EDU

CSDN学院　IT实战派

Python 深度学习

深度学习入门篇：动手实现神经网络

讲师：彭靖田

本章目录

- 深度学习框架 Keras 简介
- 搭建深度学习开发环境
- 深度学习四件套：数据、模型、损失函数与优化
- 深度学习：”Hello MNIST”
- 深度学习实战：IMDb 数据集介绍
- 深度学习实战：电影评论分类模型

EDU

CSDN学院　IT实战派

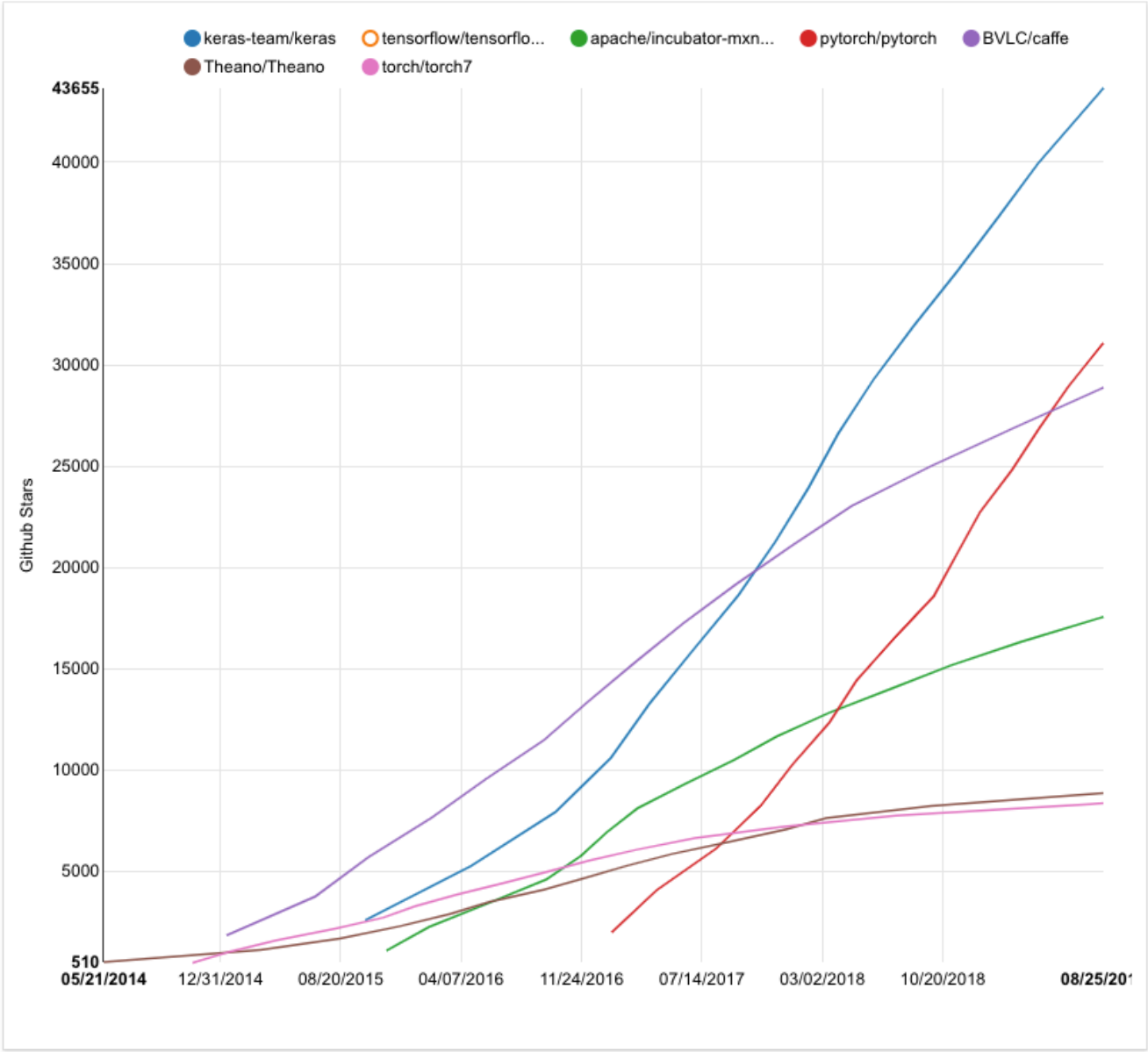
Python 深度学习

深度学习框架 Keras 简介

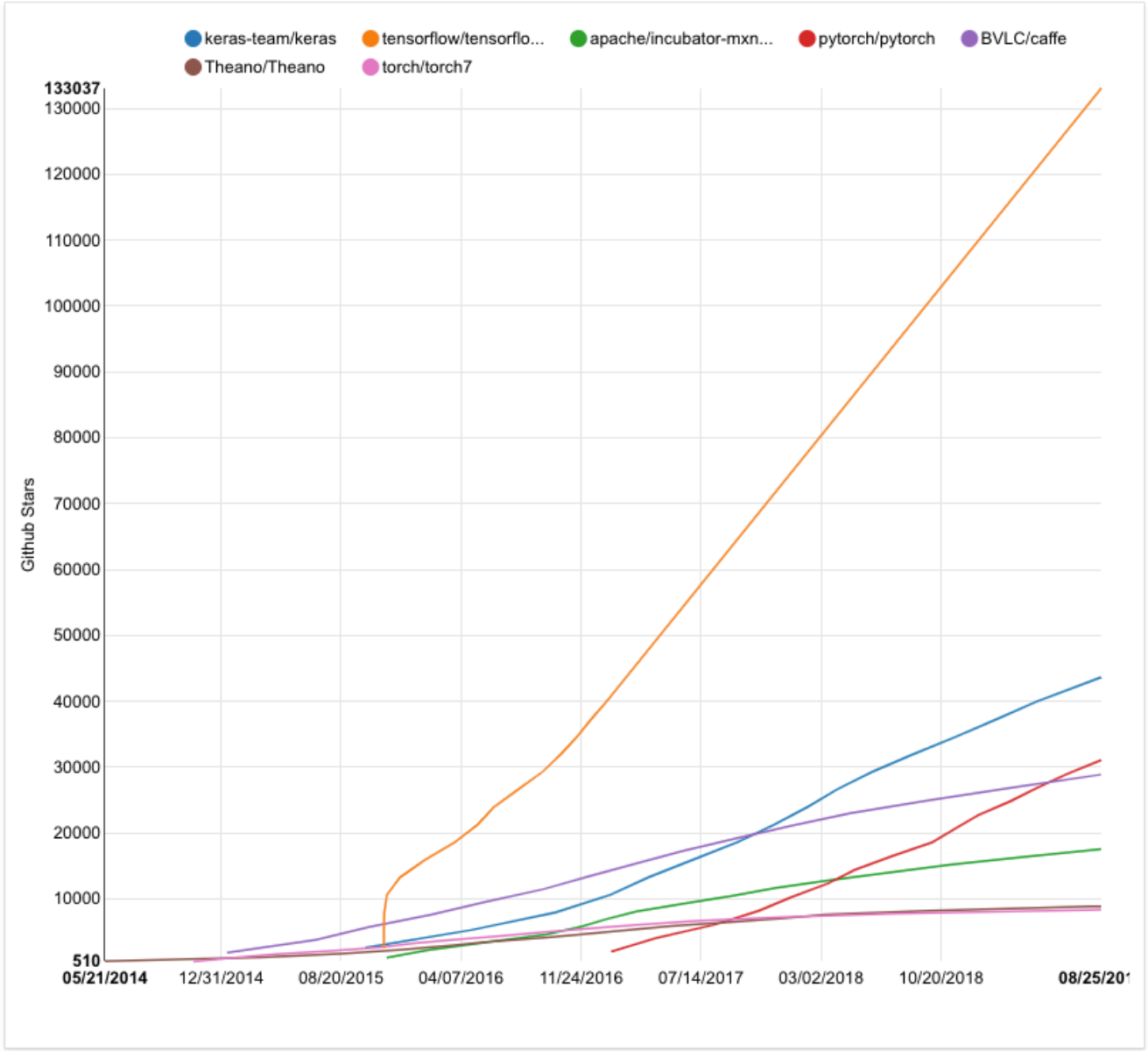
讲师： 彭靖田

Keras 简介

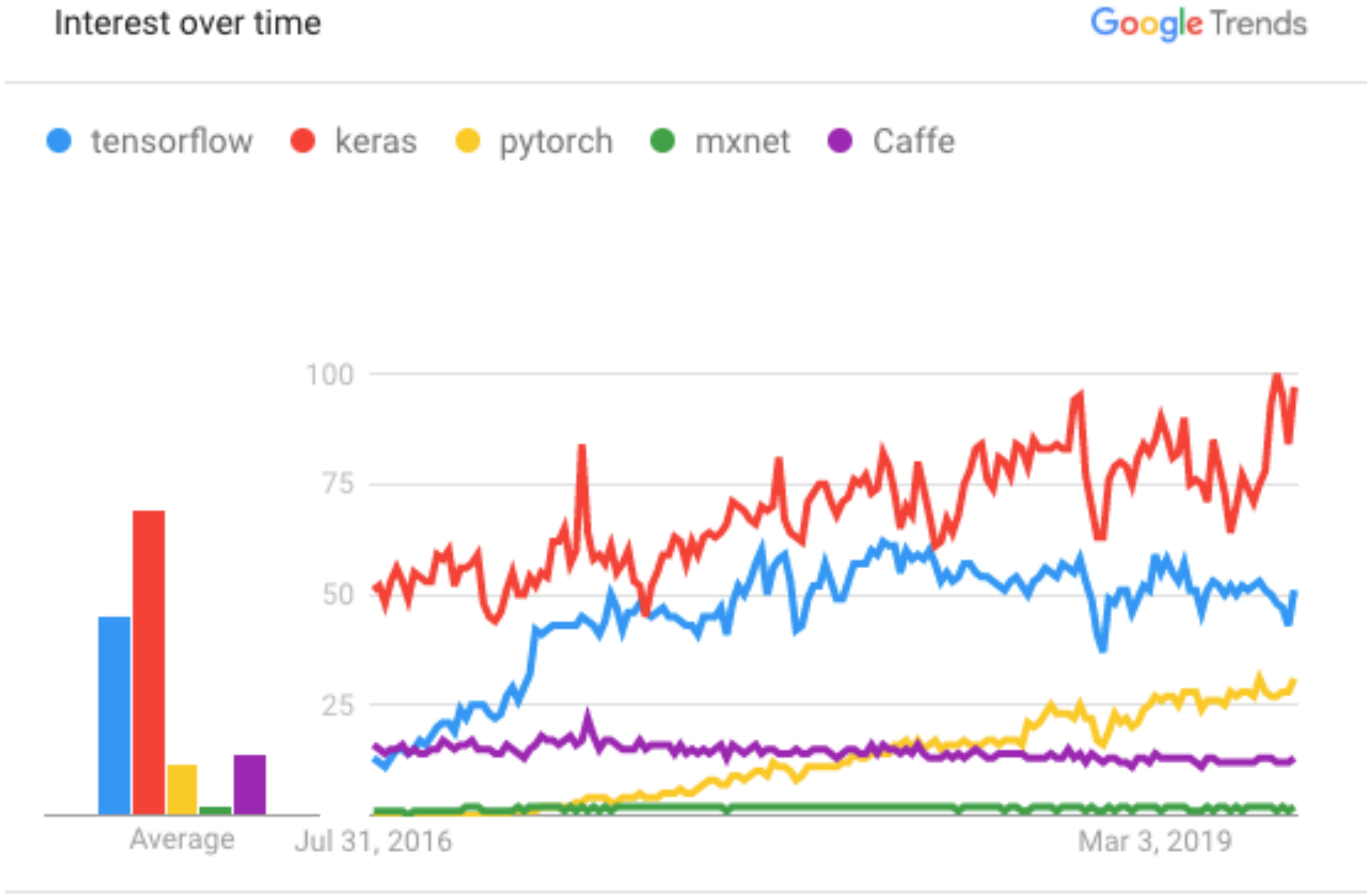
主流深度学习框架热度对比



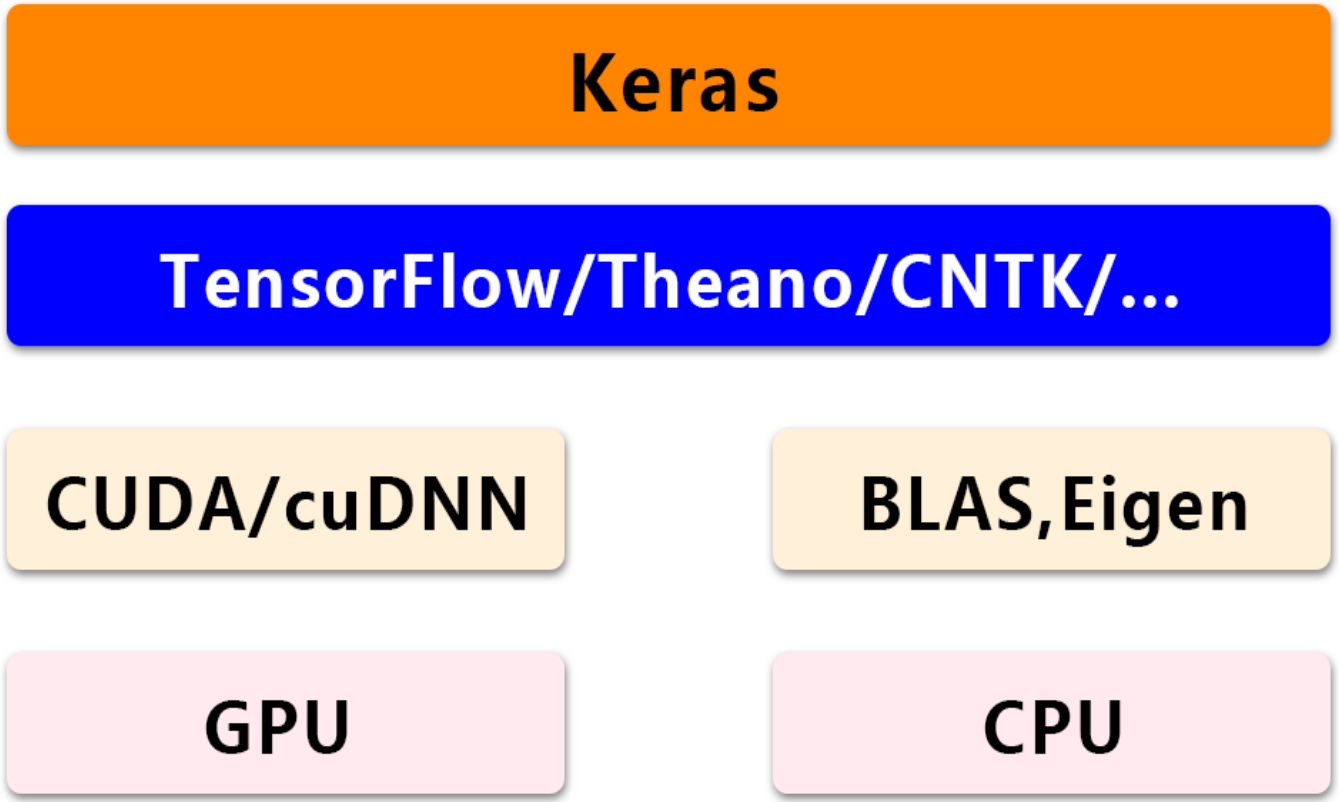
主流深度学习框架热度对比



主流深度学习框架热度对比



Kreas 支持的计算机后端




EDU
CSDN学院 IT实战派

EDU

CSDN学院　IT实战派

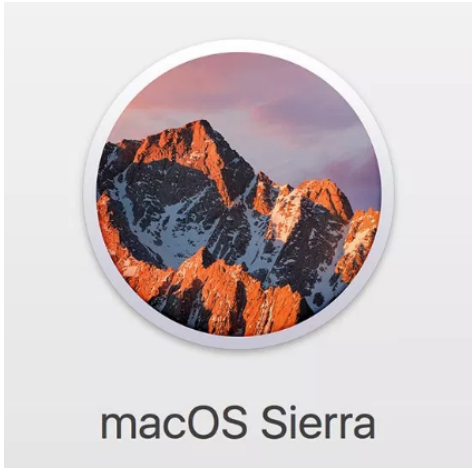
Python 深度学习

搭建深度学习开发环境

讲师： 彭靖田

Kreas 开发环境

- Python 2.7-3.6
- Support CNN & RNN
- Run on CPU & GPU



使用 pip 安装 Keras

```
1. 安装 Keras 2.2.5

$ pip install keras==2.2.5

2. 安装 Keras 模型可视化依赖 GraphViz 和 pydot

$ brew install graphviz
$ pip install pydot

3. 安装交互式开发环境 Jupyter Notebook

$ pip install jupyter
```

在Jupyter中使用Keras开发

```
(base) django:~/projects/DjangoPeng/dl-in-python (master) $ jupyter notebook
[I 14:15:41.059 NotebookApp] JupyterLab extension loaded from /Users/django/anaconda3/lib/python3.7/site-packages/jupyterlab
[I 14:15:41.060 NotebookApp] JupyterLab application directory is /Users/django/anaconda3/share/jupyter/lab
[I 14:15:41.063 NotebookApp] Serving notebooks from local directory: /Users/django/projects/DjangoPeng/dl-in-python
[I 14:15:41.063 NotebookApp] The Jupyter Notebook is running at:
[I 14:15:41.063 NotebookApp] http://localhost:8888/?token=197af9fc8556e496b5a6b26e679f02623d1218b02808d943
[I 14:15:41.064 NotebookApp]  or http://127.0.0.1:8888/?token=197af9fc8556e496b5a6b26e679f02623d1218b02808d943
[I 14:15:41.064 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 14:15:41.078 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///Users/django/Library/Jupyter/runtime/nbserver-69349-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=197af9fc8556e496b5a6b26e679f02623d1218b02808d943
     or http://127.0.0.1:8888/?token=197af9fc8556e496b5a6b26e679f02623d1218b02808d943
```

Try it


EDU
CSDN学院 IT实战派

EDU

CSDN学院　IT实战派

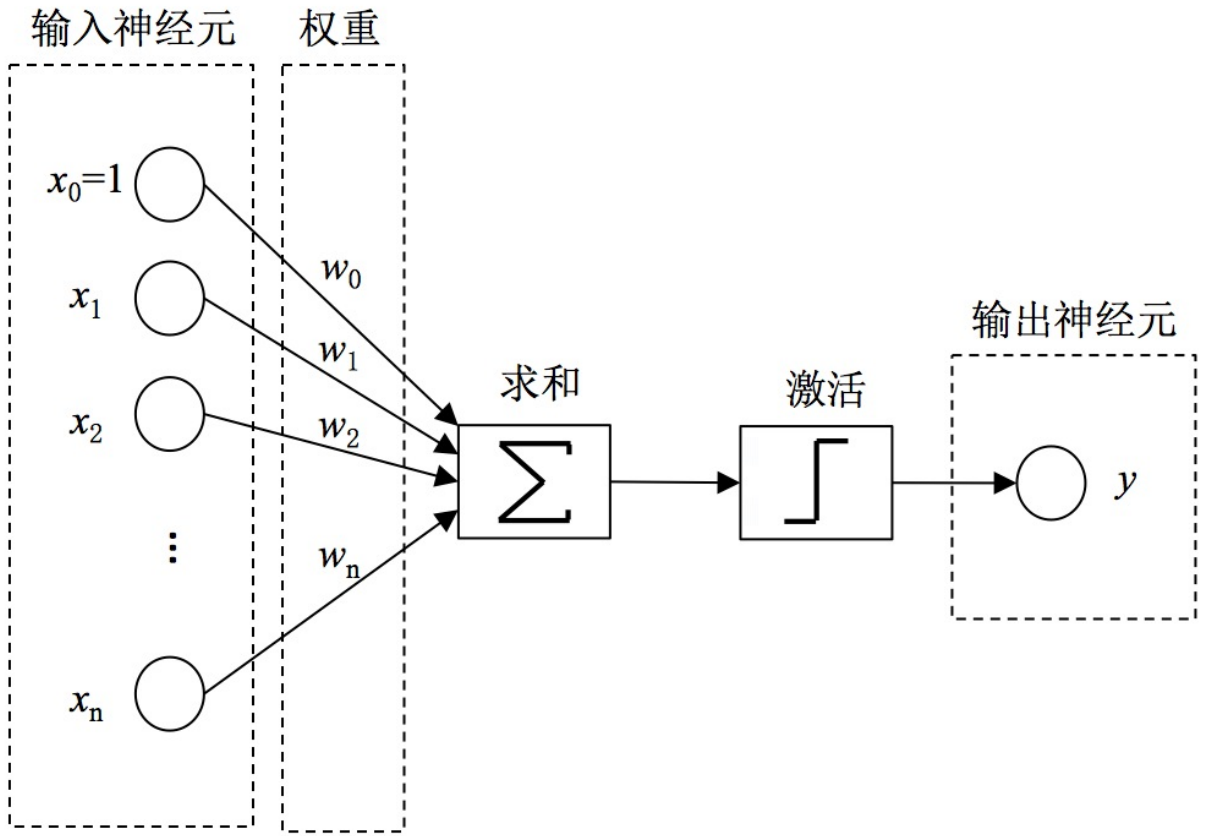
Python 深度学习

深度学习四件套：数据、模型、损失函数与优化器

讲师： 彭靖田

感知机模型

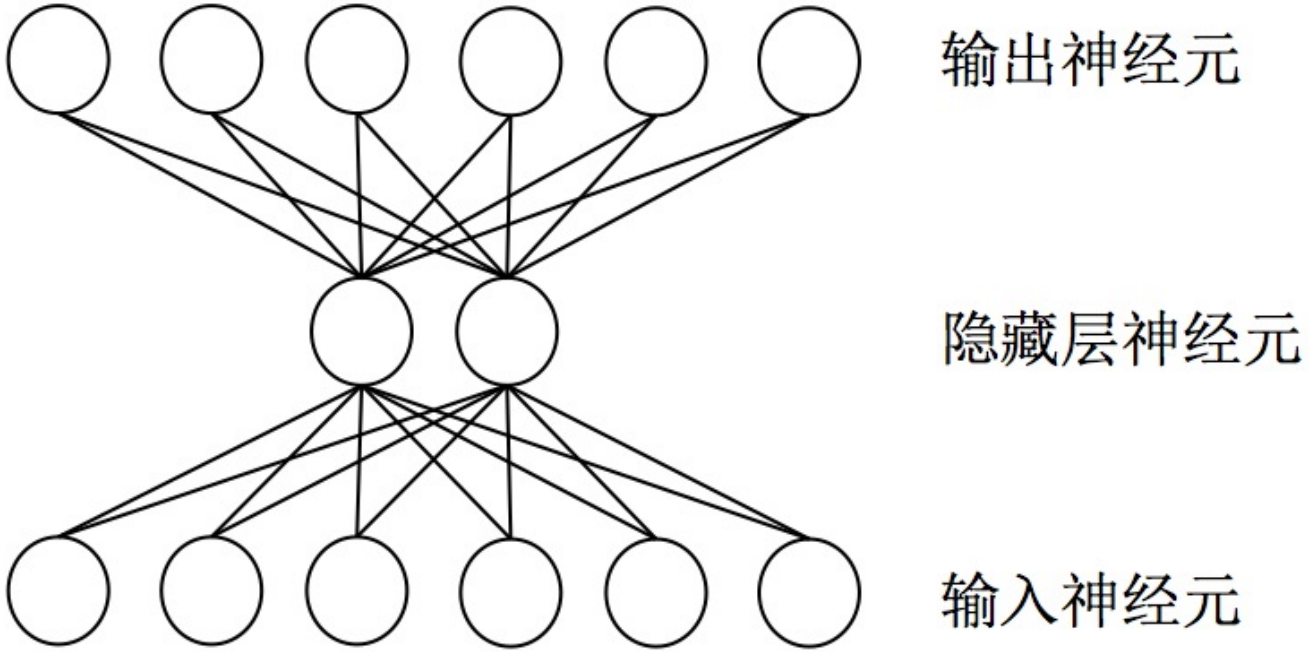
1957年，受 Warren McCulloch 和 Walter Pitts 在神经元建模方面工作的启发，心理学家 Frank Rosenblatt 参考大脑中神经元信息传递信号的工作机制，发明了神经感知机模型 Perceptron 。



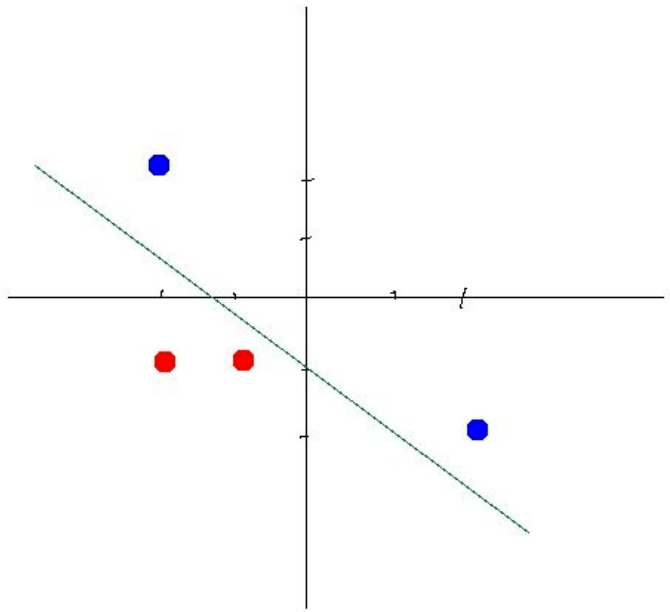
二分类模型

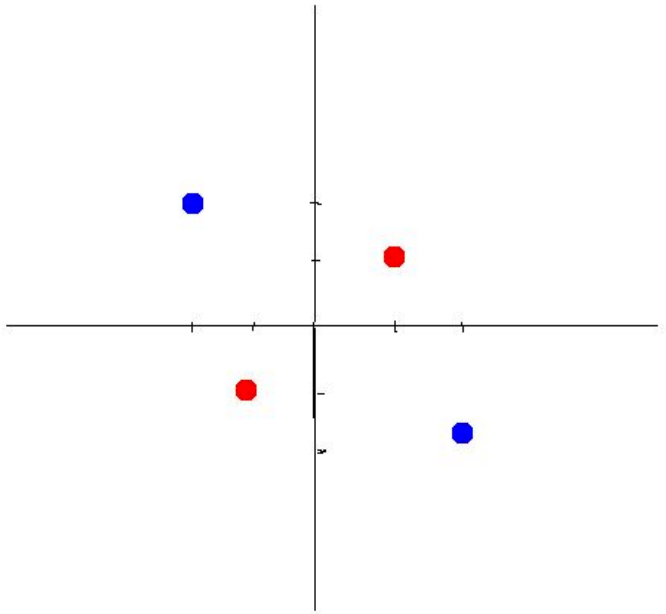
神经网络

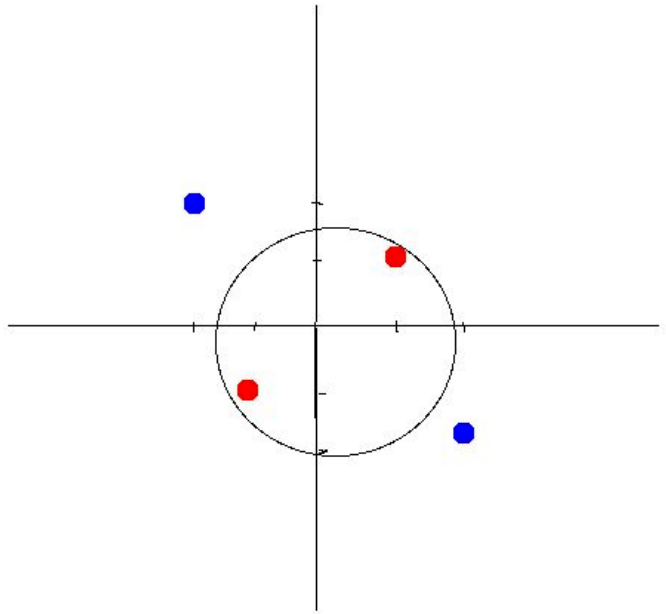
在机器学习和认知科学领域，人工神经网络（ANN），简称神经网络（NN）是一种模仿生物神经网络（动物的中枢神经系统，特别是大脑）的结构和功能的数学模型或计算模型，用于对函数进行估计或近似。神经网络是多层神经元的连接，上一层神经元的输出，作为下一层神经元的输入。



线性不可分





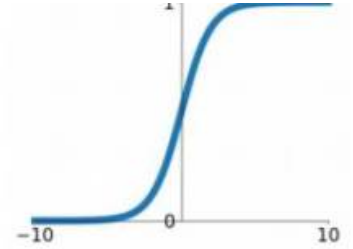


激活函数（Activation Function）

为了实现神经网络的非线性建模能力，解决一些线性不可分的问题，我们通常使用激活函数来引入非线性因素。激活函数都采用非线性函数，常用的有Sigmoid、tanh、ReLU等。

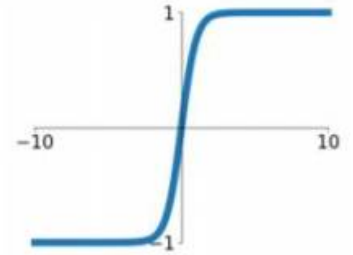
Sigmoid

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



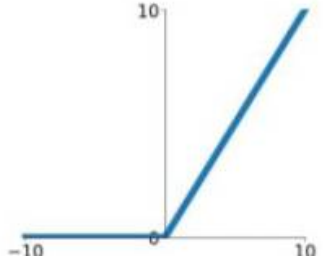
tanh

$$\tanh(x)$$



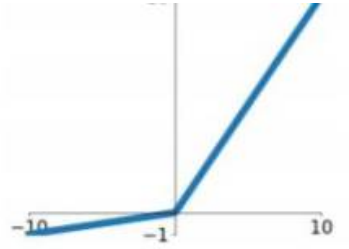
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

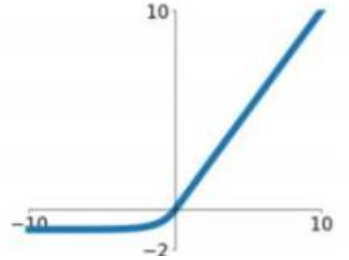


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

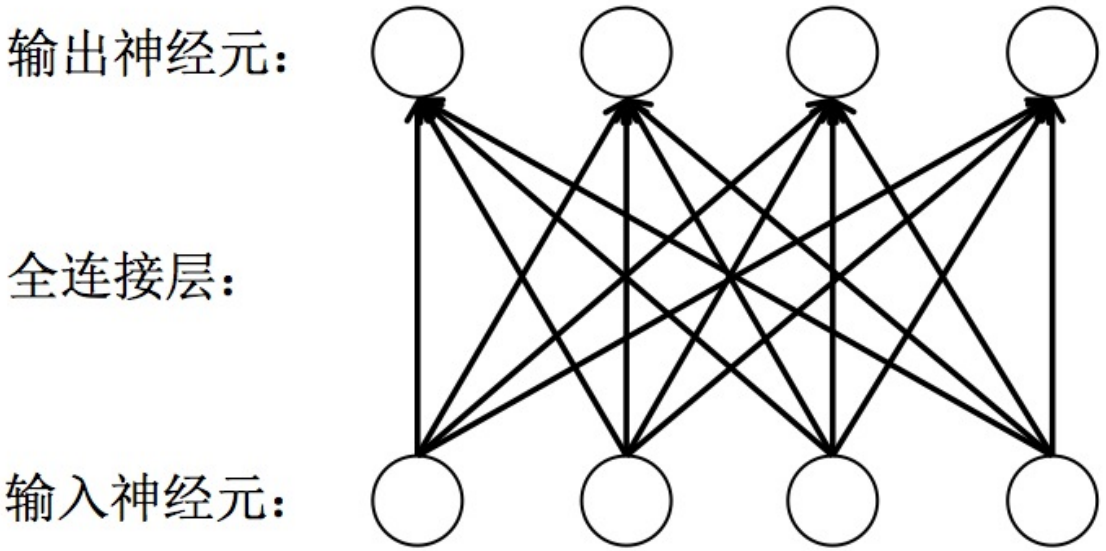
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

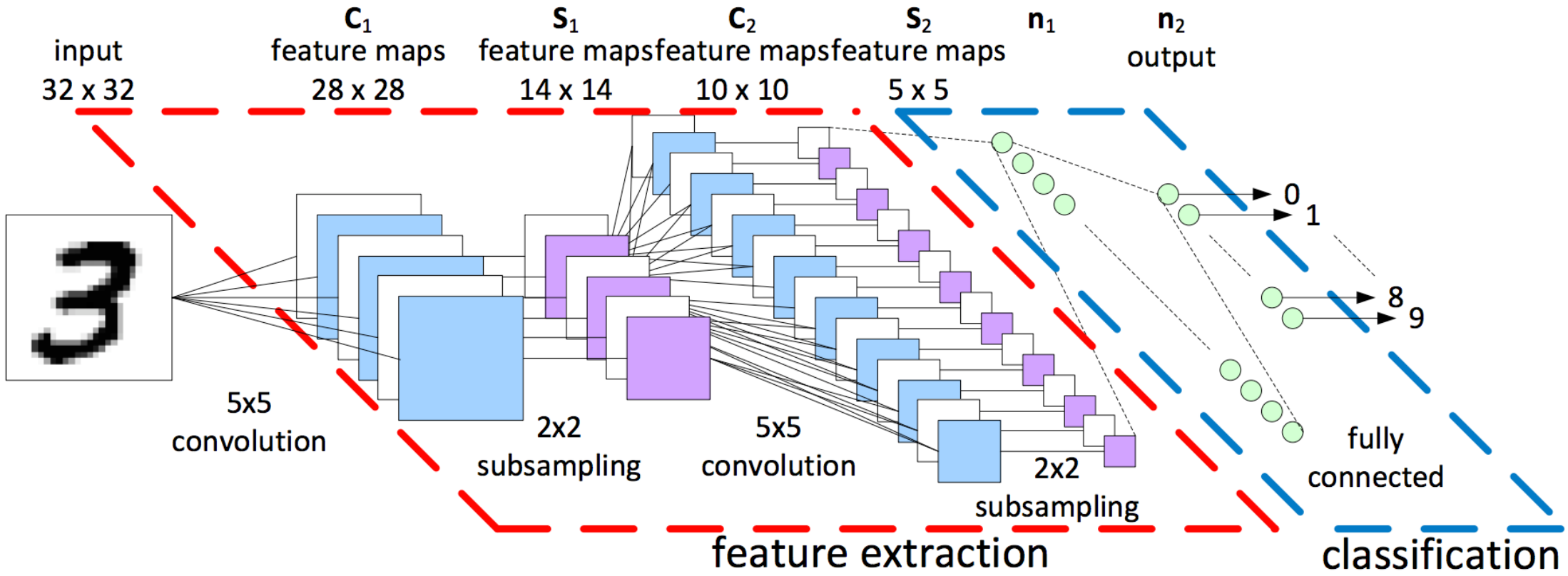


全连接层（ fully connected layers，FC ）

全连接层是一种对输入数据直接做线性变换的线性计算层。它是神经网络中最常用的一种层，用于学习输出数据和输入数据之间的变换关系。全连接层可作为特征提取层使用，在学习特征的同时实现特征融合；也可作为最终的分类层使用，其输出神经元的值代表了每个输出类别的概率。



模型：层构成的神经网络


input
32 x 32
C1
feature maps
28 x 28
S1
feature maps
14 x 14
C2
feature maps
10 x 10
S2
feature maps
5 x 5
n1
n2
output
0
1
8
9
5x5
convolution
2x2
subsampling
5x5
convolution
2x2
subsampling
fully
connected
feature extraction
classification


EDU
CSDN学院 IT实战派

EDU

CSDN学院　IT实战派

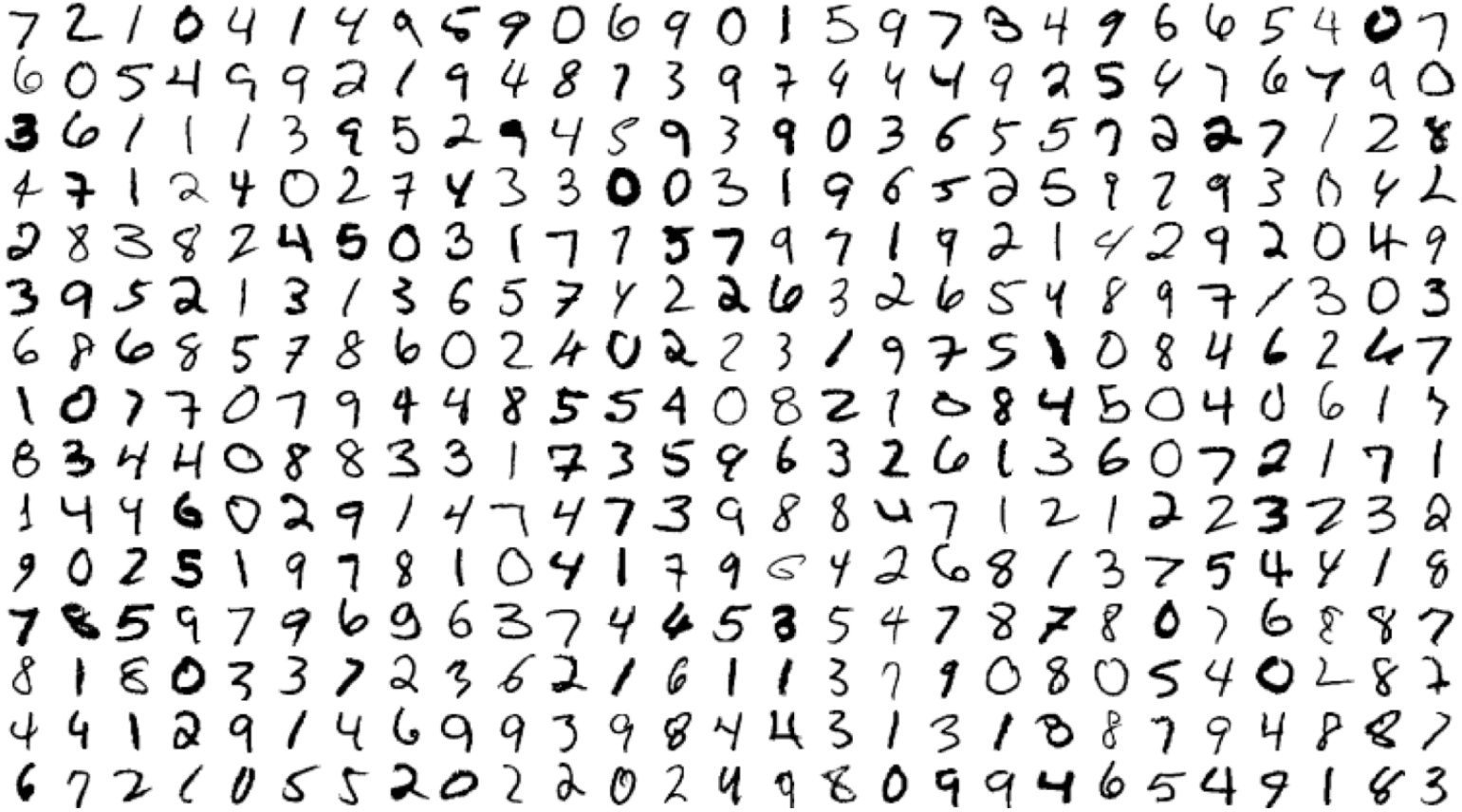
Python 深度学习

深度学习：”Hello MNIST”

讲师： 彭靖田

MNIST 数据集介绍

MNIST 是一套手写体数字的图像数据集，包含 60,000 个训练样例和 10,000 个测试样例，由纽约大学的 Yann LeCun 等人维护。



获取 MNIST 数据集

THE MNIST DATABASE

of handwritten digits

Yann LeCun, Courant Institute, NYU
Corinna Cortes, Google Labs, New York
Christopher J.C. Burges, Microsoft Research, Redmond

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.

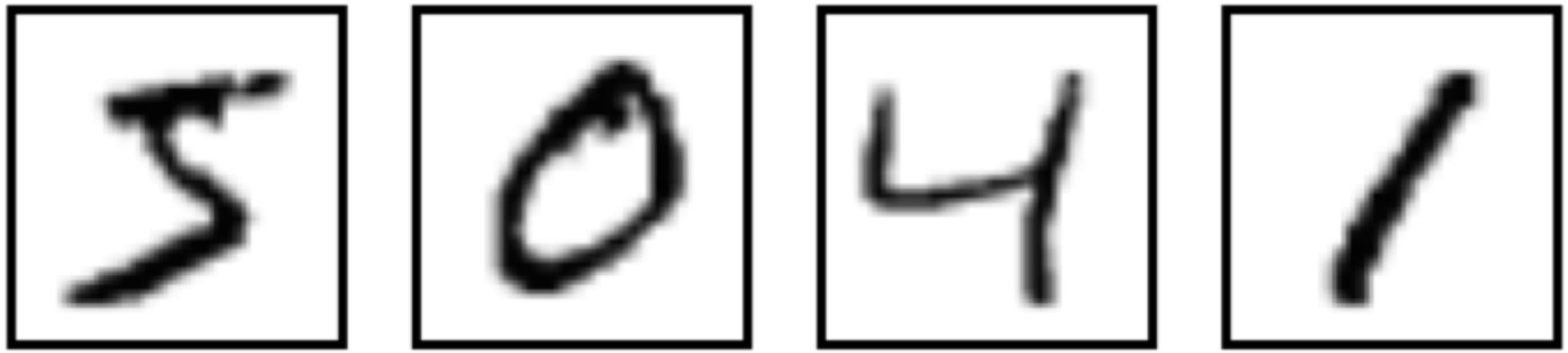
It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

Four files are available on this site:

```
train-images-idx3-ubyte.gz:  training set images (9912422 bytes)
train-labels-idx1-ubyte.gz:  training set labels (28881 bytes)
t10k-images-idx3-ubyte.gz:   test set images (1648877 bytes)
t10k-labels-idx1-ubyte.gz:   test set labels (4542 bytes)
```

MNIST 手写字体介绍

MNIST 图像数据集使用形如 [28，28] 的二阶数组来表示每个手写体数字，数组中的每个元素对应一个像素点，即每张图像大小固定为 28x28 像素。

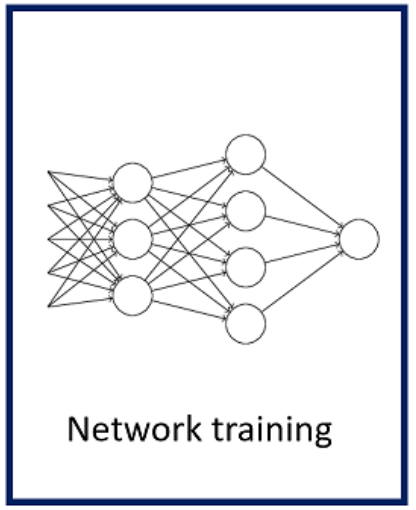


“Hello MNIST”

（输入）数据

Data & Labels

模型



0
1
2
3
4
5
6
7
8
9

使用 Keras 加载 MNIST 数据集

tf.kera.datasets.mnist.load_data(path= ‘mnist.npz’)

Arguments:

- **path:**本地缓存 MNIST 数据集(mnist.npz)的相对路径（ ~/.keras/datasets ）

Returns :

Tuple of **Numpy** arrays: `(x_train, y_train), (x_test, y_test)`.

```
In [5]: from keras.datasets import mnist

        (x_train, y_train), (x_test, y_test) = mnist.load_data('mnist/mnist.npz')
```

```
In [9]: print(x_train.shape, y_train.shape)
        print(x_test.shape, y_test.shape)
```

```
(60000, 28, 28) (60000,)
(10000, 28, 28) (10000,)
```

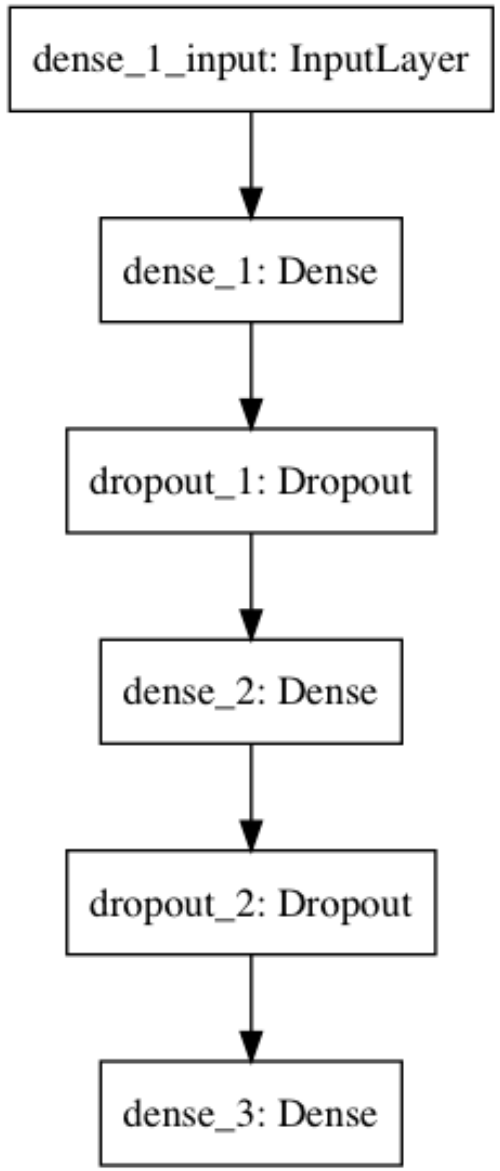
使用 Keras 定义 MNIST 模型

```
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 512)               401920
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                5130
=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
```

```
keras.utils.plot_model(model, to_file='mnist.png')
```



使用 Keras 训练 MNIST 模型

```
model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(),
              metrics=['accuracy'])

history = model.fit(x_train, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/20
60000/60000 [==============================] - 10s 168us/step - loss: 0.2
482 - acc: 0.9251 - val_loss: 0.1091 - val_acc: 0.9648
Epoch 2/20
60000/60000 [==============================] - 9s 149us/step - loss: 0.10
35 - acc: 0.9685 - val_loss: 0.0827 - val_acc: 0.9763
Epoch 3/20
60000/60000 [==============================] - 9s 153us/step - loss: 0.07
56 - acc: 0.9772 - val_loss: 0.1068 - val_acc: 0.9698
Epoch 4/20
60000/60000 [==============================] - 8s 137us/step - loss: 0.06
00 - acc: 0.9819 - val_loss: 0.0966 - val_acc: 0.9730

...

Epoch 19/20
60000/60000 [==============================] - 11s 180us/step - loss: 0.0
178 - acc: 0.9953 - val_loss: 0.1168 - val_acc: 0.9837
Epoch 20/20
60000/60000 [==============================] - 9s 150us/step - loss: 0.01
92 - acc: 0.9946 - val_loss: 0.1064 - val_acc: 0.9830
Test loss: 0.10641669383070261
Test accuracy: 0.983
```

Try it


EDU
CSDN学院 IT实战派

EDU

CSDN学院　IT实战派

Python 深度学习

深度学习实战：IMDb 数据集介绍

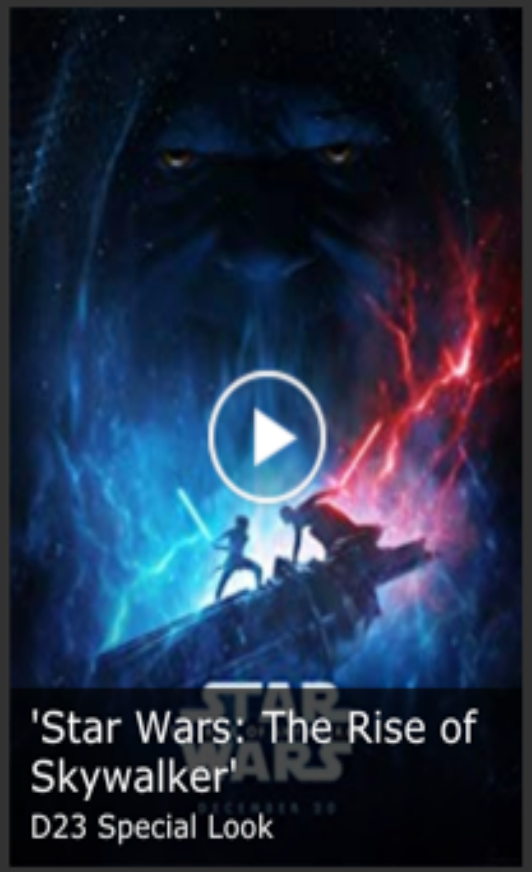
讲师： 彭靖田

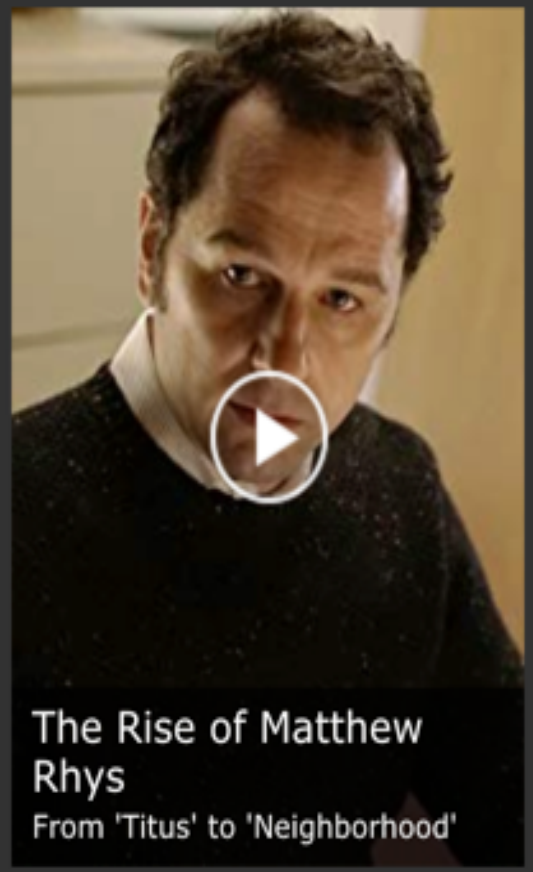
二分类是什么？

模型

狗

猫





Browse trailers »

Opening This Week

- Don't Let Go
- Saaho
- Before You Know It
- Bennett's War
- Tod@s Caen
- Ne Zha
- Killerman

See more opening this week »

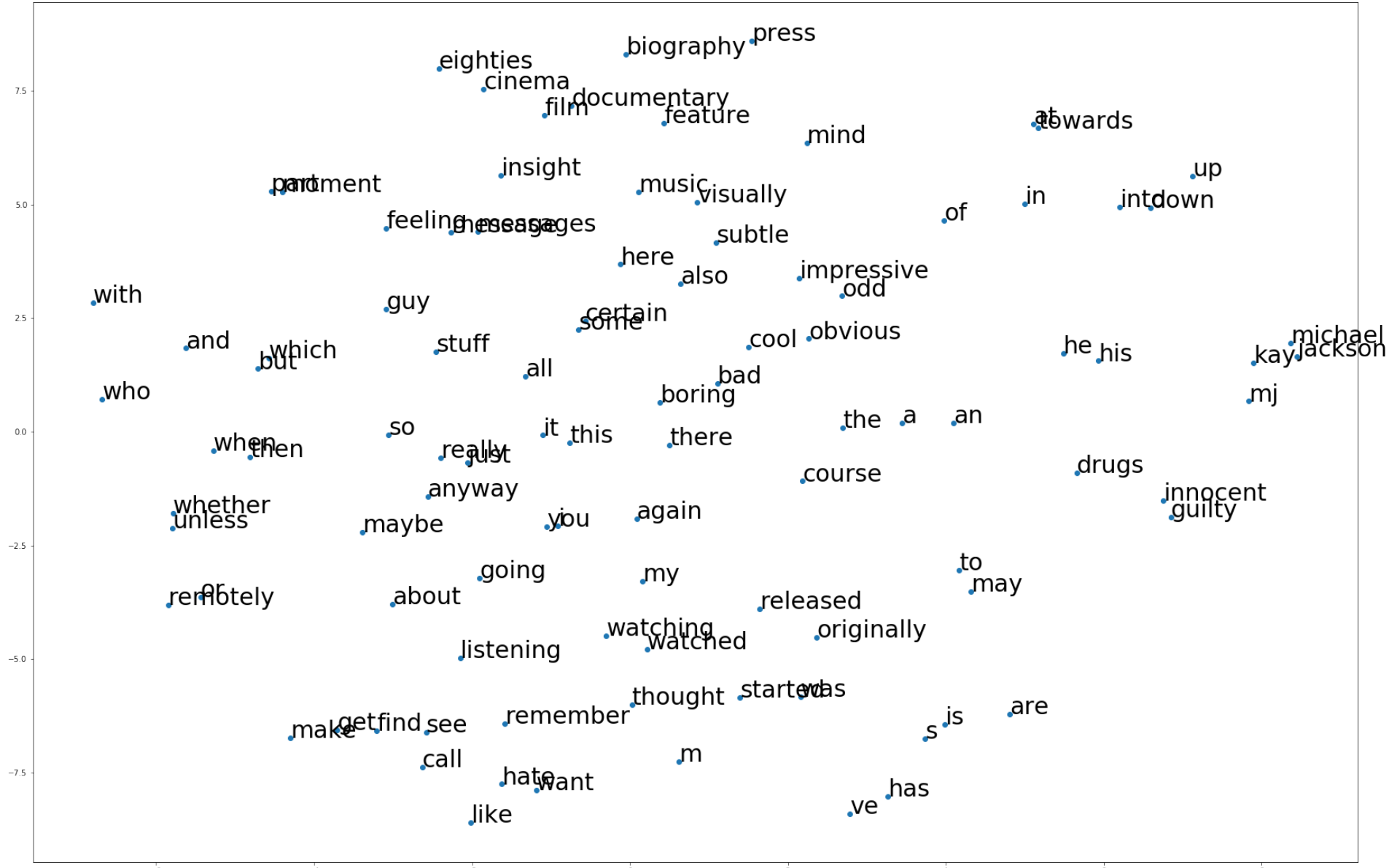
'Rise of Skywalker' Cast Brace for a 'Crazy Adventure'

IMDb Sentiment 数据集

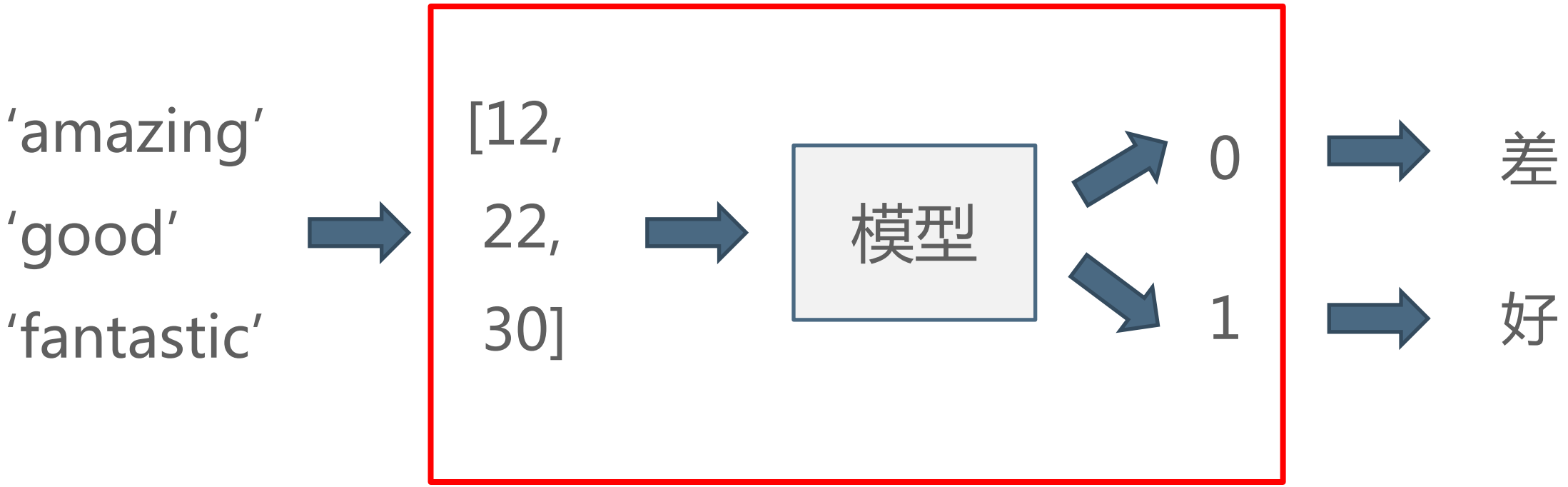
Dataset of 25,000 movies reviews from IMDB, labeled by sentiment (positive/negative). Reviews have been preprocessed, and each review is encoded as a sequence of word indexes (integers). For convenience, words are indexed by overall frequency in the dataset, so that for instance the integer "3" encodes the 3rd most frequent word in the data. This allows for quick filtering operations such as: "only consider the top 10,000 most common words, but eliminate the top 20 most common words".

As a convention, "0" does not stand for a specific word, but instead is used to encode any unknown word.

IMDb Sentiment 数据集可视化



电影评论分类模型



使用 Keras 加载 IMDb 数据集

```
import os
import keras

import numpy as np

from keras import models, layers
from keras import optimizers
from keras.datasets import imdb
```

```
Using TensorFlow backend.
```

```
(train_data, train_labels), (test_data, test_labels) = imdb.load_data(path='imdb.npz', num_words=10000)
print(train_data.shape)
print(train_labels.shape)
```

```
(25000,)
(25000,)
```

向量化数据

```
def vectorize(seqs, dim=10000):
    ret = np.zeros((len(seqs), dim))
    for i, seq in enumerate(seqs):
        ret[i, seq] = 1
    return ret

x_train = vectorize(train_data)
x_test = vectorize(test_data)

y_train = np.asarray(train_labels).astype('float32')
y_test = np.asarray(test_labels).astype('float32')
```

```
print(x_train.shape)
print(y_train.shape)
print(x_train[0])
print(y_train[0])
```

```
(25000, 10000)
(25000,)
[0. 1. 1. ... 0. 0. 0.]
1.0
```


EDU
CSDN学院 IT实战派

EDU

CSDN学院　IT实战派

Python 深度学习

深度学习实战：电影评论分类模型

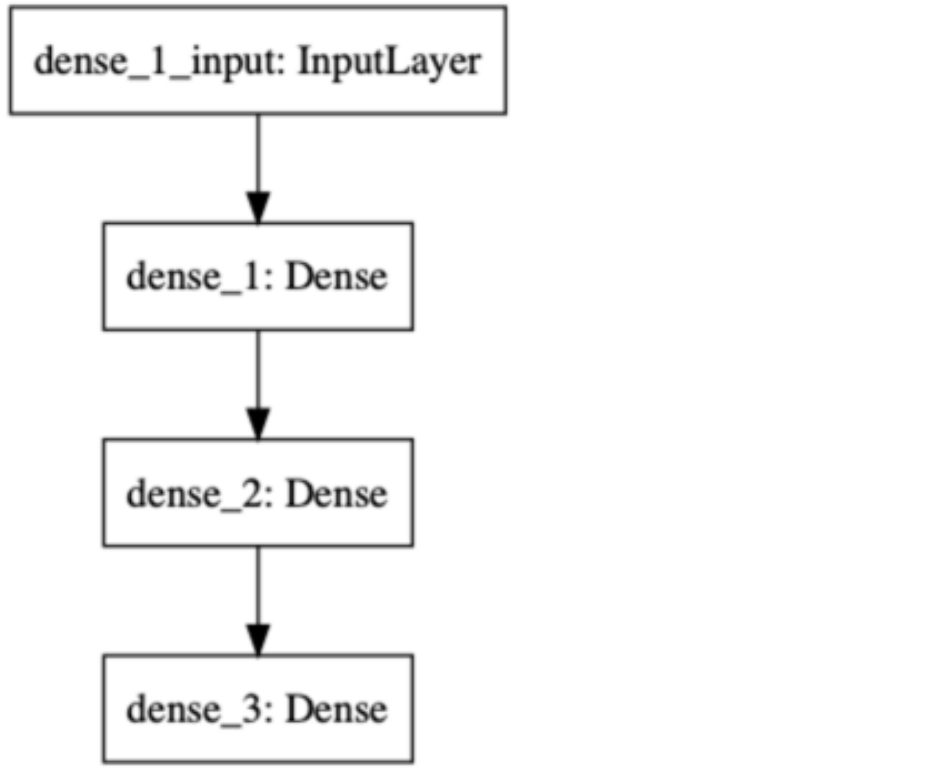
讲师： 彭靖田

使用 Keras 定义 电影评论分类模型

```
model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.summary()
```

```
WARNING:tensorflow:From /Users/django/anaconda3/lib/python3.7/site-packag
es/tensorflow/python/framework/op_def_library.py:263: colocate_with (from
tensorflow.python.framework.ops) is deprecated and will be removed in a f
uture version.
Instructions for updating:
Colocations handled automatically by placer.
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 16)                160016
_________________________________________________________________
dense_2 (Dense)              (None, 16)                272
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 17
=================================================================
Total params: 160,305
Trainable params: 160,305
Non-trainable params: 0
```

```
keras.utils.plot_model(model, to_file='imdb.png')
```




EDU
CSDN学院 IT实战派