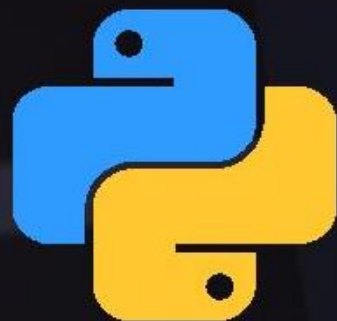


Python程序语言入门与应用



python

Life is short, use Python
人生苦短，我用Python



课程回顾



第13章 Python办公自动化

13.1 Office办公

 Excel、Word、PowerPoint

13.2 Word办公自动化

13.3 Excel办公自动化

13.4 PPT办公自动化



上周练习题



13.1 请假条word文档模板化生成

要包含如下信息：申请表标题、姓名、学号、院系、年级、请假理由、请假时间、请假天数、辅导员意见和签字。

新乡医学院请假条

填表日期：{{year}}年{{month}}月{{day}}日

姓名	{{name}}	学号	{{sno}}	院系	{{college}}
年级	{{grade}}		联系方式	{{phone}}	
请假时间	{{syear}}年{{smmonth}}月{{sday}}日至{{eyear}}年{{emmonth}}月{{eday}}日		请假天数	{{dno}}天	
请假事由	{{reason}}		本人签字：_____ 年 月 日		
辅导员意见	{{reply}}		辅导员签字：_____ 年 月 日		

模板

新乡医学院请假条

填表日期：2019年12月13日

姓名	郭德纲	学号	2017001	院系	生命科学技术学院
年级	2017级		联系方式	18903730000	
请假时间	2019年12月20日至2019年12月日			请假天数	5天
请假事由	世界那么大，我想去看看。 本人签字：_____ 年 月 日				
辅导员意见	同意 辅导员签字：_____ 年 月 日				

生成示例



上周练习题



13.1 请假条word文档模板化生成

要包含如下信息：申请表标题、姓名、学号、院系、年级、请假理由、请假时间、请假天数、辅导员意见和签字。

```
from docxtpl import DocxTemplate
from datetime import datetime

tpl = DocxTemplate('请假条模板.docx')
context = {
    'name': '郭德纲', 'sno': '2017001', 'college': '生命科学技术学院',
    'grade': '2017级', 'phone': '18903730000', 'syear': 2019, 'smmonth': 12,
    'sday': 16, 'eyear': 2019, 'emmonth': 12, 'sday': 20, 'dno': 5,
    'reason': '世界那么大,我想去看看。', 'reply': '同意',
    'year': datetime.now().year, 'month': datetime.now().month,
    'day': datetime.now().day,
}
tpl.render(context)
tpl.save('{}的请假条.docx'.format(context['name']))
```

Python代码示例



上周练习题



13.2 Excel图表绘制

使用openpyxl库自由选择数据绘制任意形式的图表。

```
from openpyxl import Workbook
from openpyxl.chart import LineChart, Reference

wb = Workbook()
ws = wb.active
rows = [
    ['品牌', '华为', 'OPPO', 'vivo', '小米', '苹果'],
    ['Q1', 24.2, 18.9, 16.3, 15.1, 11.3],
    ['Q2', 27.2, 20.2, 19.0, 14.2, 8.0],
    ['Q3', 24.6, 20.4, 21.7, 14.0, 7.4],
    ['Q4', 29.0, 19.6, 18.8, 10.0, 11.5],
]
chart_loc = ['A8', 'H8', 'A21', 'H21']

for row in rows:
    ws.append(row)
```

```
for stl in range(1, 5):
    c1 = LineChart()
    c1.title = "2018年手机市占率 (中国) "
    c1.style = stl
    c1.y_axis.title = '市占率 (%) '
    c1.x_axis.title = '季度'
    data = Reference(ws, min_col=2, min_row=1,
max_col=6, max_row=5)
    c1.add_data(data, titles_from_data=True)
    c1.width = 12
    c1.height = 6
    ws.add_chart(c1, chart_loc[stl-1])
wb.save("13.2charts.xlsx")
```

Python代码示例

运行结果

5

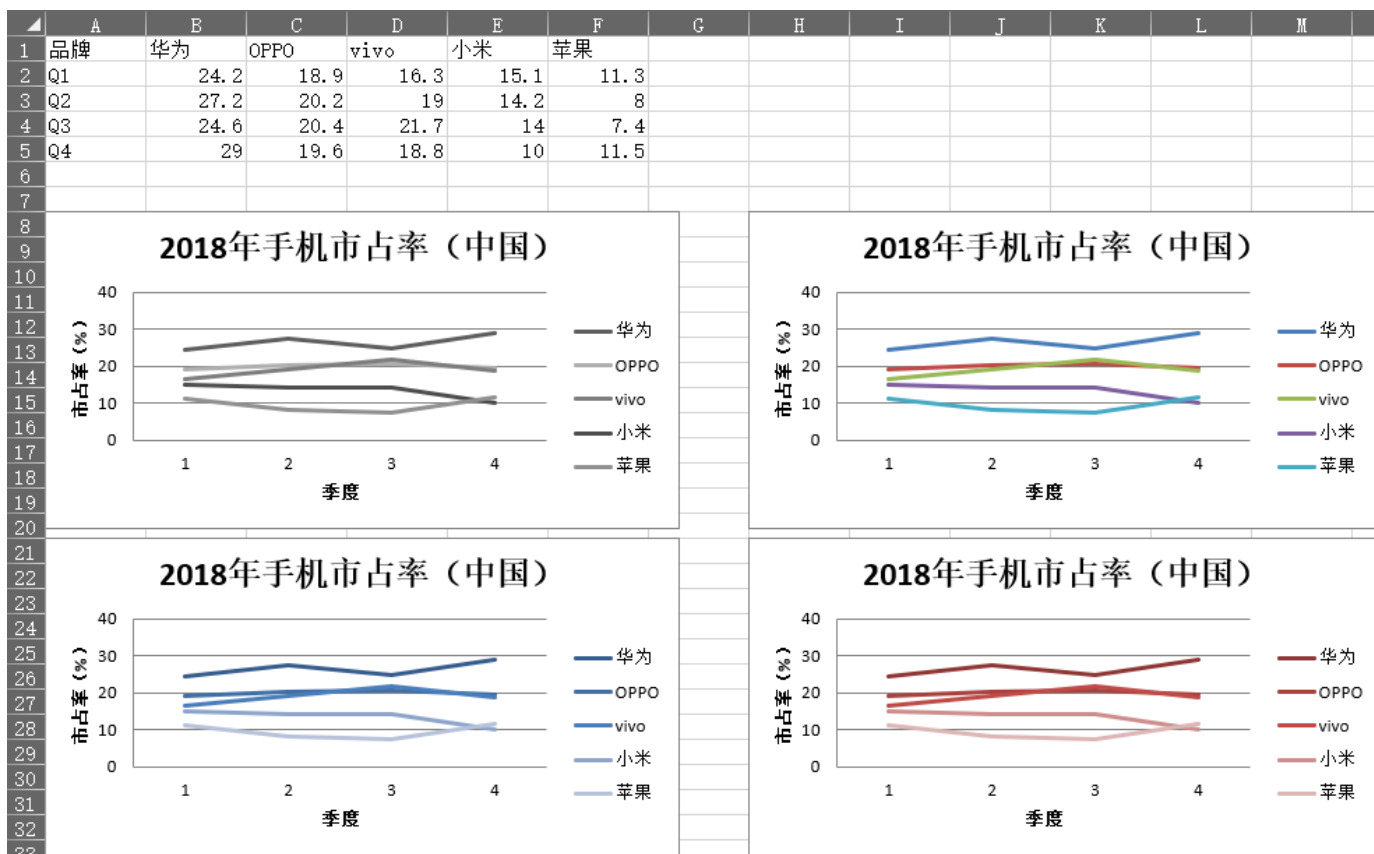


上周练习题



13.2 Excel图表绘制

使用openpyxl库自由选择数据绘制任意形式的图表。





新乡医学院

Python程序语言入门与应用

应用Python

第十四章 Python机器学习



郭长江

changjiangguo@xxmu.edu.cn

生命科学技术学院

新乡医学院





本课概要



第14章 Python机器学习

14.1 机器学习简介

- 机器学习、深度学习与人工智能
- 基本原理

14.2 机器学习经典算法与框架

- 算法简介及Python实现
- 框架简介及比较

14.3 Python机器学习实例

- MNIST
- Iris Flower



14.1 机器学习简介

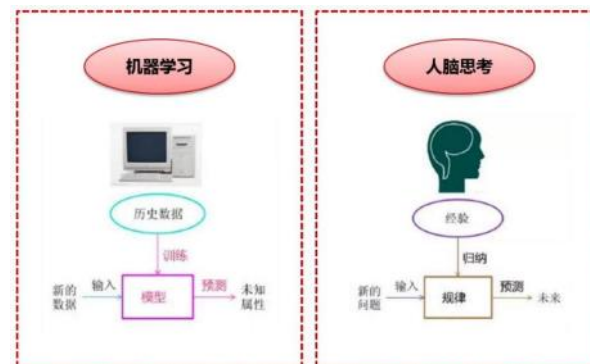
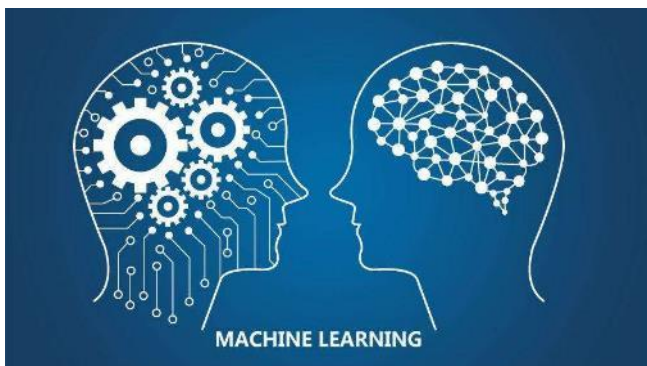


14.1 机器学习简介



机器学习Machine Learning

- 一门多领域交叉学科，涉及概率论、统计学、逼近论、凸分析、算法复杂度理论等多门学科。
- 专门研究计算机怎样模拟或实现人类的学习行为，以获取新的知识或技能，并不断改善自身的性能。
- 是人工智能的核心，使计算机具有智能的根本途径。





14.1 机器学习简介



人工智能 AI

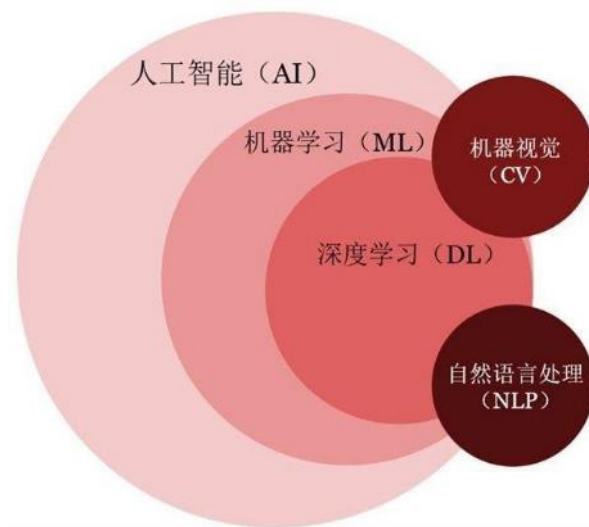
- 描述一种机器与周围世界交互的各种方式，将人类的智能与机器结合起来。

机器学习 ML

- 人工智能的一种途径或子集，赋予计算机系统“学习”的能力

深度学习 DL

- 机器学习的一个子集，是通过神经网络实现机器学习的技术。





14.1 机器学习简介



❖ 机器学习发展大事件

- ❖ 1783年，贝叶斯提出贝叶斯定理，发明从经验中学习的数学方法，这是机器学习的基本思想。
- ❖ 1950年，艾伦·图灵发明了图灵测试，只有当机器通过这项测试才能被认为是“智能的”。
- ❖ 1952年，亚瑟·塞缪尔针对跳棋游戏开发了第一个真正的机器学习程序。
- ❖ 1963年，唐纳德·米基开发了基于强化学习的tic-tac-toe项目。



14.1 机器学习简介



❖ 机器学习发展大事件

- ❖ 1997年，IBM的国际象棋电脑Deep Blue击败世界冠军Garry Kasparov。
- ❖ 2016年，AlphaGo击败了世界围棋冠军李世石。
- ❖ 2017年，AlphaGo Master (AlphaGo Zero) 在网络上与人类棋手的对阵中取得了60盘不败的战绩；
- ❖ 2017年底，DeepMind 发布AlphaZero，它只使用了基本规则，没有人的经验，从零开始训练，横扫了棋类游戏AI。





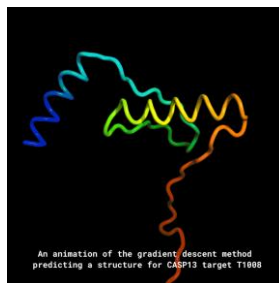
14.1 机器学习简介



机器学习发展大事件

- 2018年，DeepMind开发的AlphaFold在第13届CASP中击败了所有对手，成功预测生命基本分子——蛋白质的三维结构。
- 2018年，DeepMind开发的AlphaStar在实时策略游戏《星际争霸II》中达到登峰造极的大师级别。

你可能已经听说了
AlphaGo家族又添新狗了

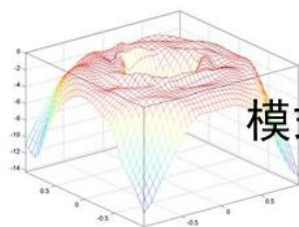




14.1 机器学习简介



机器学习应用



模式识别

计算机视觉



数据挖掘



机器学习

语音识别



统计学习



自然语言处理  Google Translate
Break through language barriers.



14.1 机器学习简介



机器学习应用领域

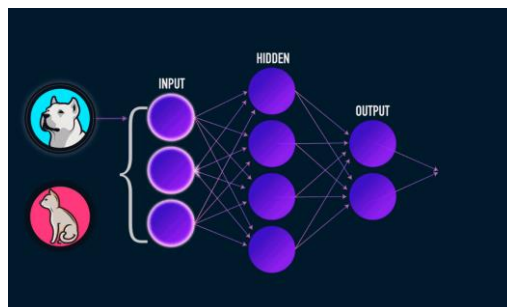
最主要的应用领域有

专家系统、认知模拟、规划和问题求解、数据挖掘、网络信息服务、图像识别、故障诊断、自然语言理解、机器人和博弈等领域

大部分的应用研究领域基本上集中于两个范畴

分类

问题求解





14.1 机器学习简介



❖ 机器学习方法分类

❖ 监督学习 (Supervised Learning)

❖ 通过已有的一部分输入数据与输出数据之间的关系，生成一个函数，将输入映射到合适的输出，例如分类、回归

❖ 非监督学习 (Unsupervised Learning)

❖ 直接对输入数据进行建模，例如聚类

❖ 半监督学习 (Semi-supervised Learning)

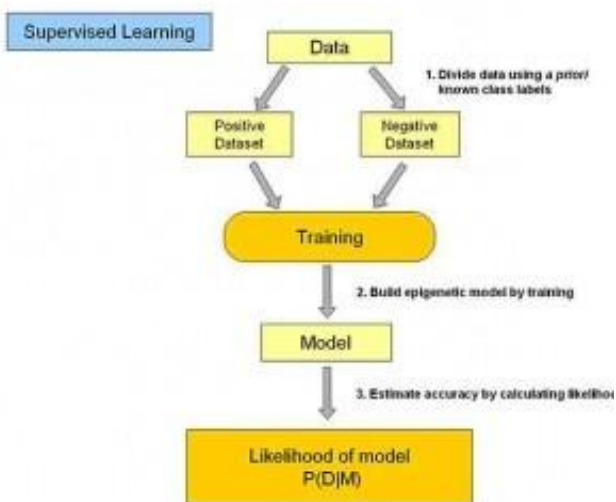
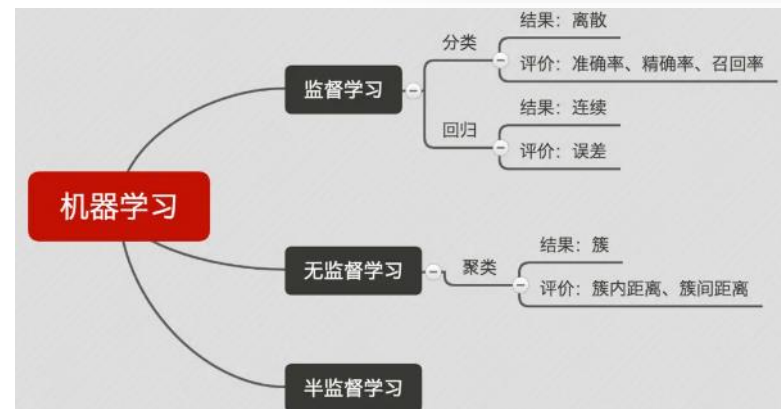
❖ 综合利用有类标的数据和没有类标的数据，来生成合适的分类函数，例如预测



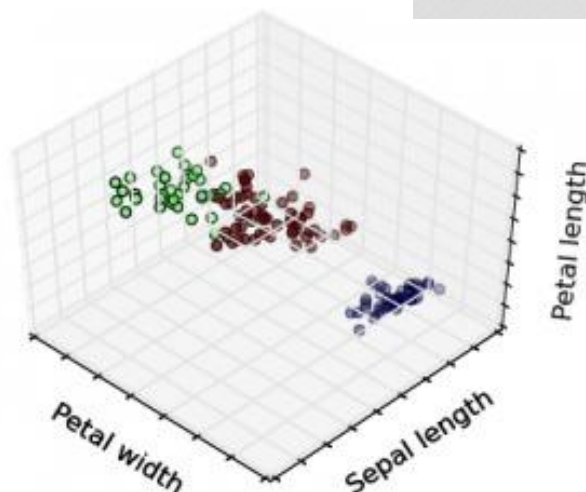
14.1 机器学习简介



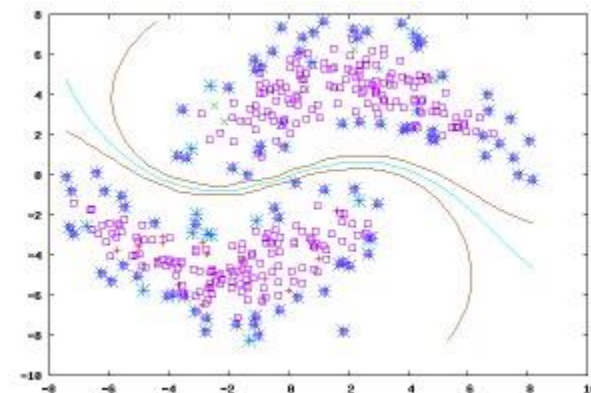
机器学习方法分类



监督学习
分类、回归



非监督学习
聚类



非监督学习
预测

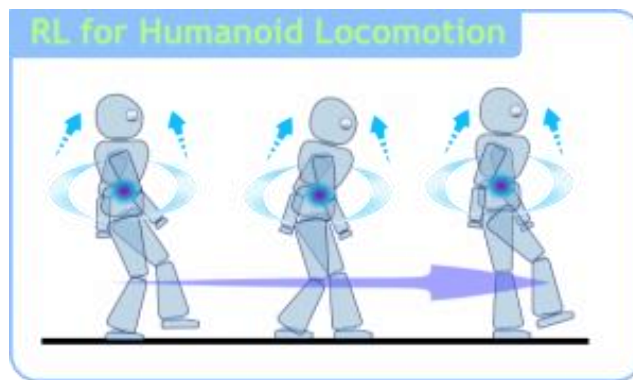


14.1 机器学习简介



机器学习之强化学习RL

- 输入数据作为对模型的反馈，不像监督模型那样，输入数据仅仅是作为一个检查模型对错的方式，在强化学习下，输入数据直接反馈到模型，模型必须对此立刻作出调整。
- 常见的应用场景包括动态系统以及机器人控制等。





14.1 机器学习简介



机器学习基本原理

- 目的：让机器可以自主获得事物规律。
- 过程：要让机器可以“学习”，必须将生活中的数据（包括但不限于图像、文字、语音）数值化，将不同事物的变化和关联转化为运算。
- 原理：概念和数值、关系和运算可以相互映射





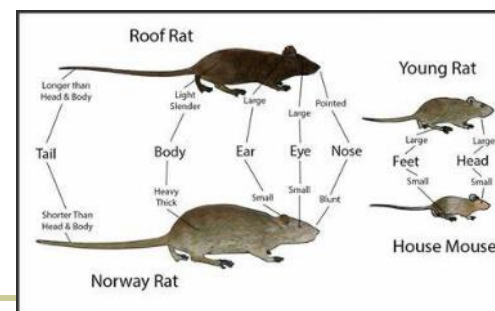
14.1 机器学习简介



机器学习基本原理

例如：猫妈妈教小猫认老鼠

- 小猫：一个机器 (Machine)
- 学习 (Learning)：小猫成为老鼠分类器的过程
- 猫妈妈给小猫看的照片：数据 (Data)
- 猫妈妈让小猫注意的几点：分类器的特征 (Feature)
- 学习的结果：成为了老鼠分类器 -> 模型 (Model)
- 小猫的思考过程：算法 (Algorithm)



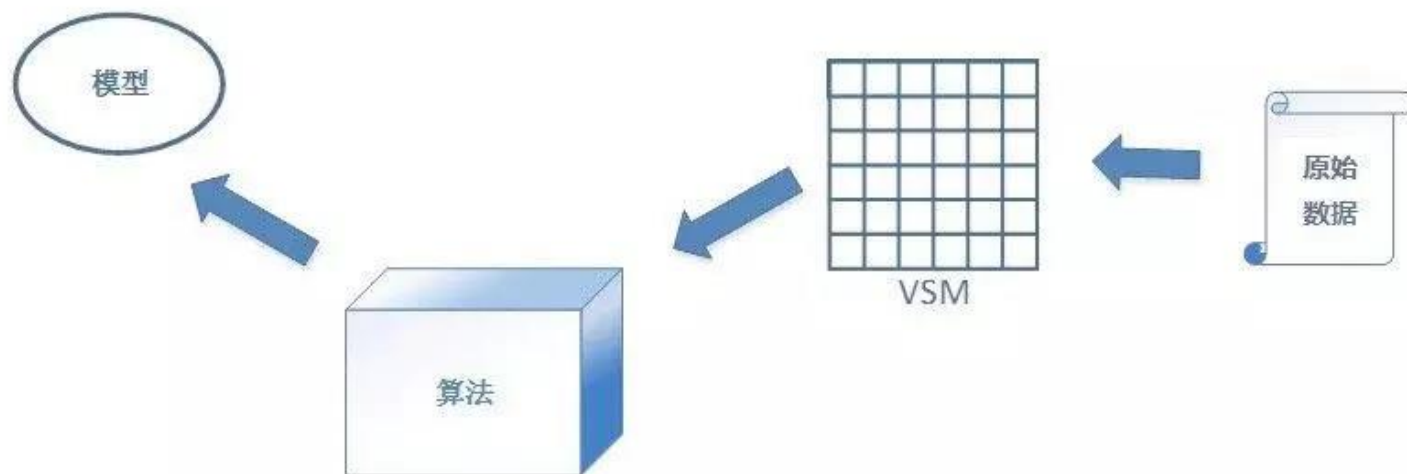


14.1 机器学习简介



机器学习三要素

数据 + 算法 => 模型



数据、模型、算法



14.2 机器学习经典算法与框架



14.2 机器学习经典算法与框架



机器学习算法

经典算法

线性回归、逻辑回归、决策树、随机森林、K最近邻 (KNN)、K均值 (KMeans) 聚类、朴素贝叶斯、支持向量机 (SVM)、降维 (PCA)、集成学习 (Ada Boosting、Gradient Boosting)、关联分析等

神经网络(Neural Network, NN)算法

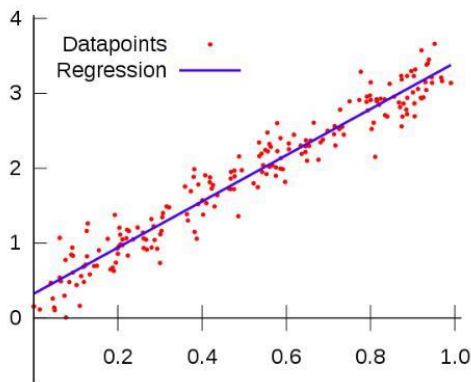
单层神经网络 (感知器)、反向传播 (BP)
多层神经网络 (深度学习)：卷积神经网络(CNN)、循环神经网络(RNN)、深度神经网络 (DNN)、长短期记忆网络(LSTM)



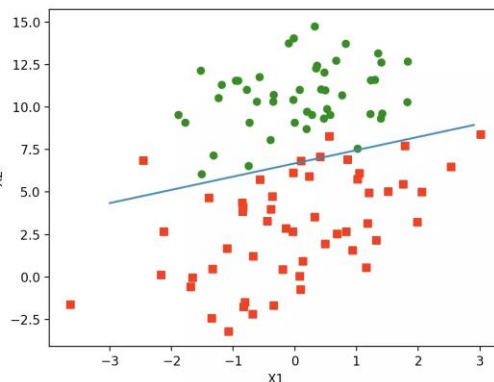
14.2 机器学习经典算法与框架



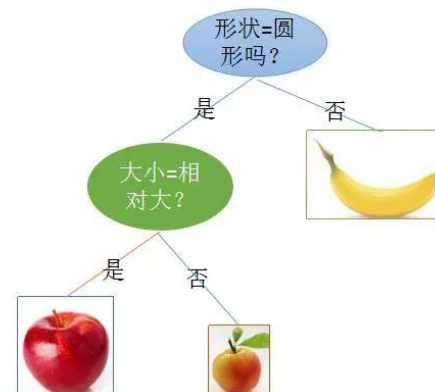
机器学习经典算法



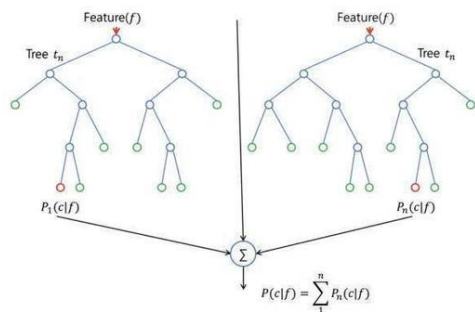
线性回归



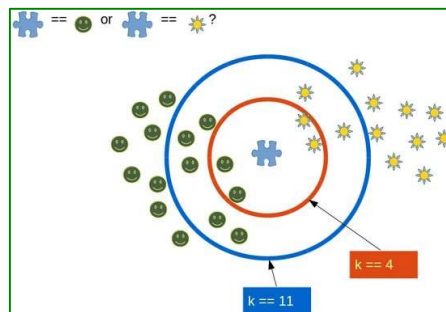
逻辑回归



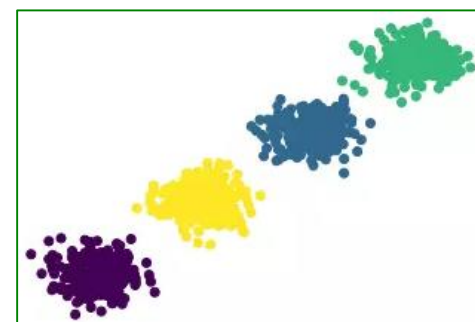
决策树



随机森林



KNN



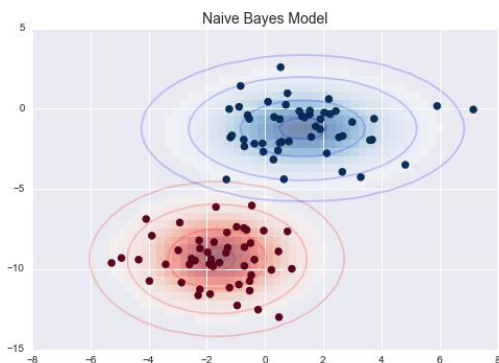
KMeans



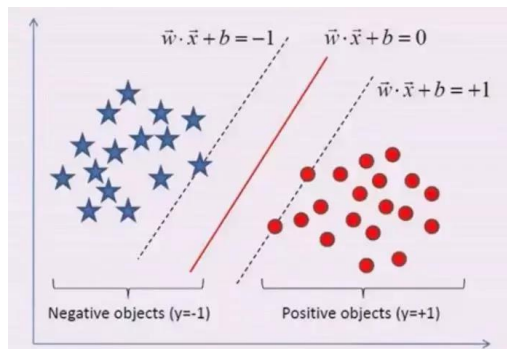
14.2 机器学习经典算法与框架



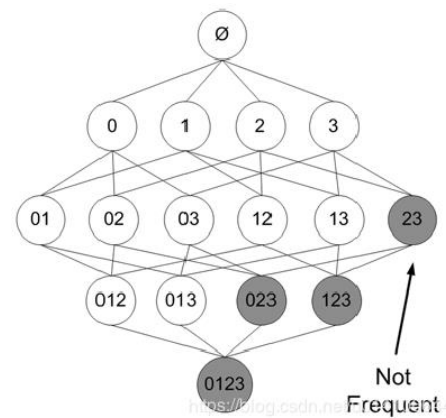
机器学习经典算法



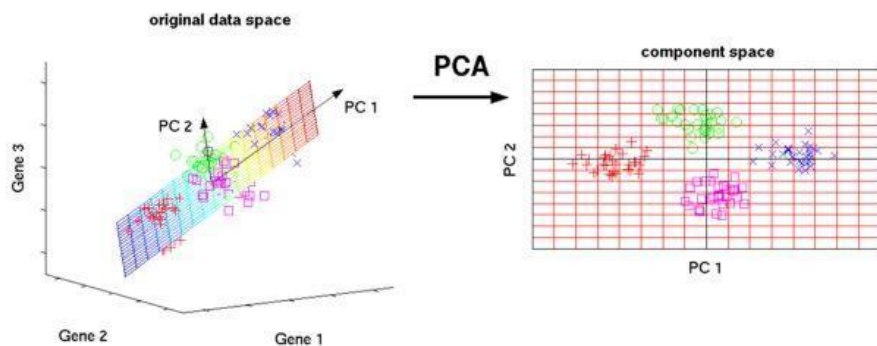
朴素贝叶斯



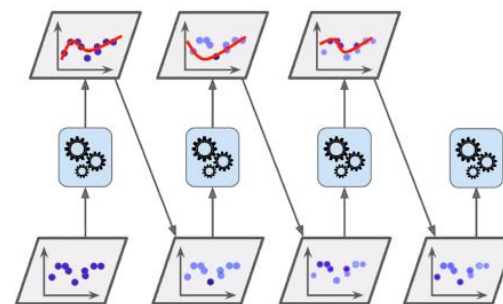
支持向量机



关联分析



PCA降维



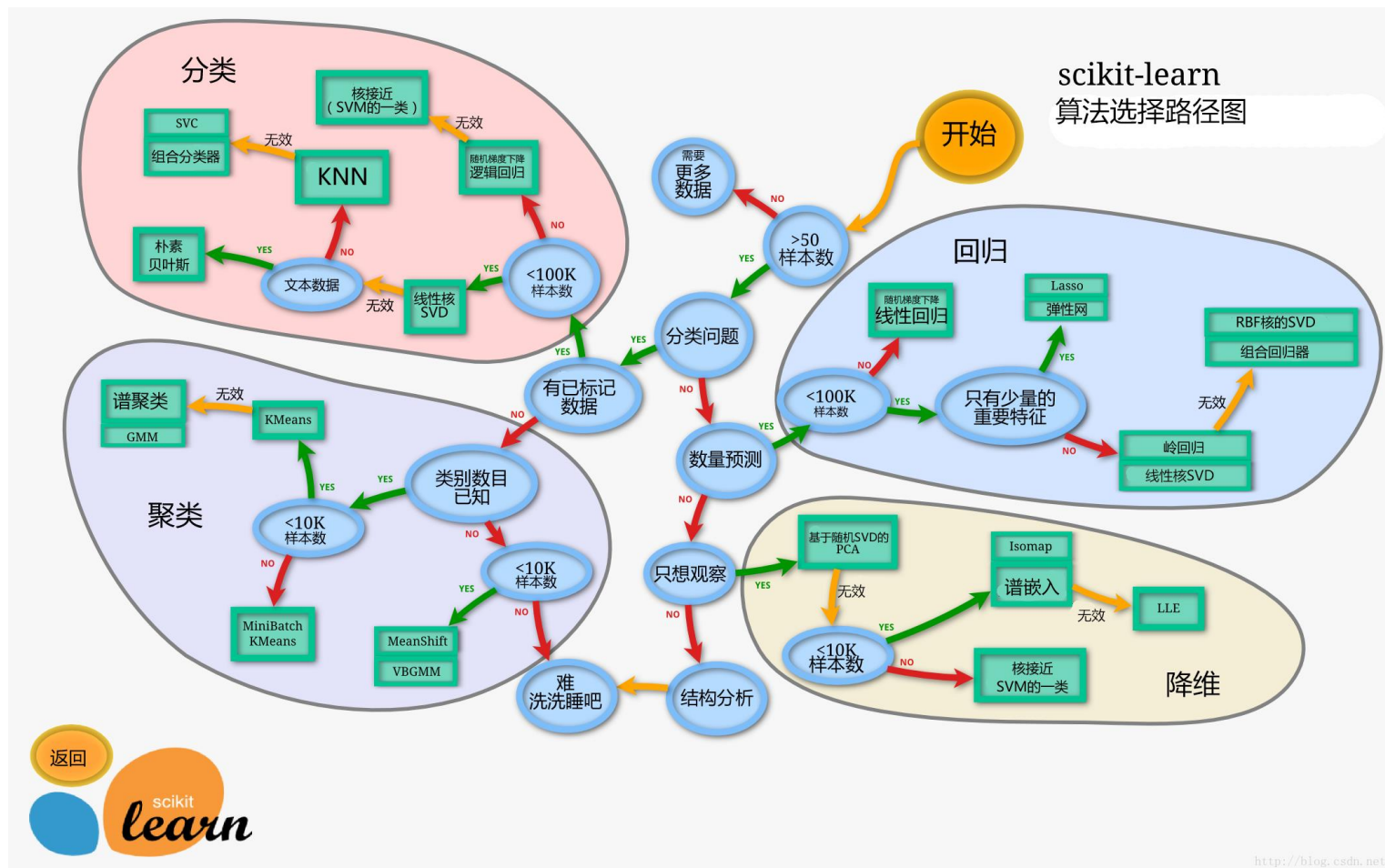
Ada Boosting



14.2 机器学习经典算法与框架



机器学习经典算法选择





14.2 机器学习经典算法与框架



机器学习经典算法

Python实现

<https://www.cnblogs.com/Anita9002/p/11219577.html>

sklearn python API

LinearRegression

```
1 from sklearn.linear_model import LinearRegression # 线性回归 #
2 module = LinearRegression()
3 module.fit(x, y)
4 module.score(x, y)
5 module.predict(test)
```

LogisticRegression

```
1 from sklearn.linear_model import LogisticRegression # 逻辑回归 #
2 module = LogisticRegression()
3 module.fit(x, y)
4 module.score(x, y)
5 module.predict(test)
```

KNN

```
1 from sklearn.neighbors import KNeighborsClassifier # K近邻 #
2 from sklearn.neighbors import KNeighborsRegressor
3 module = KNeighborsClassifier(n_neighbors=6)
4 module.fit(x, y)
5 predicted = module.predict(test)
6 predicted = module.predict_proba(test)
```

SVM

```
1 from sklearn import svm # 支持向量机 #
2 module = svm.SVC()
3 module.fit(x, y)
4 module.score(x, y)
5 module.predict(test)
6 module.predict_proba(test)
```

naive_bayes

```
1 from sklearn.naive_bayes import GaussianNB # 朴素贝叶斯分类器 #
2 module = GaussianNB()
3 module.fit(x, y)
4 predicted = module.predict(test)
```

DecisionTree

```
1 from sklearn import tree # 决策树分类器 #
2 module = tree.DecisionTreeClassifier(criterion='gini')
3 module.fit(x, y)
4 module.score(x, y)
5 module.predict(test)
```

K-Means

```
1 from sklearn.cluster import KMeans # kmeans聚类 #
2 module = KMeans(n_clusters=3, random_state=0)
3 module.fit(x, y)
4 module.predict(test)
```

RandomForest

```
1 from sklearn.ensemble import RandomForestClassifier # 随机森林 #
2 from sklearn.ensemble import RandomForestRegressor
3 module = RandomForestClassifier()
4 module.fit(x, y)
5 module.predict(test)
```

GBDT

```
1 from sklearn.ensemble import GradientBoostingClassifier # Gradient Boosting
2 from sklearn.ensemble import GradientBoostingRegressor
3 module = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, max_
4 module.fit(x, y)
5 module.predict(test)
```

PCA

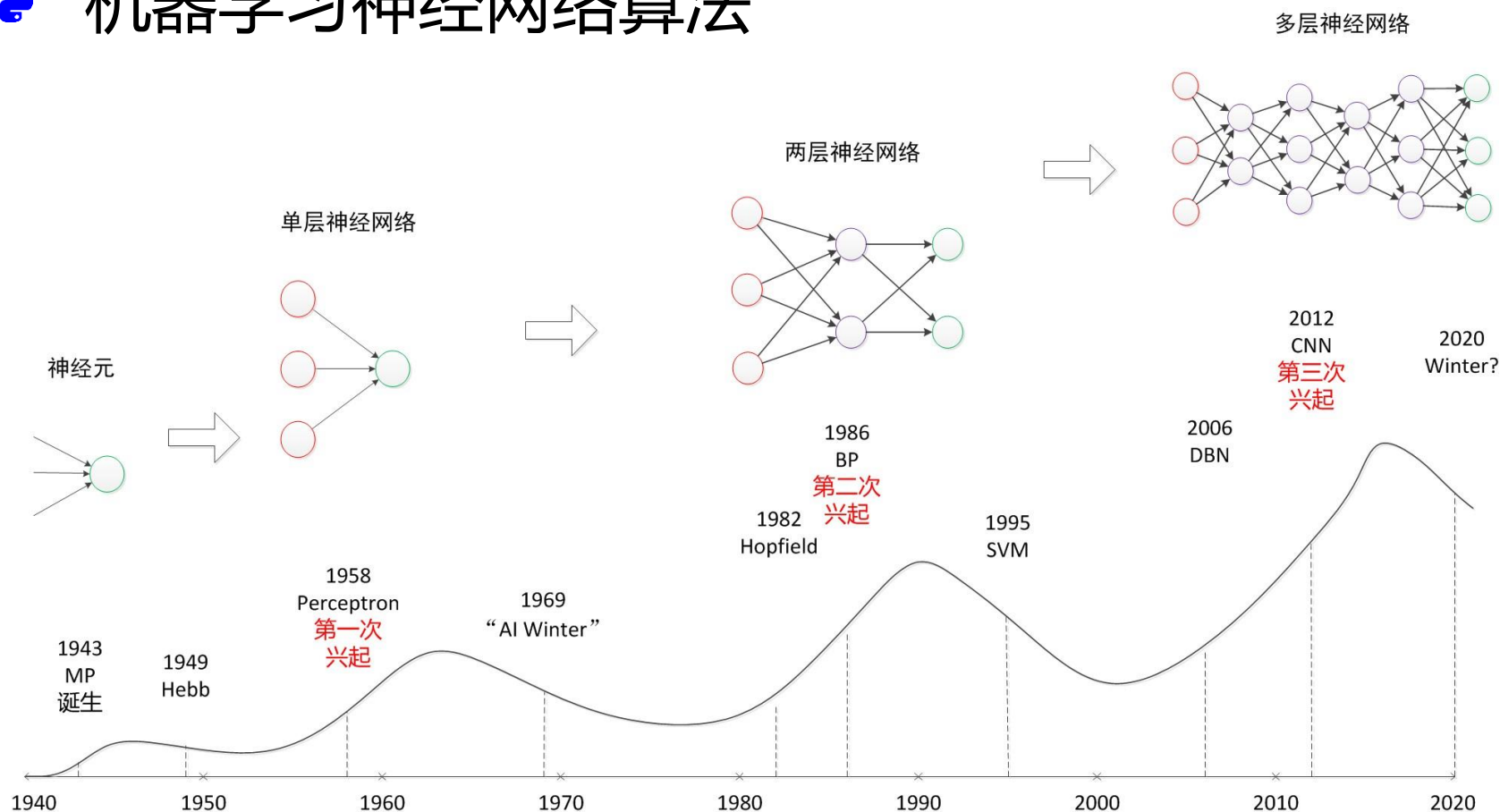
```
1 from sklearn.decomposition import PCA # PCA特征降维 #
2 train_reduced = PCA.fit_transform(train)
3 test_reduced = PCA.transform(test)
```



14.2 机器学习经典算法与框架



机器学习神经网络算法





14.2 机器学习经典算法与框架



机器学习神经网络算法

单层感知机的Python实现

<https://blog.csdn.net/song430/article/details/88718602>

反向传播算法的Python实现

https://blog.csdn.net/qq_28888837/article/details/84296673

卷积神经网络的Python实现

<https://www.cnblogs.com/yszd/p/10003087.html>

循环神经网络的Python实现

<https://blog.csdn.net/u014556057/article/details/85173830>



14.2 机器学习经典算法与框架



机器学习开源框架

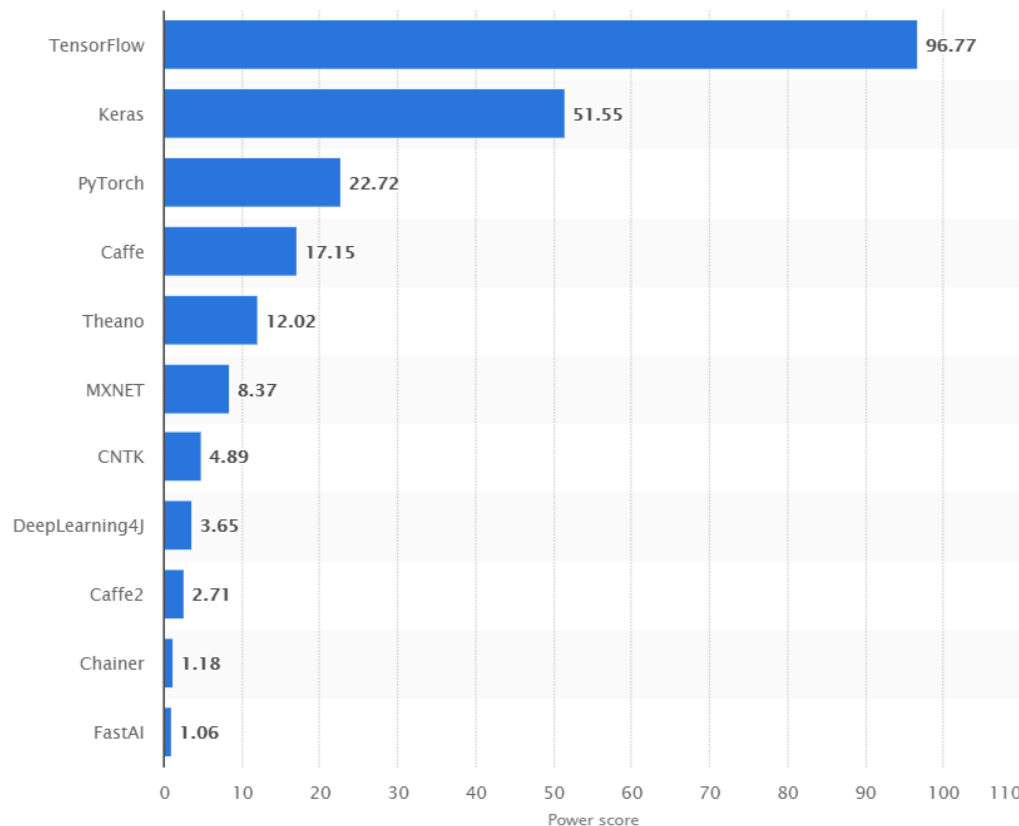
- ✚ **TensorFlow**: 谷歌开发, 多平台支持Python, CPU/GPU版本
- ✚ CNTK: 微软开发, 支持Python, 跨平台的CPU/GPU支持
- ✚ **PyTorch**: Facebook开发, 支持Python并具有 python 风格
- ✚ MXNet: 亚马逊官方深度学习库, 支持Python, 轻量便携灵活
- ✚ paddlepaddle: 百度飞桨, 支持Python, 精选模型+中文支持
- ✚ DL4J: 基于Java, 用于多项商业和科研应用, 不支持Python
- ✚ **Keras**: 基于Python, 可作为Tensorflow等的高阶应用程序接口
- ✚ Theano: 最老牌的底层深度学习Python库, 偏向研究, 已停更
- ✚ Caffe: 老牌库, UCB开发, 清晰而高效, 支持Python
- ✚ **Scikit Learn**: 基于SciPy的通用机器学习Python库, 自由学术



14.2 机器学习经典算法与框架



机器学习开源框架












14.2 机器学习经典算法与框架



TensorFlow

-  主要用于进行机器学习和深度神经网络研究，使用静态图
-  基础的底层系统，可应用于众多领域
-  依靠Google迅速成为如今用户最多的深度学习框架
-  缺点：设计复杂、接口晦涩多变、文档混乱，多依赖Keras、TensorLayer等第三方库提供易用接口
-  不完美但最流行的深度学习框架，社区强大，适合生产环境

PyTorch

-  简洁优雅且高效快速的框架，使用动态图，依靠FaceBook
-  在研究领域发展迅速，各类深度学习问题都有利用PyTorch实现的解决方案在GitHub上开源

<https://www.jianshu.com/p/cbba871100a6>

<https://www.toutiao.com/a6761203614636573197/>



14.2 机器学习经典算法与框架



- ✧ TensorFlow的设计是Make It Complicated
- ✧ PyTorch的设计是Keep it Simple, Stupid
- ✧ 使用TensorFlow能找到很多别人的代码
- ✧ 使用PyTorch能轻松实现自己的想法



14.3 PYTHON机器学习实例



14.3 Python机器学习实例



机器学习入门经典案例（1）

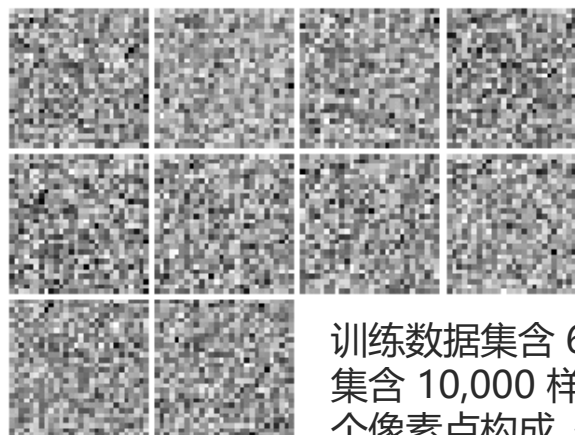
MNIST数据集

Yann LeCun提供的手写数字数据库，官方地址

✓ <http://yann.lecun.com/exdb/mnist/>

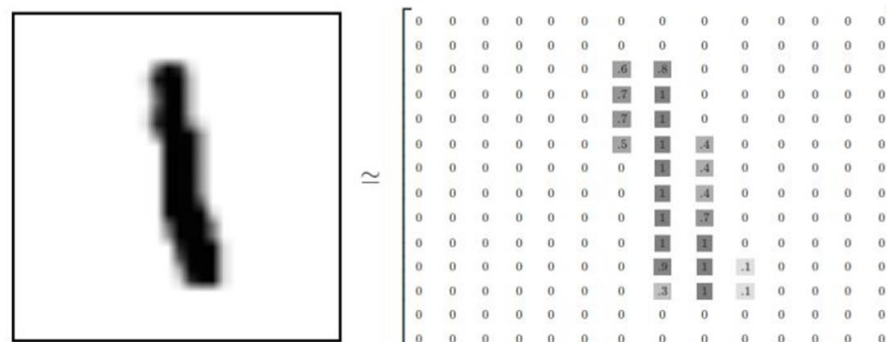
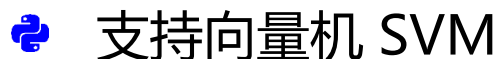
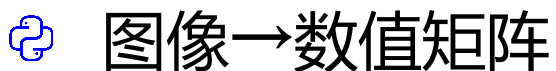
机器学习界的“Hello World”

数据集扩展：QMNIST、Fashion-MNIST



MNIST手写数字数据集

训练数据集含 60,000 个样本，测试数据集含 10,000 样本。每张图片由 28 x 28 个像素点构成，像素点用灰度值表示。



<https://gitee.com/tysunrui/mnist-classification>



14.3 Python机器学习实例



🔗 MNIST数据集

🔗 机器学习模块: PyTorch + Scikit-learn

🔗 PyTorch



🔗 官网: <https://pytorch.org/>

🔗 安装

- ✓ `pip3 install torch==1.3.1+cpu torchvision==0.4.2+cpu -f https://download.pytorch.org/whl/torch_stable.html`

PyTorch Build	Stable (1.3)		Preview (Nightly)	
Your OS	Linux	Mac	Windows	
Package	Conda	Pip	LibTorch	Source
Language	Python 2.7	Python 3.5	Python 3.6	Python 3.7 C++
CUDA	9.2	10.1	None	
Run this Command:	<code>pip3 install torch==1.3.1+cpu torchvision==0.4.2+cpu -f https://download.pytorch.org/whl/torch_stable.html</code>			



14.3 Python机器学习实例



❧ MNIST数据集

❧ 机器学习模块: PyTorch + Scikit-learn

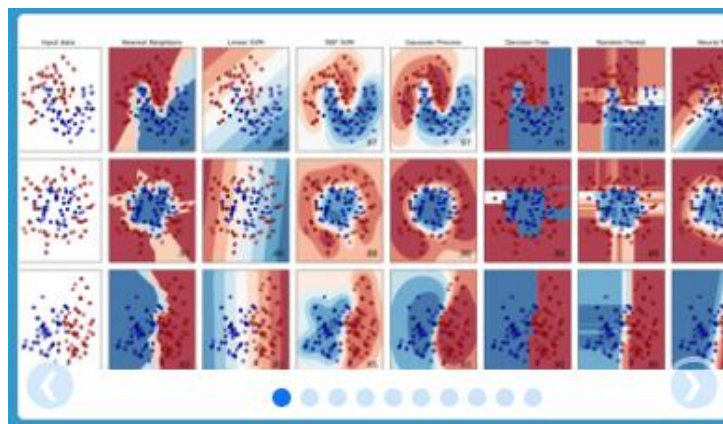
❧ Scikit-learn



❧ 官网: <https://scikit-learn.org/>

❧ 安装

✓ `pip install -i https://pypi.tuna.tsinghua.edu.cn/simple scikit-learn`



scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license



14.3 Python机器学习实例



MNIST数据集

CNN算法

```
# MNIST_CNN.py
import torch
import torch.nn as nn
import torch.utils.data as Data
import torchvision
# 参数
EPOCH = 1          # 训练次数
BATCH_SIZE = 50     # 分批大小
LR = 0.001          # 学习率
train_data = torchvision.datasets.MNIST(
    root='./mnist/',
    train=True,
    transform=torchvision.transforms.ToTensor(),
    download=True,
)
# 训练数据
# 将图像转换为numpy
# 是否下载数据

print(train_data.train_data.size())    # (60000, 28, 28)
print(train_data.train_labels.size())  # (60000)
```




14.3 Python机器学习实例



MNIST数据集

CNN算法

```
# 载入训练数据
```

```
train_loader = Data.DataLoader(dataset=train_data,  
batch_size=BATCH_SIZE, shuffle=True)
```

```
# 选择2000个样本加速测试
```

```
test_data = torchvision.datasets.MNIST(root='./mnist/', train=False)  
test_x = torch.unsqueeze(test_data.test_data,  
dim=1).type(torch.FloatTensor)[:2000]/255.0  
test_y = test_data.test_labels[:2000]
```



14.3 Python机器学习实例



☼ MNIST数据集

☼ CNN算法

定义算法模型

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Sequential(
            nn.Conv2d(
                in_channels=1, out_channels=16,
                kernel_size=5,
                stride=1, padding=2,)),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2),
        )
        self.conv2 = nn.Sequential(
            nn.Conv2d(16, 32, 5, 1, 2),
            nn.ReLU(),
            nn.MaxPool2d(2),
        )
        self.out = nn.Linear(32 * 7 * 7, 10)
```

```
def forward(self, x):
    x = self.conv1(x)
    x = self.conv2(x)
    x = x.view(x.size(0), -1)
    output = self.out(x)
    return output, x
```

```
cnn = CNN()
print(cnn) # 网络架构
```



14.3 Python机器学习实例



🔗 MNIST数据集

🔗 CNN算法

```
optimizer = torch.optim.Adam(cnn.parameters(), lr=LR)      # 参数优化
loss_func = nn.CrossEntropyLoss()                          # 损失函数
# 训练与测试
for epoch in range(EPOCH):
    for step, (b_x, b_y) in enumerate(train_loader):        # 分批
        output = cnn(b_x)[0]                                # cnn output
        loss = loss_func(output, b_y)                       # cross entropy loss
        optimizer.zero_grad()                               # clear gradients for this training
    step
    loss.backward()                                          # backpropagation, compute gradients
    optimizer.step()                                        # apply gradients
    if step % 50 == 0:
        test_output, last_layer = cnn(test_x)
        pred_y = torch.max(test_output, 1)[1].data.numpy()
        accuracy = float((pred_y == test_y.data.numpy()).astype(int).sum())
        / float(test_y.size(0))
        print('Epoch: ', epoch, '| train loss: %.4f' % loss.data.numpy(),
              '| test accuracy: %.2f' % accuracy)
```



14.3 Python机器学习实例



MNIST数据集

CNN算法

显示10次测试结果

```
test_output, _ = cnn(test_x[:10])
pred_y = torch.max(test_output, 1)[1].data.numpy()
print(pred_y, 'prediction number')
print(test_y[:10].numpy(), 'real number')
```

```
Epoch: 0 | train loss: 2.3182 | test accuracy: 0.14
Epoch: 0 | train loss: 0.4617 | test accuracy: 0.78
Epoch: 0 | train loss: 0.4063 | test accuracy: 0.89
Epoch: 0 | train loss: 0.3383 | test accuracy: 0.91
Epoch: 0 | train loss: 0.2967 | test accuracy: 0.93
Epoch: 0 | train loss: 0.0534 | test accuracy: 0.94
Epoch: 0 | train loss: 0.1174 | test accuracy: 0.94
Epoch: 0 | train loss: 0.1279 | test accuracy: 0.95
Epoch: 0 | train loss: 0.2413 | test accuracy: 0.95
Epoch: 0 | train loss: 0.1353 | test accuracy: 0.95
Epoch: 0 | train loss: 0.1051 | test accuracy: 0.96
Epoch: 0 | train loss: 0.0225 | test accuracy: 0.97
Epoch: 0 | train loss: 0.2445 | test accuracy: 0.97
```

```
Epoch: 0 | train loss: 0.0419 | test accuracy: 0.97
Epoch: 0 | train loss: 0.0877 | test accuracy: 0.97
Epoch: 0 | train loss: 0.0690 | test accuracy: 0.97
Epoch: 0 | train loss: 0.0417 | test accuracy: 0.97
Epoch: 0 | train loss: 0.0504 | test accuracy: 0.97
Epoch: 0 | train loss: 0.0291 | test accuracy: 0.97
Epoch: 0 | train loss: 0.1164 | test accuracy: 0.98
Epoch: 0 | train loss: 0.1374 | test accuracy: 0.97
Epoch: 0 | train loss: 0.0714 | test accuracy: 0.97
Epoch: 0 | train loss: 0.0601 | test accuracy: 0.98
Epoch: 0 | train loss: 0.1312 | test accuracy: 0.97
[7 2 1 0 4 1 4 9 5 9] prediction number
[7 2 1 0 4 1 4 9 5 9] real number
```



14.3 Python机器学习实例



MNIST数据集

其他算法

 mnist_ML.py

Model	epoch 1	epoch 3	epoch 5	epoch 7	epoch 10	epoch 20
Logistic Regression	.888	.901	.904	.904	.904	.911
SVM	.258	.373	.497	.491	.516	.707
KNN	.962	.962	.962	.962	.962	.962
MLP	.891	.903	.902	.897	.902	.903
CNN	.989	.992	.993	.996	.981	.993
RNN	.952	.973	.981	.973	.976	.959

不同算法的准确率比较



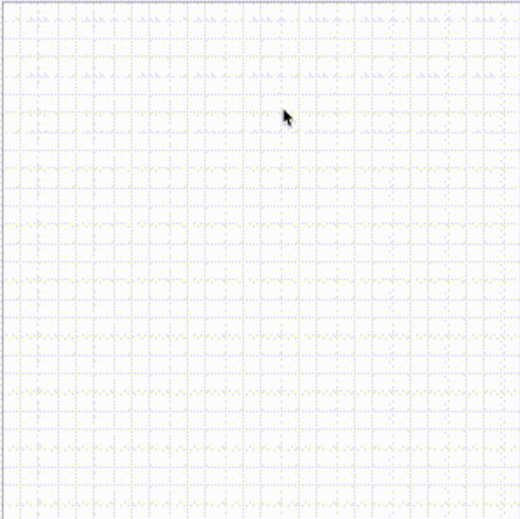
14.3 Python机器学习实例




MNIST数据集

应用

draw a digit here!



input:



output:

	simple	convolutional
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		

https://github.com/LinguoLi/mnist_tutorial



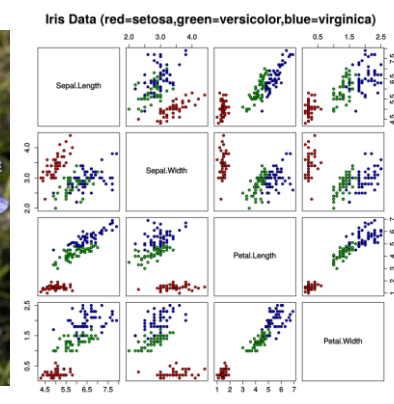
14.3 Python机器学习实例



机器学习入门经典案例（2）

鸢尾花卉数据集

- 常用的分类实验数据集，由Fisher收集整理
- 由3种不同类型的鸢尾花的各50个样本数据构成
- 数据地址：<http://archive.ics.uci.edu/ml/datasets/Iris>



鸢尾花卉数据集

数据集内包含 3 类共 150 条记录，每类各 50 个数据，每条记录都有 4 项特征：花萼长度、花萼宽度、花瓣长度、花瓣宽度。



14.3 Python机器学习实例



鸢尾花卉数据集

库: sklearn

算法: 逻辑回归&K近邻

```
# Iris_sklearn.py
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
import numpy as np

# 载入数据集
iris_dataset = load_iris()
# 数据划分
x_train, x_test, y_train, y_test =
train_test_split(iris_dataset['data'], iris_dataset['target'],
test_size=0.2, random_state=0)
print("训练集:{}\n测试集:{}".format(len(x_train), len(x_test)))
```



14.3 Python机器学习实例



鸢尾花卉数据集

```
# K近邻算法
print("-----KNN算法-----")
# 设置邻居数
knn = KNeighborsClassifier(n_neighbors=1)
# 构建基于训练集的模型
knn.fit(x_train, y_train)
# 得出测试集x_test测试集的分數
print("测试精
度:{:.2f}".format(knn.score(x_test, y_test)))
print("\n")
# 逻辑回归算法
print("-----逻辑回归算法-----")
log = LogisticRegression(penalty='l2',
solver="liblinear", multi_class="auto")
log.fit(x_train, y_train)
# 准确率
print("测试精
度:{:.2f}".format(log.score(x_test, y_test)))
```

训练集:120

测试集:30

-----KNN算法-----

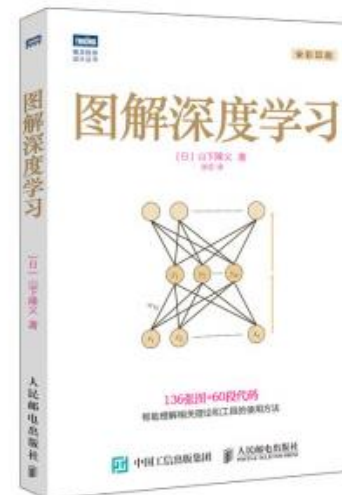
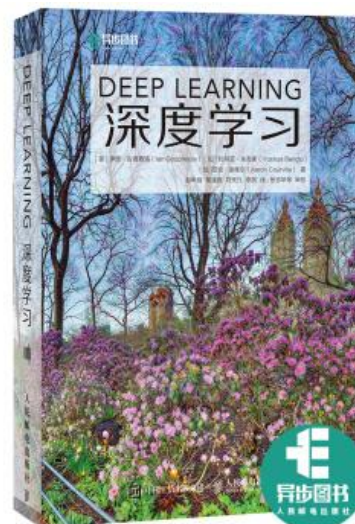
测试精度:1.00

-----逻辑回归算法-----

测试精度:0.97

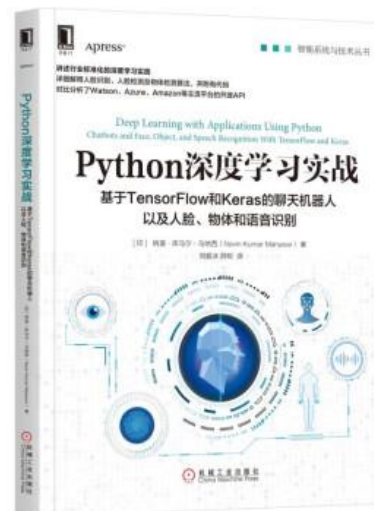
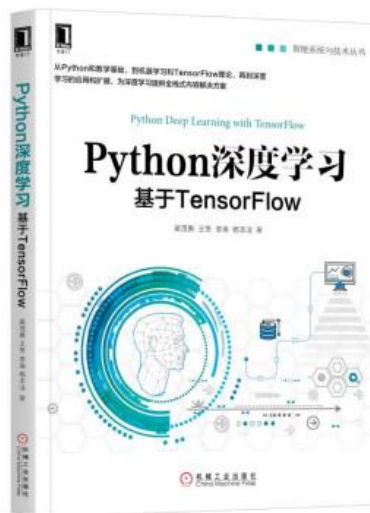
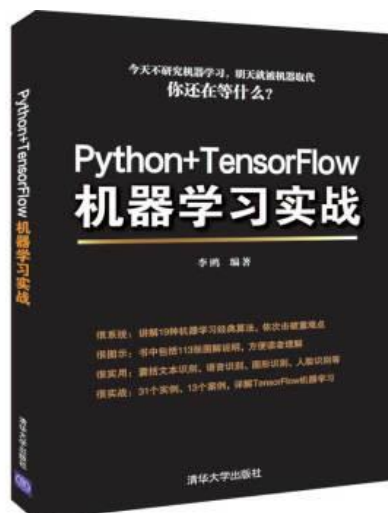
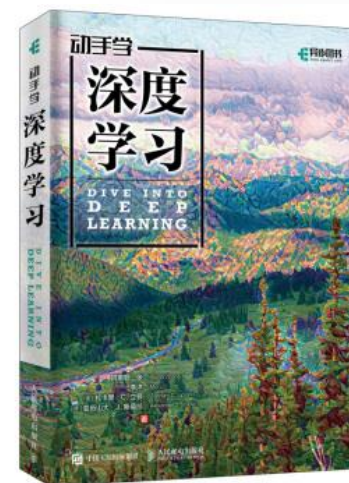
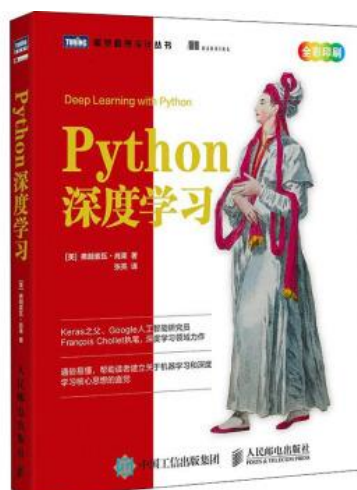


进阶书籍





进阶书籍





推荐教程



 吴恩达Deeplearning.ai



 https://mooc.study.163.com/university/deeplearning_ai#/c

 麻省理工 深度学习

 <https://deeplearning.mit.edu>

 微软 AI school

 <https://aischool.microsoft.com/en-us>

 Google & Udacity: 深度学习

 <https://cn.udacity.com/course/deep-learning--ud730>

 CMU: 深度学习入门

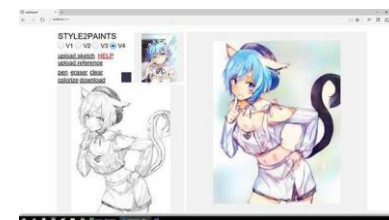
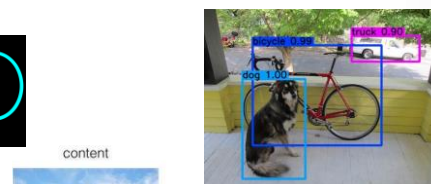
 <https://simons.berkeley.edu/talks/tutorial-deep-learning>



有趣案例



- 目标检测 YOLOv3 
- 风格迁移/黑白照片上色
- 线稿上色 Style2Paints
- 搜索引擎Magi
- 无损放大BigJPG
- 打游戏 SerpentAI
- 翻译宝宝哭声 Maggie App














14.2 机器学习经典算法与框架



机器学习资料

-  <https://www.cnblogs.com/lsgsanxiao/p/6955502.html>
-  <https://github.com/scutan90/DeepLearning-500-questions>
-  <https://github.com/exacity/deeplearningbook-chinese>
-  https://github.com/lawlite19/MachineLearning_Python
-  <https://github.com/trekhleb/homemade-machine-learning>
-  <https://github.com/rasbt/python-machine-learning-book>
-  <https://github.com/josephmisiti/awesome-machine-learning>
-  <https://github.com/apacheecn/AiLearning>
-  <https://github.com/d2l-ai/d2l-zh>





本课内容





第14章 Python机器学习

14.1 机器学习简介

-  机器学习、深度学习与人工智能
-  基本原理

14.2 机器学习经典算法与框架

-  算法简介及Python实现
-  框架简介及比较

14.3 Python机器学习实例

-  MNIST
-  Iris Flower



练习题



将前13章未完成的作业
继续完成。

.py代码文件和结果文件打包(姓名
+学号/章节号/作业代码)发送到
python_xxmu@163.com

```
./张三2019001/  
├── 01  
│   ├── 1.1.py  
│   └── 1.2.py  
├── 02  
│   └── 2.1.py  
├── 03  
│   └── 3.1.py  
├── 04  
│   └── 4.1.py  
├── 05  
│   └── 5.1.py  
├── 06  
│   └── 6.1.py  
├── 07  
│   └── 7.1.py  
├── 08  
│   └── 8.1.py  
├── 09  
│   └── 9.1.py  
├── 10  
│   └── 10.1.py  
├── 11  
│   └── 11.1.py  
├── 12  
│   └── 12.1.py  
└── 13  
    └── 13.1.py
```



下周安排



- ❖ 期末考试 (60%)
 - ❖ 开卷笔试 (仅限于携带纸质材料)
 - ❖ 内容为二级Python语言考试大纲结合课堂内容
 - ❖ 考试时间: 2019年12月20日19:00~20:20

编程辣么好，还等什么？开始学习吧！



Programing is an Art