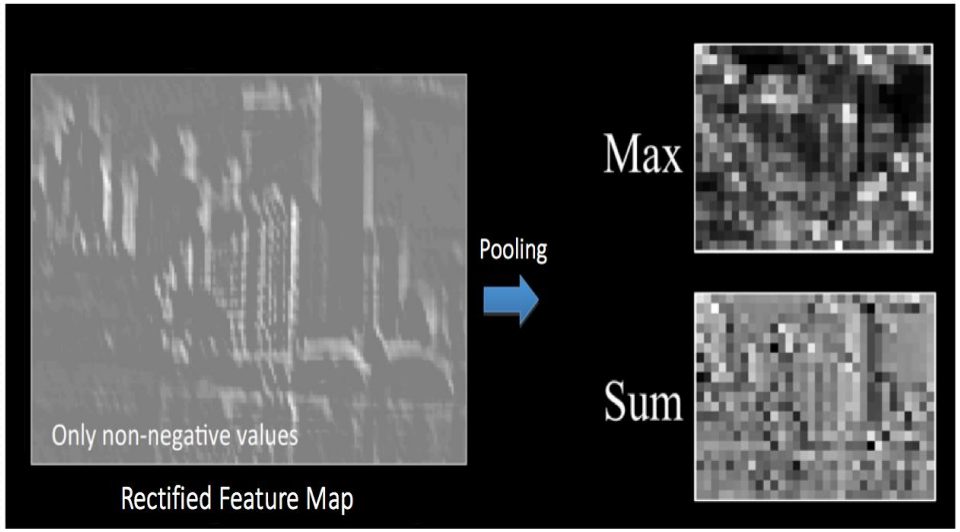
Xavier Glorot.Deep Sparse Rectifier Neural Networks,AISTAS,2011

MNIST手写数字识别实践

卷积神经网络-池化层

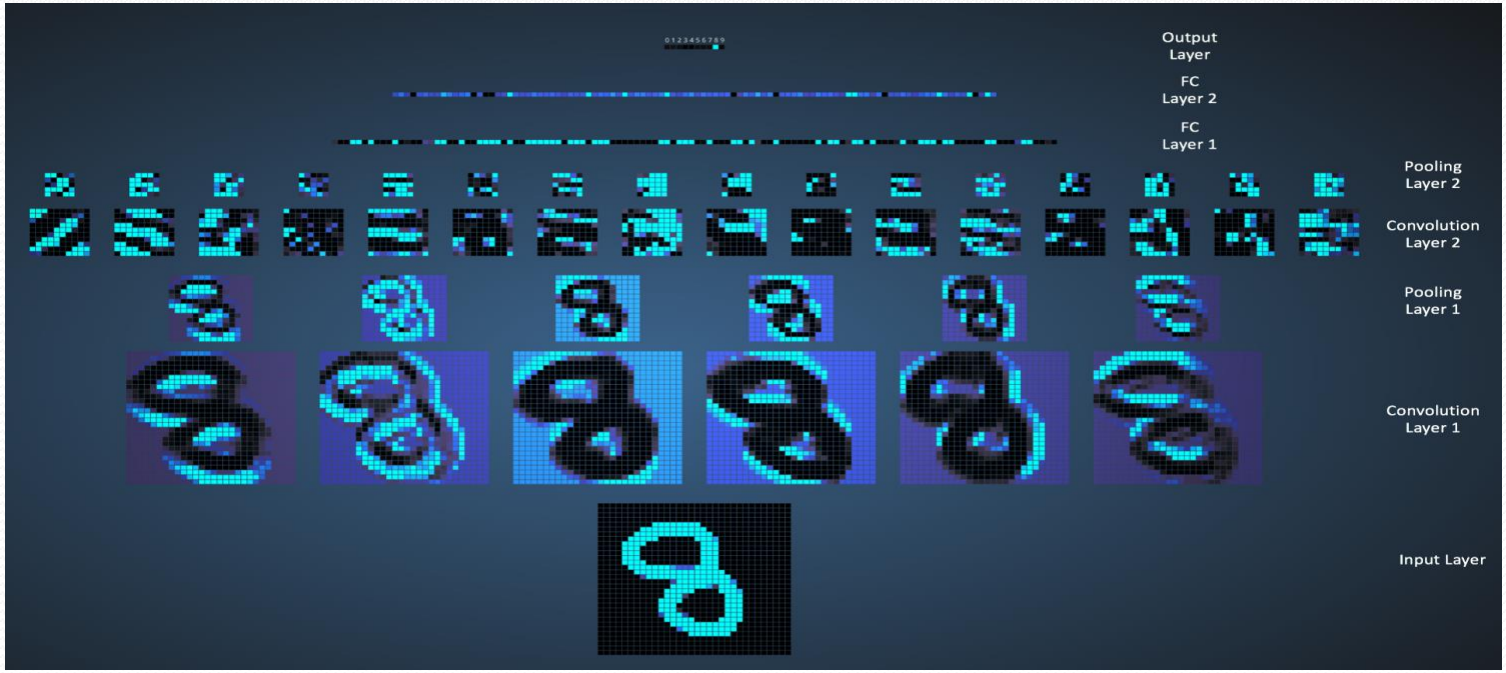
池化层：降低了各个特征图的维度，但可以保持大部分重要的信息；
空间池化几种常见方式：最大化、平均化、加和等；
同时还可以抑制 过拟合；





MNIST手写数字识别实践

卷积神经网络-Mnist训练步骤



3D网址:http://scs.ryerson.ca/~aharley/vis/conv/

MNIST手写数字识别实践

卷积神经网络方法

```
def model(X, w, w2, w3, w4, w_o, p_keep_conv, p_keep_hidden):
    l1a = tf.nn.relu(tf.nn.conv2d(X, w,                           # l1a shape=(?, 28, 28, 32)
                        strides=[1, 1, 1, 1], padding='SAME'))
    l1 = tf.nn.max_pool(l1a, ksize=[1, 2, 2, 1],                  # l1 shape=(?, 14, 14, 32)
                        strides=[1, 2, 2, 1], padding='SAME')
    l1 = tf.nn.dropout(l1, p_keep_conv)

    l2a = tf.nn.relu(tf.nn.conv2d(l1, w2,                         # l2a shape=(?, 14, 14, 64)
                        strides=[1, 1, 1, 1], padding='SAME'))
    l2 = tf.nn.max_pool(l2a, ksize=[1, 2, 2, 1],                  # l2 shape=(?, 7, 7, 64)
                        strides=[1, 2, 2, 1], padding='SAME')
    l2 = tf.nn.dropout(l2, p_keep_conv)

    l3a = tf.nn.relu(tf.nn.conv2d(l2, w3,                         # l3a shape=(?, 7, 7, 128)
                        strides=[1, 1, 1, 1], padding='SAME'))
    l3 = tf.nn.max_pool(l3a, ksize=[1, 2, 2, 1],                  # l3 shape=(?, 4, 4, 128)
                        strides=[1, 2, 2, 1], padding='SAME')
    l3 = tf.reshape(l3, [-1, w4.get_shape().as_list()[0]])        # reshape to (?, 2048)
    l3 = tf.nn.dropout(l3, p_keep_conv)

    l4 = tf.nn.relu(tf.matmul(l3, w4))
    l4 = tf.nn.dropout(l4, p_keep_hidden)

    pyx = tf.matmul(l4, w_o)
    return pyx
```

课程代码：cell2　Machine_Learning\03_MNIST\ 05_convolutional_net.ipynb

MNIST手写数字识别实践

卷积神经网络方法

```
    for i in range(10):
        training_batch = zip(range(0, len(trX), batch_size),
                             range(batch_size, len(trX)+1, batch_size))
        for start, end in training_batch:
            sess.run(train_op, feed_dict={X: trX[start:end], Y: trY[start:end],
                                          p_keep_conv: 0.8, p_keep_hidden: 0.5})

        test_indices = np.arange(len(teX)) # Get A Test Batch
        np.random.shuffle(test_indices)
        test_indices = test_indices[0:test_size]

        print(i, np.mean(np.argmax(teY[test_indices], axis=1) ==
                         sess.run(predict_op, feed_dict={X: teX[test_indices],
                                                         Y: teY[test_indices],
                                                         p_keep_conv: 1.0,
                                                         p_keep_hidden: 1.0})))
```

```
0 0.93359375
1 0.984375
2 0.98828125
3 0.98828125
4 0.98828125
5 0.9921875
6 0.99609375
7 1.0
8 1.0
9 1.0
```

课程代码：cell2　Machine_Learning\03_MNIST\ 05_convolutional_net.ipynb

MNIST手写数字识别实践

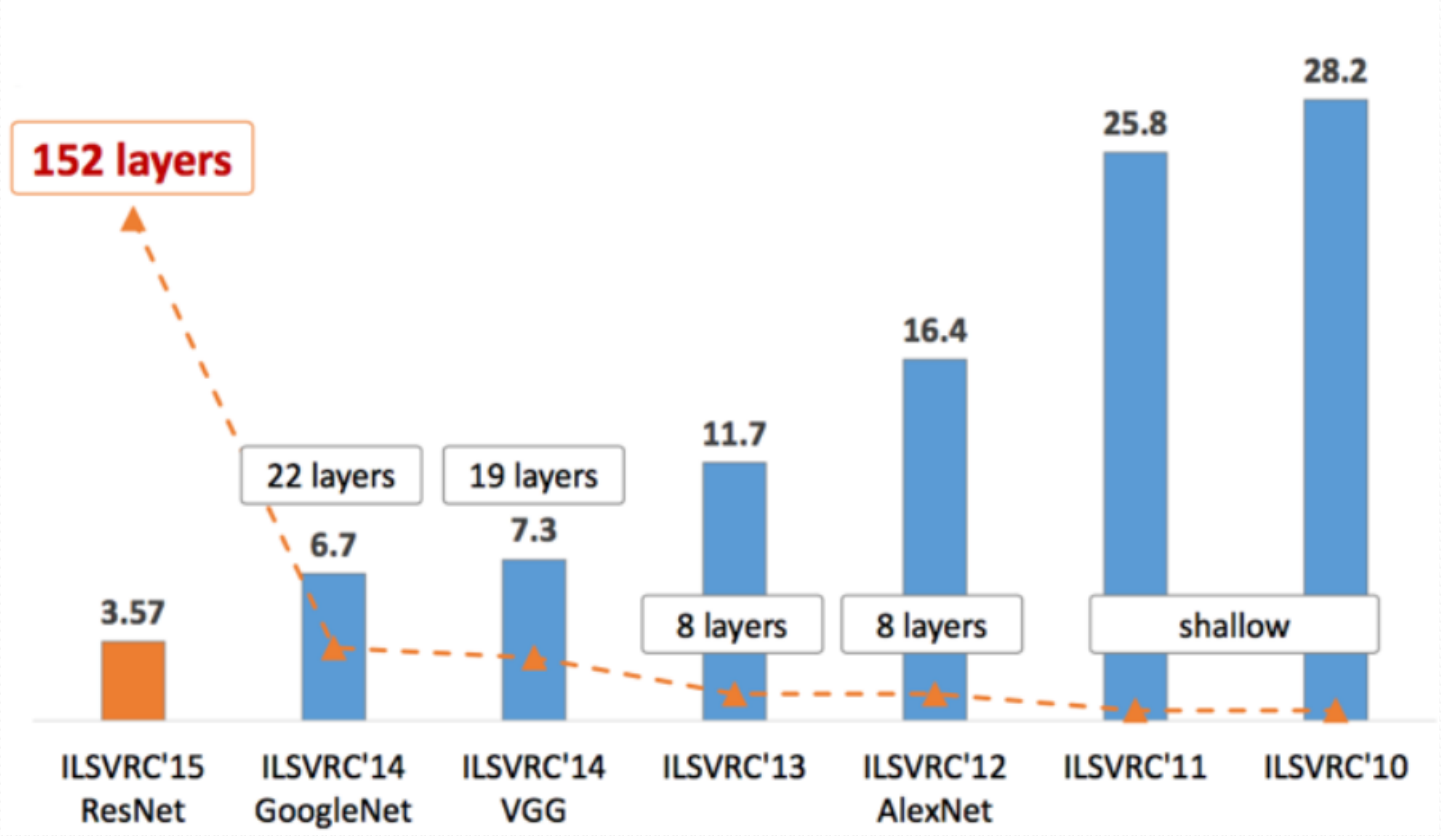
卷积神经网络源码目录

https://github.com/tensorflow/models/tree/master/research/slim/nets

- __init__.py
- alexnet.py
- alexnet_test.py
- cifarnet.py
- cyclegan.py
- cyclegan_test.py
- dcgan.py
- dcgan_test.py
- inception.py
- pix2pix.py
- pix2pix_test.py
- resnet_utils.py
- resnet_v1.py
- resnet_v1_test.py
- resnet_v2.py
- resnet_v2_test.py
- vgg.py
- vgg_test.py
- inception_v4.py
- inception_v4_test.py
- lenet.py
- mobilenet_v1.md
- mobilenet_v1.png
- mobilenet_v1.py
- mobilenet_v1_test.py

MNIST手写数字识别实践

更多卷积神经网络



LeNet、AlexNet、VGG、GoogLeNet、ResNet

目录

- MNIST数据集
- MNIST手写数字识别实践
  - 逻辑回归方法
  - 人工神经网络方法
  - 卷积神经网络CNN方法
- 调试技巧： TensorFlow可视化
- 小结

MNIST手写数字识别实践

TensorFlow可视化

```
w_h = init_weights([784, 625], "w_h")
w_h2 = init_weights([625, 625], "w_h2")
w_o = init_weights([625, 10], "w_o")

# Add histogram summaries for weights
tf.summary.histogram("w_h_summ", w_h)
tf.summary.histogram("w_h2_summ", w_h2)
tf.summary.histogram("w_o_summ", w_o)

p_keep_input = tf.placeholder("float", name="p_keep_input")
p_keep_hidden = tf.placeholder("float", name="p_keep_hidden")
py_x = model(X, w_h, w_h2, w_o, p_keep_input, p_keep_hidden)

with tf.name_scope("cost"):
    cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=py_x, labels=Y))
    train_op = tf.train.RMSPropOptimizer(0.001, 0.9).minimize(cost)
    # Add scalar summary for cost
    tf.summary.scalar("cost", cost)

with tf.name_scope("accuracy"):
    correct_pred = tf.equal(tf.argmax(Y, 1), tf.argmax(py_x, 1)) # Count correct predicti
    acc_op = tf.reduce_mean(tf.cast(correct_pred, "float")) # Cast boolean to float to av
    # Add scalar summary for accuracy
    tf.summary.scalar("accuracy", acc_op)
```

课程代码： Machine_Learning\03_MNIST\09_tensorboard.ipynb

MNIST手写数字识别实践

TensorFlow可视化

```
with tf.Session() as sess:
    # create a log writer. run 'tensorboard --logdir=./logs/nn_logs'
    writer = tf.summary.FileWriter("./logs/nn_logs", sess.graph)
    merged = tf.summary.merge_all()

    # you need to initialize all variables
    tf.global_variables_initializer().run()

    for i in range(10):
        for start, end in zip(range(0, len(trX), 128), range(128, len(trX)+1, 128)):
            sess.run(train_op, feed_dict={X: trX[start:end], Y: trY[start:end],
                                          p_keep_input: 0.8, p_keep_hidden: 0.5})
        summary, acc = sess.run([merged, acc_op], feed_dict={X: teX, Y: teY,
                                          p_keep_input: 1.0, p_keep_hidden: 1.0})
        writer.add_summary(summary, i)  # Write summary
        print(i, acc)                   # Report the accuracy
```

课程代码： Machine_Learning\03_MNIST\09_tensorboard.ipynb

MNIST手写数字识别实践

TensorFlow可视化

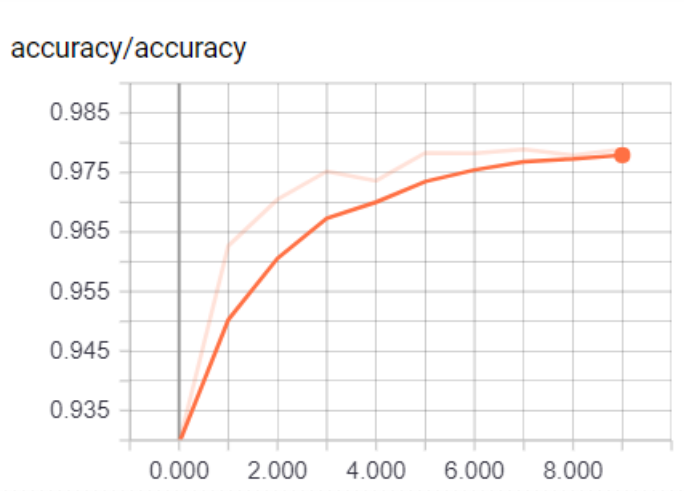
训练好模型之后, cmd 或者Anaconda Prompt 界面中 跳转到代码文件夹，输入
tensorboard --logdir=path/to/log-directory 并运行
注意：path/to/log-directory是你的logs实际路径，如下图

```
\03_MNIST\logs>tensorboard --logdir=.\nn_logs
```

这里是在代码当前路径下输入
tensorboard –logdir= .\nn_logs

这是tensorboard已经启动服务并在端口6006监听

打开浏览器并输入 http://127.0.0.1:6006/



课程代码： Machine_Learning\03_MNIST\09_tensorboard.ipynb

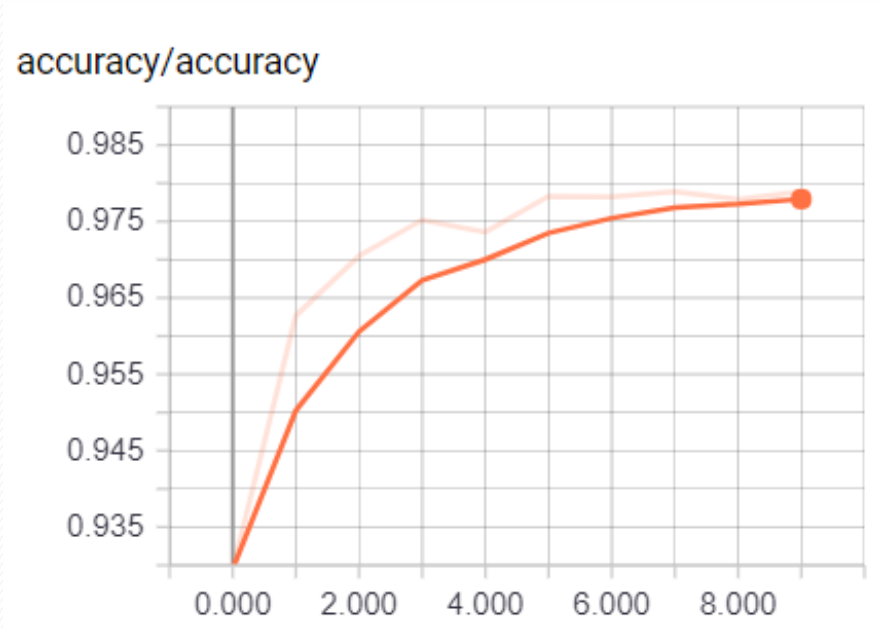
MNIST手写数字识别实践

TensorFlow可视化

打开浏览器并输入 http://127.0.0.1:6006/

```
        writer.add_summary(
        print(i, acc)
```

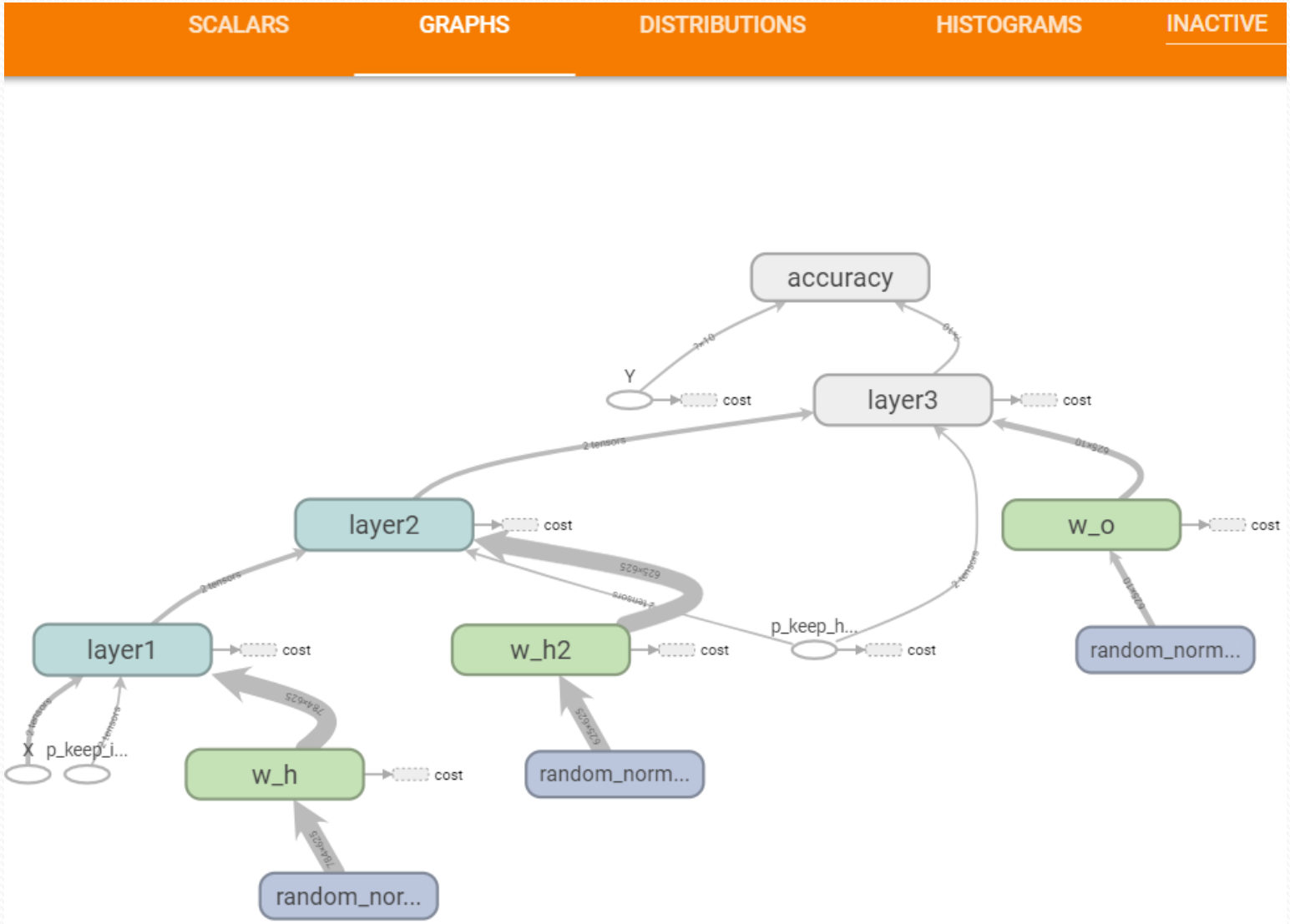
```
0 0.9295
1 0.9627
2 0.9705
3 0.9752
4 0.9736
5 0.9783
6 0.9782
7 0.9789
8 0.9779
9 0.9789
```



课程代码： Machine_Learning\03_MNIST\09_tensorboard.ipynb

MNIST手写数字识别实践

TensorFlow可视化：数据流图



小结

- 从逻辑回归到神经元工作原理
- 感知器到多层神经网络
- 卷积神经网络要素及实践

Thanks