如何实现机器学习算法

机器学习的概念及分类

概念：机器学习是一类算法的总称，是从大量历史数据中挖掘出其中隐含的规律（寻找一个函数，这个函数可能过于复杂，以至于不太方便写出表达式），并用于定量预测或者定性预测或者做一个动作。

数据挖掘的工具—机器学习

机器学习的概念及分类

按学习理论划分：有监督学习，半监督学习，无监督学习，强化学习

有监督学习：训练样本带有分类标签

4.4,3.0,1.3,0.2,Iris-setosa （山鸢尾）
5.1,3.4,1.5,0.2,Iris-setosa
6.4,3.2,4.5,1.5,Iris-versicolor （变色鸢尾）
6.9,3.1,4.9,1.5,Iris-versicolor

前面这些数值型数据就是特征数据，后面的字符型数据就是分类标签数据。当用这些数据训练好一个模型后，再输入某个样本的特征数据，就可以预测出分类。

机器学习的概念及分类

半监督学习：训练样本部分有分类标签，部分无分类标签

4.4,3.0,1.3,0.2,Iris-setosa （山鸢尾）
5.1,3.4,1.5,0.2
6.4,3.2,4.5,1.5,Iris-versicolor （变色鸢尾）
6.9,3.1,4.9,1.5

机器学习的概念及分类

无监督学习：训练样本全部无分类标签

4.4,3.0,1.3,0.2 （山鸢尾）
5.1,3.4,1.5,0.2
6.4,3.2,4.5,1.5 （变色鸢尾）
6.9,3.1,4.9,1.5

强化学习：是一个学习最优策略，可以让本体在特定环境中，根据当前状态，做出行动，从而获得最大回报。如象棋对弈。

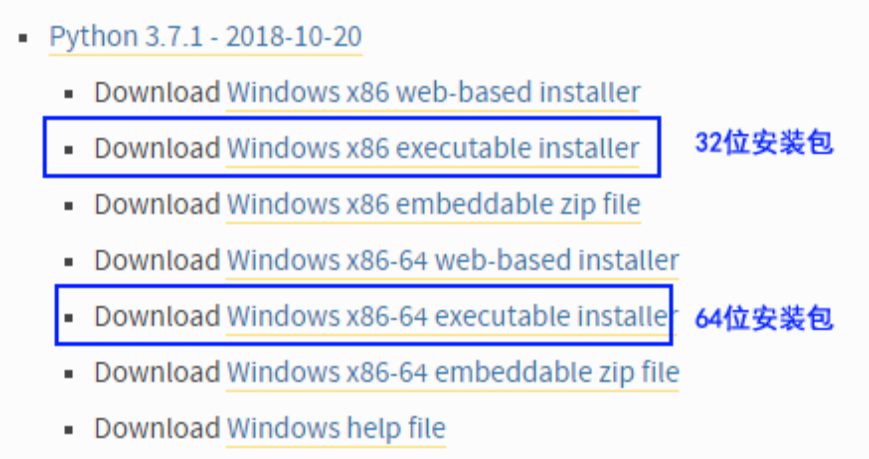
任务：编写（读懂）机器学习算法

具体实践过程

1.搭建Python环境 （10分钟）

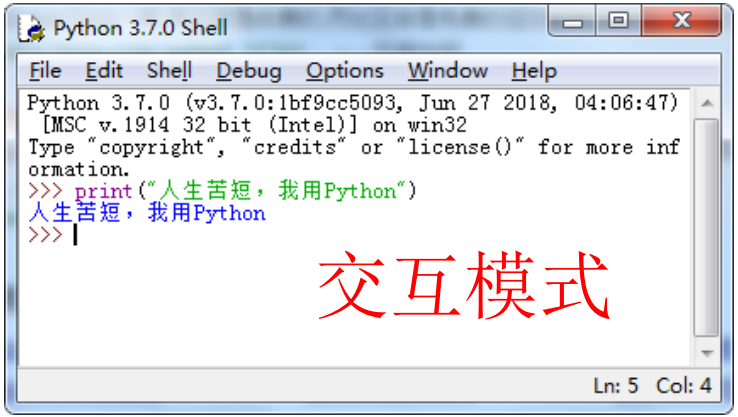
（1）你可以阅读：第二章 Python环境安装

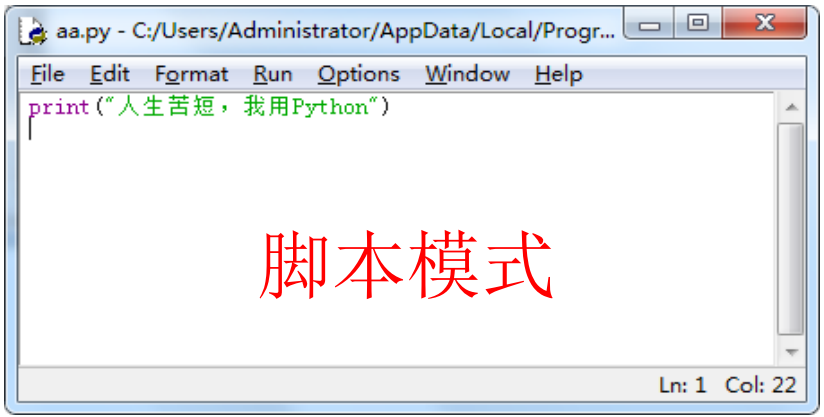
（2）你还可以参考下面的方案： 首先访问Python官网首页（www.Python.org），选择“Downloads”->“windows”下载安装包。



具体实践过程

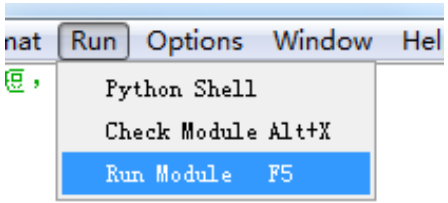
在开发大中型项目时推荐使用pycharm编辑器。好处很多，请你慢慢体会。





运行程序方式：直接输出结果

运行程序方式：



安装绘图模块：pip install matplotlib

具体实践过程

2. 训练集分析

该数据集测量了所有150个样本的4个特征，3个分类，每个分类50个样本

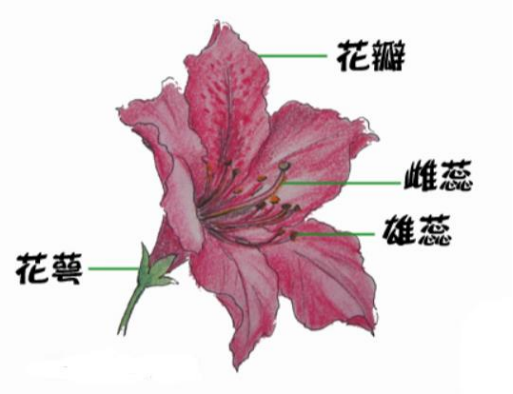
4.4， 3.0， 1.3， 0.2， Iris-setosa
5.1， 3.4， 1.5， 0.2， Iris-setosa
6.4， 3.2， 4.5， 1.5， Iris-versicolor
6.9， 3.1， 4.9， 1.5， Iris-versicolor
6.3， 3.3， 6.0， 2.5， Iris-virginica
5.8， 2.7， 5.1， 1.9， Iris-virginica

↑ 花萼长度　↑ 花萼宽度　↑ 花瓣长度　↑ 花瓣宽度　↑ 分类

山鸢尾

变色鸢尾

维吉尼亚鸢尾



具体实践过程

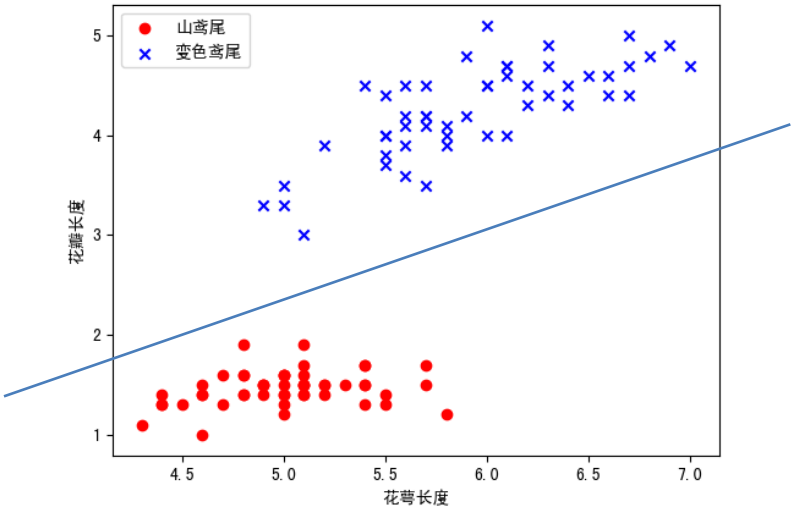
为简单起见，我们只使用花萼长度、花瓣长度作为特征数据进行训练。

训练目标是找到一条直线：$w \cdot x + b = 0$，
（w为法向量，b为截距，x为特征向量）能把数据集按分类标签划分为两部分。

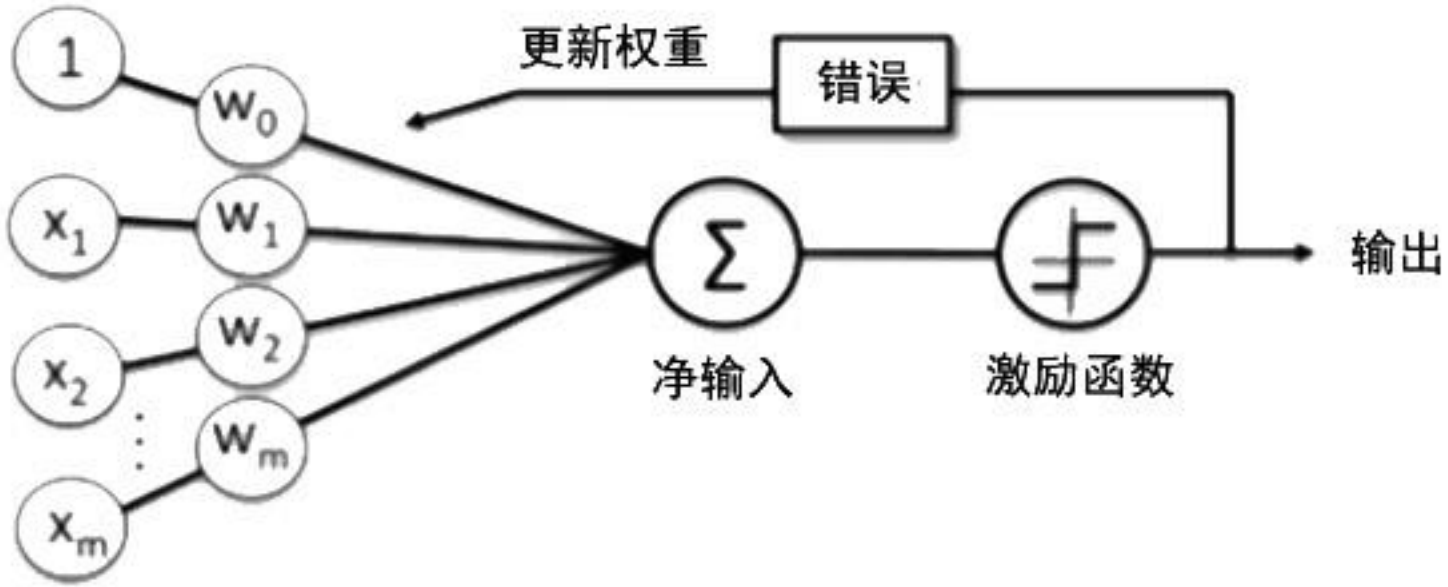
令$w = (w_1, w_2), x = (x_1, x_2), b = w_0 \times 1$，则直线方程变形为：

$$(w_0, w_1, w_2)(1, x_1, x_2) = 0$$

找到一个w，对所有特征数据都能按分类标签进行分类，即：满足：$w.x > 0$或$w.x < 0$



3. 模型训练（建模）过程描述



3. 模型训练（建模）过程描述

学习过程：

1.设$w=(w_0,w_1,w_2)$，w_0为偏置项，w_1,w_2为权重。初始化:$w=(0,0,0)$

2.对每个特征数据计算估计值，如：$sign(0\times1+0\times4.4+0\times1.3)=0$

3.比较估计值与真实值，若不相等，调整W重新计算，直到估计值与真实值相等为止。

4.重复2,3直到所有特征数据的估计值都与真实值相等。

3. 模型训练（建模）过程描述

这个过程就是机器学习的过程，也即**模型训练**。这些用于训练的数据称为**训练集**。为了验证模型的正确率，我们需要准备另一组带有分类标签的数据，用训练好的模型（W）计算估计值，与真实值比较，统计正确率，从而评估模型的好坏。这样的数据称为**测试集**。如果模型通过了测试，就可以用于实际生产中，产生经济效益。

预测分类的方法：需要预测的特征数据与W计算估计值。

调整W的方法：

$w_j = w_j + \eta(A - E)x_j$ （w_j为权重分量，x_j为特征分量，η为学习速率，A为真实值，E为估计值）

4.面向过程编程实现（定义主逻辑，8分钟）

```
01 def crDataSet(datafile):                    #构造数据集
02     pass
03 def train():                                #训练
04     pass                                    #占位，啥也不做
05 def test():                                 #测试
06     pass
07 if __name__ == "__main__":
08     train_data = crDataSet('train.data')
09     test_data = crDataSet('test.data')
10     train()
11     test()
```

面向过程程序设计思想——以函数为中心”

具体实践过程（面向过程）

5. 面向过程编程实现（定义激励函数，10分钟）

```
def sign(value):              #定义函数sign
    if value>0:               #如果value大于0
        return 1              #返回 1
    elif value<0:             #否则如果value小于0
        return -1             #返回 -1
    else:                     #否则
        return 0              #返回0
```

Python 实现

```
int sign(float value)
{
    if (value>0) {
        return 1;
    }else if(value<0){
        return -1;
    }else{
        return 0;
    }
}
```

C 实现

6.面向过程编程实现（定义构造数据集函数,15分钟）

```
01 def crDataSet(datafile):
02     result = []                                  #定义空列表
03     with open(datafile,'r') as file:             #读取数据文件
04         data = file.read().splitlines()          #按行分割
05         for line in data:                        #遍历所有行
06             temp = []                            #定义临时列表
07             items = line.split(',')              #用,分割字符串
08             temp.append(float(items[0]))         #花萼长度
09             temp.append(float(items[2]))         #花瓣长度
10             if items[4] =='Iris-setosa':
11                 temp.append(1)
12             else:
13                 temp.append(-1)
14             result.append(temp)                  #向列表追加元素
15     return result                                #返回列表
```

7.面向过程编程实现（定义train函数，10分钟）

具体实践过程（面向过程）

```
01 def train():
02     error =1                    #是否有错误
03     while error:                #当错误存在时
04         error = 0
05         for item in train_data:                          #变量训练集中的每一项
06             outvalue = sign(dotProduct(item)             #计算估计值
07             while outvalue !=item[2]:                    #当估计值与真实值不相等时
08                 error = 1                                #存在错误
09                 upWeight(item,outvalue)                  #更新权重
10                 outvalue = sign(dotProduct(item)         #重新计算估计值
```

8.面向过程编程实现（更新权重函数，净输入函数，15分钟）

```
01 weight =[0,0,0]                                          #初始化权重
02 rate = 0.1                                               #学习率
03 def upWeight(item,estimate):                             #更新权重
04     weight[0] +=  rate*(item[2]-estimate)                 #更新偏置项
05     for i in range(2):
06         weight[i+1] += rate*(item[2]-estimate)*item[i]    #更新权重

07 def dotProduct(item):                                    #计算点积
08     sum = weight[0]*1
09     for i in range(2):
10         sum += item[i]*weight[i+1]
11     return sum
```

9.面向过程编程实现（定义test函数，5分钟）

具体实践过程（面向过程）

```
01 def test():
02     error = 0
03     for item in test_data:          #用测试集集数据进行测试
04         if sign(dotProduct(item)) != item[2]:
05             error +=1
06     print("错误数：",error)
07     print(weight)                   #查看权重
```

预测：print(sign(dotProduct([2.2,3.3])))

1.面向对象编程实现（定义对象，10分钟）

具体实践过程（面向对象）

```
class CrDataSet():
    def __init__(self,datafile):        #初始化
        pass
    def crDataSet(self,datafile):       #构造数据集
        pass
```

```
class Perceptron():
    def __init__(self,rate,weight, ,train,test):    #初始化
        pass
    def dotProduct(self,item):                       #计算点积
        pass
    def upWeight(self,item,estimate):                #更新权重
        pass
    def sign(self,value):                            #激励函数
        pass
    def train(self):                                 #训练模型
        pass
    def test(self):                                  #测试模型
        pass
    def forecast(self,features):                     #预测
        pass
```

面向对象程序设计思想——以对象为中心”

类的设计原则

类的设计原则简记：SOLID

S：单一职责原则；O：开闭原则；L：里氏替换原则；

I：接口隔离原则；D：依赖倒置原则

S：**单一职责原则** 要求一个类的功能尽量单一，类的方法也要尽量单一。核心思想是：高内聚，低耦合。也即类、方法尽量具有原子性。这个原则只使用于基础类，不适用于基于基础类构建复杂的聚合类。在构成聚合类时优先使用类组合，而不使用类继承。

O：**开闭原则** 核心思想是对扩展开放，对修改关闭。换句话说，提供者（生产者）能扩展功能，使用者（消费者）不需要修改代码。在实现中，类和类之间是通过“接口”交互的，如果提供者的接口不变（方法名、参数、返回值）则可以应用开闭原则，如果接口改变（方法名、参数、返回值、新增方法）则不适应开闭原则。

L：**里氏替换原则** 子类必须能替换成他们的父类，即当程序基于父类实现时，如果用子类去替换父类而程序不需要修改。里氏替换原则是针对类的继承来讨论的，在应用这个原则实现子类时必须：①子类必须实现或者继承父类所有的共有方法；②子类每个方法的输入参数必须和父类一样；③子类每个方法的输出必须不比父类少。

参考资料（类的设计原则）

I：**接口隔离原则** 客户端不应该被强迫去依赖那些它不需要的接口。这个原则是针对接口的设计提出的，主要是指在设计接口时不要将一些功能不相关的方法做在一个接口中，将功能不相关的方法放在另外的接口实现隔离。在使用接口时要注意控制接口的粒度，接口定义的粒度不能太细，也不能太粗。 接口粒度太细，系统中就会出现接口泛滥，接口和实现类急剧膨胀，反而不易维护；接口粒度太粗，就会违背ISP，系统的灵活性就会降低，不易维护和扩展。

D：**依赖倒置原则** 传统的依赖关系是：高层模块依赖低层模块，依赖倒置原则是：高层的模块不依赖低层模块，高层模块和低层模块都依赖抽象。变化是在原来的高层模块与低层模块之间增加了一个抽象接口。抽象不依赖具体，具体依赖抽象。其目的是解耦。依赖倒置原则的核心是要面向接口编程，理解了面向接口编程，也就理解了依赖倒置。

2.面向对象编程实现（执行逻辑，5分钟）

具体实践过程（面向对象）

```
if __name__ =="__main__":
    train = CrDataSet('train.data')                          #构建训练集
    test  = CrDataSet('test.data')                           #构建测试集
    perceptron = Perceptron(0.1,[0,0,0],train,test)          #实例化感知器
    perceptron.train()                                       #训练模型
    perceptron.test()                                        #测试模型
    perceptron.forecast([5.3,2.5])                           #使用模型预测
```

3.面向对象编程实现（完善CrDataSet类，30分钟）

具体实践过程（面向对象）

```
class CrDataSet():
    def __init__(self,datafile):                    #类似于c++中的构造函数
        self.data = []                               #属性：data
        self.crDataSet(datafile)
    def crDataSet(self,datafile):                    #方法：crDataSet
        with open(datafile,'r') as file:
            data = file.read().splitlines()
            for line in data:
                temp = []
                items = line.split(',')
                temp.append(float(items[0]))
                temp.append(float(items[2]))
                if items[4] =='Iris-setosa':
                    temp.append(1)
                else:
                    temp.append(-1)
                self.data.append(temp)
```

从面向过程拷贝过来，稍作修改即可。重点是需要理解self的意义

4.面向对象编程实现（完善Perceptron类，5分钟）

具体实践过程（面向对象）

```
def __init__(self,rate,weight,train,test):
        self.rate = rate                          #属性：学习速率
        self.weight = weight                      #属性：权重
        self.trainData  = train                   #属性：训练集
        self.testData  = test                     #属性：测试集


def dotProduct(self,item):
        """计算点积"""                    #这段注释文字在一些高级的编辑中会以功能提示的形式出现
        sum = self.weight[0]*1
        for i in range(2):
            sum += item[i]*self.weight[i+1]
        return sum
```

4.面向对象编程实现（完善Perceptron类，30分钟）

具体实践过程（面向对象）

```
def upWeight(self,item,estimate):
        """更新权重"""
        self.weight[0] +=  self.rate*(item[2]-estimate)
        for i in range(2):
          self.weight[i+1] +=  self.rate*(item[2]-estimate)*item[i]
def train(self):
      """训练模型"""
       error =1
       while error:
            error = 0
            for item in  self.trainData.data:
              outvalue = self.sign(self.dotProduct(item))
              while outvalue !=item[2]:
                 error = 1
                 self.upWeight(item,outvalue)
                 outvalue = self.sign(self.dotProduct(item))
```

4.面向对象编程实现（完善Perceptron类，30分钟）

具体实践过程（面向对象）

```
def sign(self,value):
        """激励函数"""
        if value>0:
           return 1
        elif value<0:
           return -1
        else:
           return 0
def test(self):
        """测试模型"""
        error = 0
        for item in self.testData.data:
            if self.sign(self.dotProduct(item)) != item[2]:
                error +=1
        print("错误数：",error)
        print(self.weight)
def forecast(self,features):
        """预测"""
        print(self.sign(self.dotProduct(features)))
```

1.使用第三方库编程实现（安装库，5分钟）

具体实践过程（使用第三方库）

在DOS窗口中依次执行

```
pip3 install  numpy                 #提供一个N维数组类型ndarray
pip3 install  pandas                #基于numpy的数据分析库
pip3 install   scipy                #基于numpy的科学计算库
pip3 install -U scikit-learn        #通用机器学习库
```

2.使用第三方库编程实现（20分钟）

具体实践过程（使用第三方库）

```
import numpy as np
import pandas as pd
from sklearn.linear_model import Perceptron
from sklearn.externals import joblib
import matplotlib.pyplot as plt

def crData():
    """构建训练集和测试集"""
    global train_features,train_target,test_features,test_target
    train_data = pd.read_csv('train.data',header=None)
    train_features = train_data[[0,2]].values       #提取特征数据
    train_target=train_data[4].replace(['Iris-setosa','Iris-versicolor'],[1,-1])
    test_data = pd.read_csv('test.data',header=None) #装载验证集
    test_features = test_data[[0,2]].values
    test_target=test_data[4].replace(['Iris-setosa','Iris-versicolor'],[1,-1])
```

具体实践过程（使用第三方库）

```
def fit():
    """训练模型"""
    global model
    model = Perceptron(max_iter=20)
    model.fit(train_features,train_target)      #训练模型
    joblib.dump(model,'save/model.pkl')      #保存模型

def predict():
    """使用模型预测"""
    m =joblib.load('save/model.pkl')
    print(m.predict(np.array([[5.0,1.5]])))
```

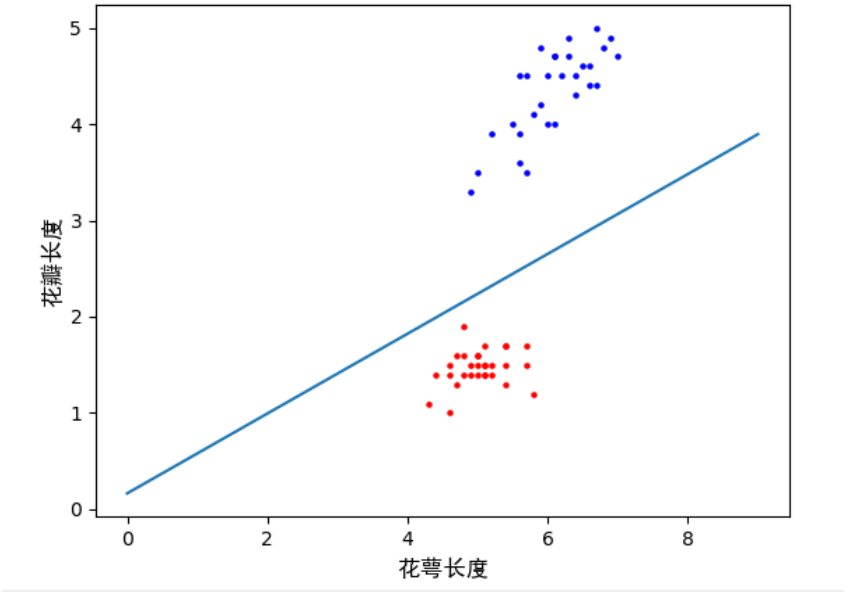
具体实践过程（使用第三方库）

```
def dispImg():
    """画图像"""
    plt.xlabel('花萼长度',fontproperties = 'SimHei',fontsize = 12)
    plt.ylabel('花瓣长度',fontproperties = 'SimHei',fontsize = 12)
    positive_x1 = [train_features[i][0] for i in range(60) if train_target[i] == 1]
    positive_x2 = [train_features[i][1] for i in range(60) if train_target[i] == 1]
    negetive_x1 = [train_features[i][0] for i in range(60) if train_target[i] == -1]
    negetive_x2 = [train_features[i][1] for i in range(60) if train_target[i] == -1]

    plt.scatter(positive_x1,positive_x2,s=5,c='red')
    plt.scatter(negetive_x1,negetive_x2,s=5,c='blue')
    line_x = np.arange(0,10)
    line_y = (-model.intercept_-model.coef_[0][0] *line_x )/ model.coef_[0][1]
    plt.plot(line_x,line_y)
    plt.show()
```

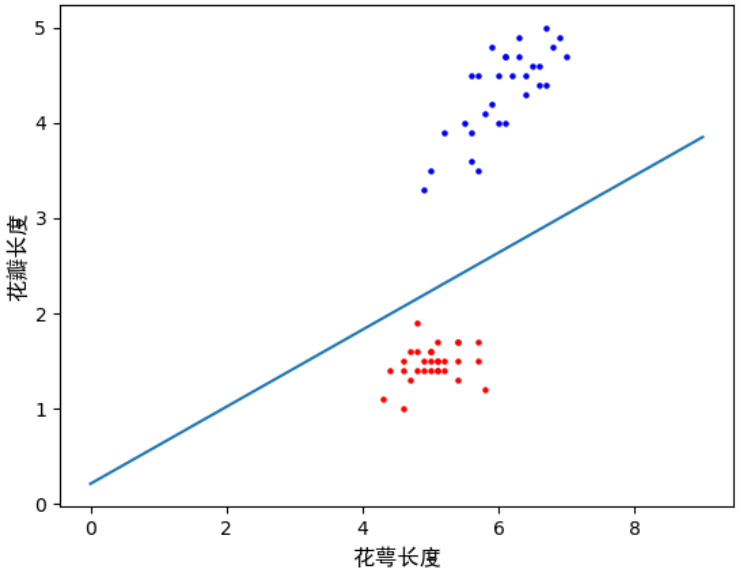
具体实践过程（使用第三方库）

```
if __name__ == "__main__":
    train_features = None
    train_target = None
    test_features = None
    test_target = None
    model = None
    crData()
    fit()

    print("模型评价:",model.score(test_features,test_target))     #评估模型
    print(model.predict(np.array([[3,10]])))                      #使用模型预测
    print(model.coef_)
    print(model.get_params())
    dispImg()
```

自定义算法 PK sklearn



自定义算法 拟合直线



sklearn 拟合直线

现实中的数据集大都不是线性可分的，所以感知器算法用得并不多，为了能够对线性不可分的数据进行分类，在感知器算法的基础上，产生了支持向量机和神经网络（两种算法，在现实中应用很多）。

附录：sklearn库

sklearn 库简介

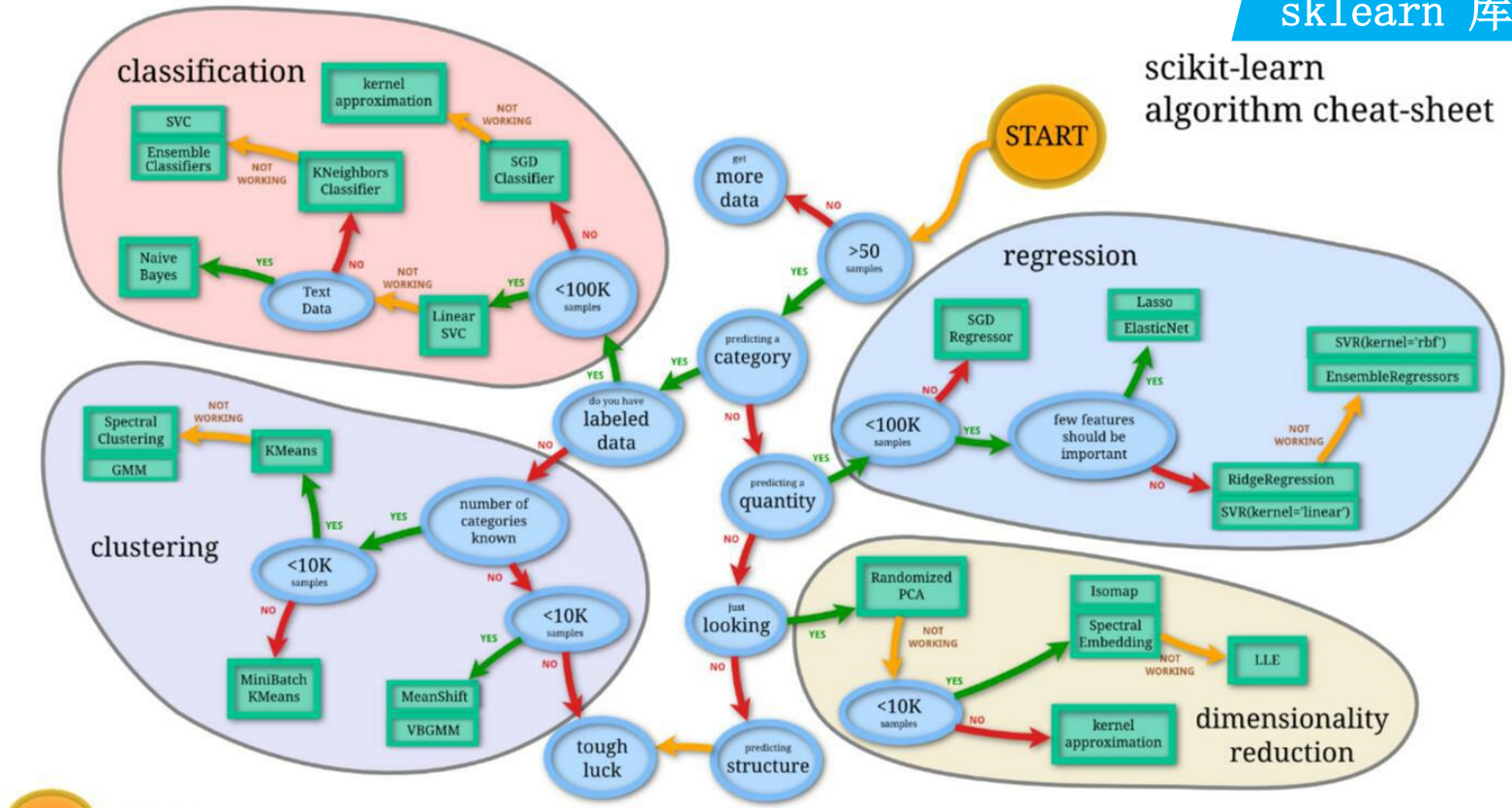
sklearn是机器学习中一个常用的python第三方模块，是scikit-learn的简称，里面对一些常用的机器学习方法进行了封装，在进行机器学习任务时，并不需要每个人都实现所有的算法，只需要简单的调用sklearn里的模块就可以实现大多数机器学习任务。

sklearn库是在NumPy、Scipy和matplotlib的基础上开发而成的。因此，在使用sklearn之前需要先安装NumPy、Scipy和matplotlib模块。

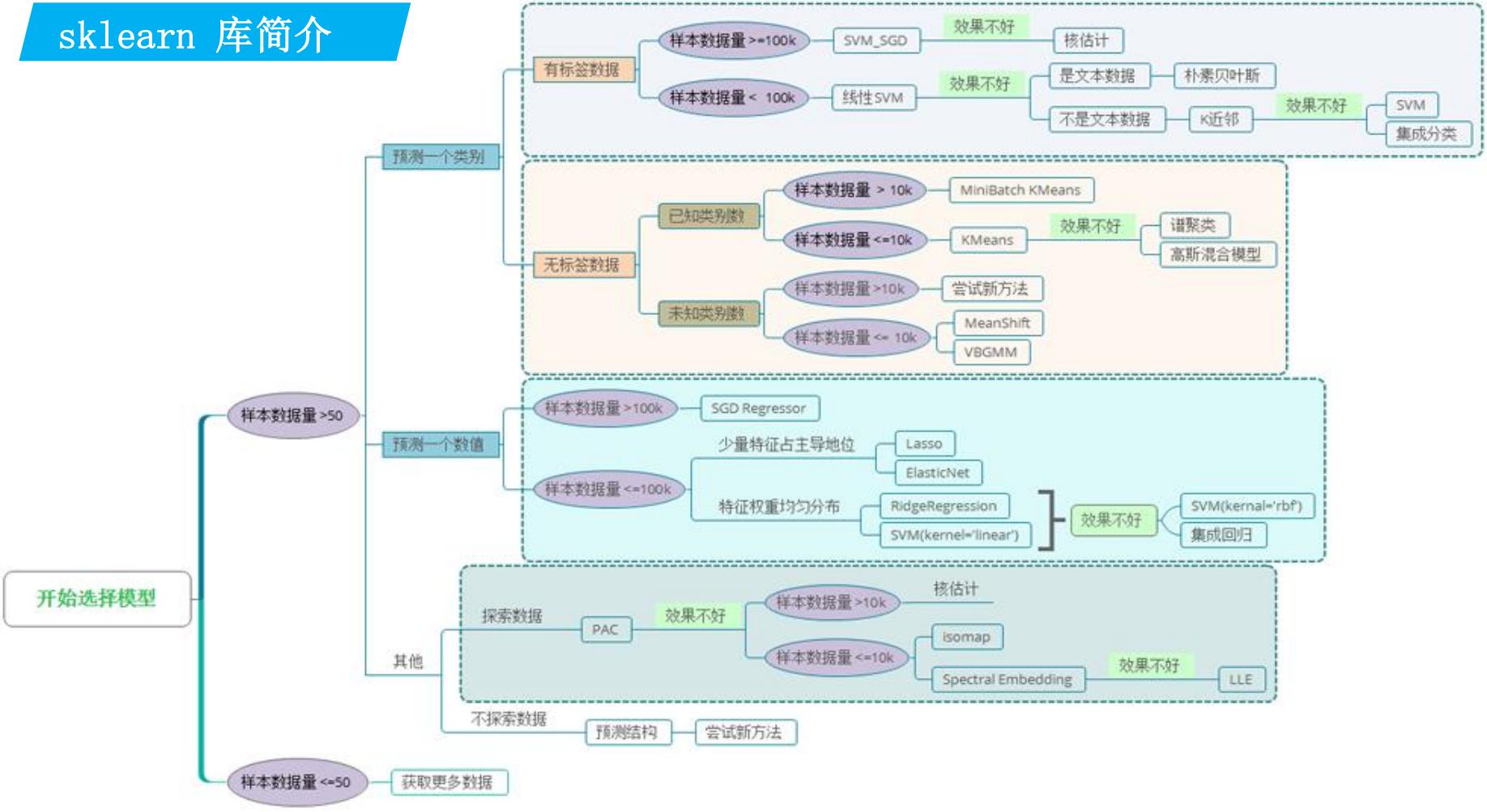
sklearn库的功能分为6大部分：分类、回归、聚类、降维、模型选择、数据预处理。

传统机器学习任务从开始到建模的一般流程：

获取数据 -> 数据预处理 -> 训练建模 -> 模型评估 -> 保存模型->应用



sklearn 库简介


开始选择模型
样本数据量>50
预测一个类别
有标签数据
样本数据量>=100k
SVM_SGD
效果不好
核估计
样本数据量< 100k
线性SVM
效果不好
是文本数据
朴素贝叶斯
不是文本数据
K近邻
效果不好
SVM
集成分类
无标签数据
已知类别数
样本数据量 > 10k
MiniBatch KMeans
样本数据量<=10k
KMeans
效果不好
谱聚类
高斯混合模型
未知类别数
样本数据量>10k
尝试新方法
样本数据量<= 10k
MeanShift
VBGMM
预测一个数值
样本数据量>100k
SGD Regressor
样本数据量<=100k
少量特征占主导地位
Lasso
ElasticNet
特征权重均匀分布
RidgeRegression
SVM(kernel='linear')
效果不好
SVM(kernal='rbf')
集成回归
其他
探索数据
PAC
效果不好
样本数据量>10k
核估计
样本数据量<=10k
isomap
Spectral Embedding
效果不好
LLE
不探索数据
预测结构
尝试新方法
样本数据量<=50
获取更多数据

分类：根据特征数据判断样本实例所属类别。包括二分类和多分类。属于有监督学习

分类的例子：根据人的某些生理数据判断是否健康（二分类）；预测明天的天气状况是晴、阴、多云等（多分类）

常用分类模型：

模型名称	加载模块
最近邻算法	neighbors.NearestNeighbors
支持向量机	svm.SVC
朴素贝叶斯	naive_bayes.GaussianNB
决策树	tree.DecisionTreeClassifier
集成方法	ensemble.BaggingClassifier
神经网络	neural_network.MLPClassifier

回归：根据特征数据预测样本的数量输出。可以分为线性回归和非线性回归。属于有监督学习

回归的例子：预测明天的温度

常用回归模型：

模型名称	加载模块
岭回归	linear_model.Ridge
Lasso回归	linear_model.Lasso
弹性网络	linear_model.ElasticNet
最小角回归	linear_model.Lars
贝叶斯回归	linear_model.BayesianRidge
逻辑回归	linear_model.LogisticRegression
多项式回归	preprocessing.PolynomialFeatures

聚类：把样本根据特征数据进行分类，不用于预测，属于无监督学习。

聚类的例子：根据考试成绩对学生分类

常用聚类模型：

模型名称	加载模块
K-means	cluster.KMeans
AP聚类	cluster.AffinityPropagation
均值漂移	cluster.MeanShift
层次聚类	cluster.AgglomerativeClustering
DBSCAN	cluster.DBSCAN
BIRCH	cluster.Birch
谱聚类	cluster.SpectralClustering

降维：将高维数据转化为低维数据，以达到计算简单，减小计算时间的目的

常用降维模型：

模型名称	加载模块
主成分分析	decomposition.PCA
截断SVD和LSA	decomposition.TruncatedSVD
字典学习	decomposition.SparseCoder
因子分析	decomposition.FactorAnalysis
独立成分分析	decomposition.FastICA
非负矩阵分解	decompositon.NMF
LDA	decompositon.LatentDirichletAllocation

sklearn模型的固定使用方法：

（1）实例化算法对象：如model = Perceptron(模型参数)
（2）拟合模型：如model.fit(X,[Y]),无监督学习时省略Y
（3）评价模型：如model.score(X, y)
（4）模型预测：如model.predict(X)