



华中师范大学

自然语言处理

第六章 文本分类

主讲教师：曾江峰

华中师范大学 信息管理学院

jfzeng@ccnu.edu.cn

语料清洗

- ❖ 本章首先回顾了文本分类的历史，其次重点介绍了朴素贝叶斯和支持向量机两种机器学习方法。
- ❖ 关于朴素贝叶斯，介绍了贝叶斯定理、三种贝叶斯分类方法，使用朴素贝叶斯进行新闻分类、垃圾邮件的文本分类。
- ❖ 关于支持向量机，介绍支持向量机的原理，了解三种核函数——线性核函数、多项式核函数和高斯核函数。使用支持向量机对鸢尾花的分类。



语料清洗

- 1 历史回顾
- 2 文本分类方法
- 3 朴素贝叶斯
- 4 朴素贝叶斯进行新闻分类
- 5 朴素贝叶斯进行垃圾邮件分类
- 6 支持向量机
- 7 支持向量机进行鸢尾花分类

1. 历史回顾

文本分类是指根据文本内容自动确定文本类别的过程。文本分类的研究可以追溯到上世纪六十年代，早期的文本分类主要是基于知识工程（Knowledge Engineering），通过手工定义规则对文本进行分类，这种方法费时费力，而且必须对某一领域有足够的了解，才能写出合适的规则。



1. 历史回顾

到了上世纪九十年代，机器学习应用到文本分类。1971年，Rocchio通过用户反馈修正类权重向量，构成简单的线性分类器。1979年，van Rijsbergen将准确率、召回率等相关概念引入文本分类。1992年，Lewis在其论文Representation and Learning in Information Retrieval中系统地介绍了文本分类系统实现的各个细节，并且在自己建立的数据集Reuters21578上进行了测试，这篇博士论文成为文本分类的经典之作。



1. 历史回顾

其后，Yiming Yang对各种特征选择的方法，如信息增益、互信息、统计量等，进行比较研究。1995年，Vipnik基于统计理论提出了支持向量机(Support Vector Machine)方法。Thorsten Joachims第一次将线性核函数的支持向量机用于文本分类，取得显著效果。



2. 文本分类方法

- 朴素贝叶斯
- 支持向量机



3. 朴素贝叶斯

朴素贝叶斯模型，或朴素贝叶斯分类器(Naive Bayes Classifier, NBC) 发源于古典数学理论，是基于贝叶斯理论与特征条件独立假设的分类方法、通过单独考量每一特征被分类的条件概率，做出分类预测。



3. 朴素贝叶斯

贝叶斯算法具有如下优点：

- (1) 对待预测样本进行预测，简单高效。
- (2) 对于多分类问题同样有效。
- (3) 在分布独立假设成立的情况下，所需样本量较少，效果好于逻辑回归。
- (4) 对于类别变量，效果非常好。



3. 朴素贝叶斯

贝叶斯算法具有如下缺点：

- (1) 朴素贝叶斯有分布独立的假设前提，而现实生活中很难是完全独立。
- (2) 对输入数据的数据类型较为敏感。



3. 朴素贝叶斯

3.1 贝叶斯定理

条件概率(conditional probability)又称后验概率， $P(A|B)$ 是指事件 A 在另一个事件 B 已经发生条件下的发生概率，读作“在 B 条件下 A 的概率”，条件概率公式如下所示：

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

其中， $P(A \cap B)$ 为事件 A 和 B 的联合概率，表示两个事件共同发生的概率。 A 与 B 的联合概率也可以表示为 $P(A, B)$



3. 朴素贝叶斯

3.1 贝叶斯定理

$$P(A \cap B) = P(A|B)P(B)$$

$$P(A \cap B) = P(B|A)P(A)$$

因此，有

$$P(A|B)P(B) = P(B|A)P(A)$$

从而，得到贝叶斯公式

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$



3. 朴素贝叶斯

3.1 贝叶斯定理

【例1】现有 x 、 y 两个容器，容器 x 有7个红球和3个白球，容器 y 有1个红球和9个白球。现从两个容器里任取一个红球，问红球来自容器 x 的概率是多少？

设取出红球为事件 B ，选中容器 x 为事件 A ，则所求为 $P(A|B)$

$$P(B) = \frac{7 + 1}{7 + 3 + 1 + 9} = \frac{2}{5}, \quad P(A) = \frac{1}{2}, \quad P(B|A) = \frac{P(A \cap B)}{P(A)} = \frac{\frac{7}{7+3+1+9}}{\frac{1}{2}} = \frac{7}{10}$$

$$\dots\dots\dots P(B|A)P(A) \dots\dots\dots$$



3. 朴素贝叶斯

3.2 朴素贝叶斯

朴素贝叶斯方法是在贝叶斯算法的基础上进行了相应的简化，即假定给定目标值时属性之间相互条件独立。虽然这个简化方式在一定程度上降低了贝叶斯分类算法的分类效果，但是在实际的应用场景中，极大地简化了贝叶斯方法的复杂性。



3. 朴素贝叶斯

3.2 朴素贝叶斯

`sklearn.naive_bayes`模块具有3种贝叶斯分类方法，分别是GaussianNB、MultinomialNB和BernoulliNB。其中，GaussianNB是高斯分布的朴素贝叶斯。MultinomialNB是多项式分布的朴素贝叶斯。BernoulliNB是伯努利分布的朴素贝叶斯。

- (1) GaussianNB适合样本特征是正态分布的数值情况。
- (2) MultinomialNB适合非负离散数值特征的分类情况。
- (3) BernoulliNB适合二元离散值或者很稀疏的多元离散值情况。



3. 朴素贝叶斯

3.3 GaussianNB

Sklearn提供GaussianNB用于高斯分布，适合样本特征分布是连续值

`GaussianNB(priors=True)`

GaussianNB类的主要参数只有一个，即先验概率priors。



3. 朴素贝叶斯

3.3 GaussianNB

【例2】 GaussianNB举例

```
# 例2: GaussianNB举例
import numpy as np
from sklearn.datasets import make_blobs
from sklearn.naive_bayes import GaussianNB
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
X, y = make_blobs(n_samples = 500, centers = 5, random_state = 8)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 8)

gnb = GaussianNB()
gnb.fit(X_train, y_train)
print('模型得分: {:.3f}'.format(gnb.score(X_test, y_test)))

x_min, x_max = X[:,0].min() - 0.5, X[:,0].max() + 0.5
y_min, y_max = X[:,1].min() - 0.5, X[:,1].max() + 0.5
xx, yy = np.meshgrid(np.arange(x_min, x_max, .02), np.arange(y_min, y_max, .02))
z = gnb.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)
```



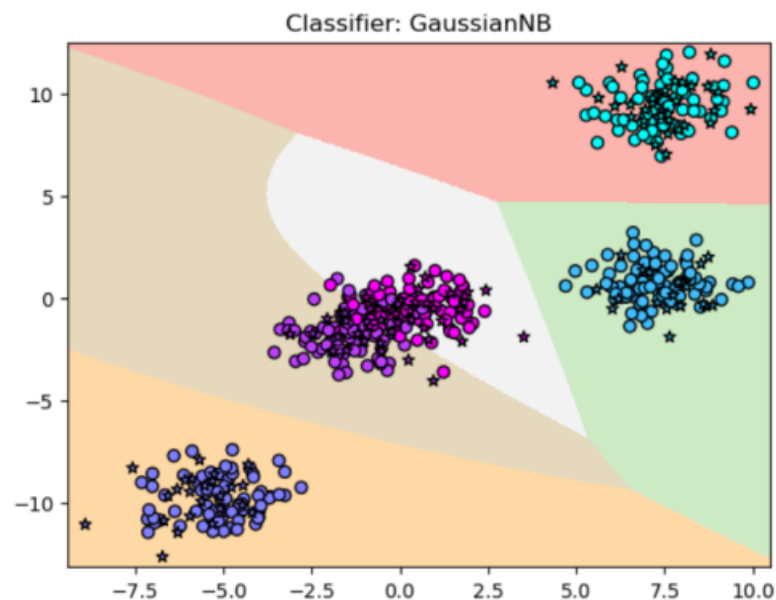
3. 朴素贝叶斯

3.3 GaussianNB

【例2】 GaussianNB举例

```
plt.pcolormesh(xx,yy,z,cmap = plt.cm.Pastel1)
plt.scatter(X_train[:,0],X_train[:,1],c = y_train,cmap = plt.cm.cool,edgecolor = 'k')
plt.scatter(X_test[:,0],X_test[:,1],c = y_test,cmap = plt.cm.cool,marker = '*', edgecolor = 'k')
plt.xlim(xx.min(),xx.max())
plt.ylim(yy.min(),yy.max())
plt.title('Classifier: GaussianNB')
plt.show()
```

模型得分: 0.968



3. 朴素贝叶斯

3.4 MultinomialNB

多项式朴素贝叶斯假设特征由一个简单多项式分布生成，非常适用于描述出现次数或者出现次数比例的特征，例如文本分类，其特征都是指待分类文本的单词出现次数或者频次。



3. 朴素贝叶斯

3.4 MultinomialNB

Sklearn提供MultinomialNB用于多项式分布

`MultinomialNB(alpha=1.0, fit_prior=True, class_prior=None)`

`alpha`: 先验平滑因子，默认等于1，当等于1时表示拉普拉斯平滑。

`fit_prior`: 是否去学习类的先验概率，默认是True。

`class_prior`: 各个类别的先验概率，如果没有指定，模型根据数据自动学习， 每个类别的先验概率相同，为类标记总个数N分之一。



3. 朴素贝叶斯

3.4 MultinomialNB

【例3】MultinomialNB举例

```
# 例3: MultinomialNB举例
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets, naive_bayes
from sklearn.model_selection import train_test_split

def load_data():
    digits = datasets.load_digits() # 加载scikit-learn自带的digits数据集
    return train_test_split(digits.data, digits.target, test_size = 0.25, random_state = 0, stratify = digits.target)

# 多项式贝叶斯分类器MultinomialNB模型
def test_MultinomialNB(*data):
    X_train, X_test, y_train, y_test = data
    cls = naive_bayes.MultinomialNB()
    cls.fit(X_train, y_train)
    print('Training Score: %.2f' % cls.score(X_train, y_train))
    print('Testing Score: %.2f' % cls.score(X_test, y_test))
```



3. 朴素贝叶斯

3.4 MultinomialNB

【例3】MultinomialNB举例

```
# 产生用于分类问题的数据集
X_train,X_test,y_train,y_test = load_data()

# 调用 test_GaussianNB
test_MultinomialNB(X_train,X_test,y_train,y_test)

# 测试 MultinomialNB 的预测性能随alpha参数的影响
def test_MultinomialNB_alpha(*data):
    X_train,X_test,y_train,y_test = data
    alphas = np.logspace(-2,5,num = 200)
    train_scores = []
    test_scores = []
    for alpha in alphas:
        cls = naive_bayes.MultinomialNB(alpha=alpha)
        cls.fit(X_train,y_train)
        train_scores.append(cls.score(X_train,y_train))
        test_scores.append(cls.score(X_test, y_test))
```

```
# 绘图
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.plot(alphas,train_scores,label = "Training Score")
ax.plot(alphas,test_scores,label = "Testing Score")
ax.set_xlabel(r"$\alpha$")
ax.set_ylabel("score")
ax.set_ylim(0,1.0)
ax.set_title("MultinomialNB")
ax.set_xscale("log")
plt.show()

# 调用 test_MultinomialNB_alpha
test_MultinomialNB_alpha(X_train,X_test,y_train,y_test)
```



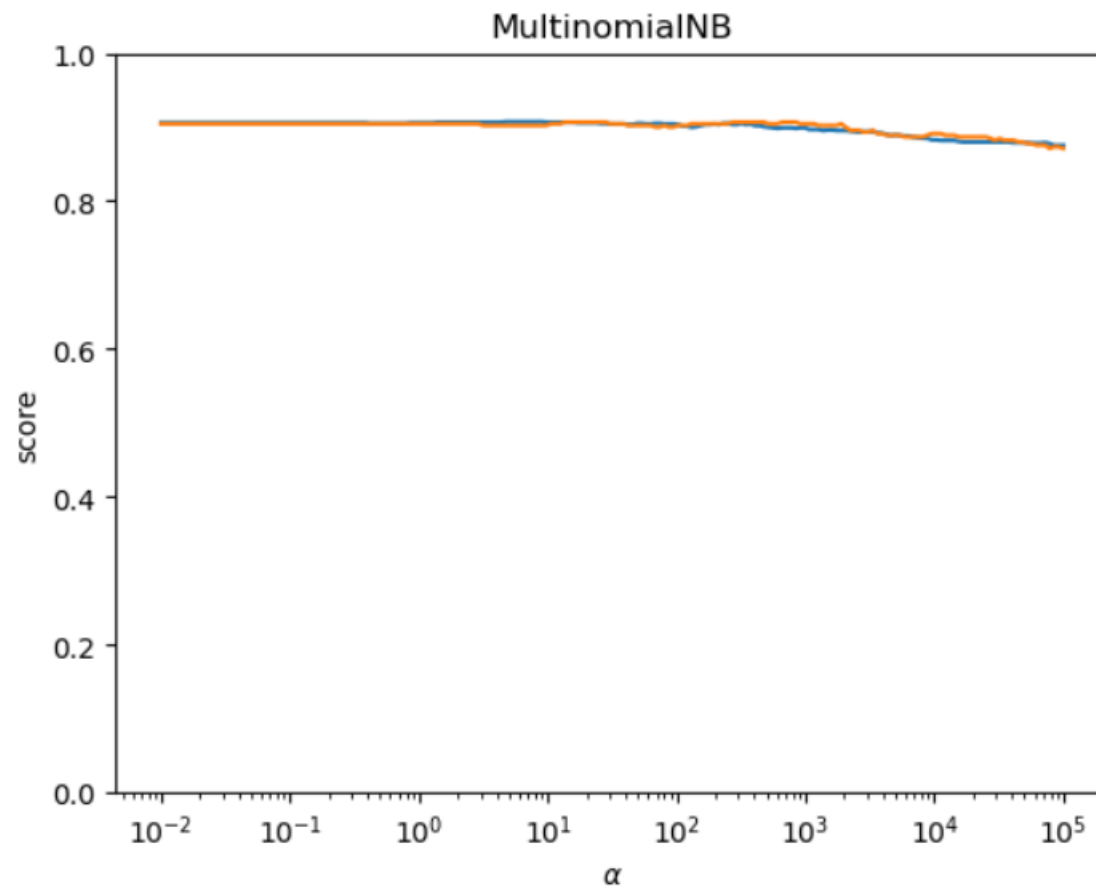
3. 朴素贝叶斯

3.4 MultinomialNB

【例3】MultinomialNB举例

Training Score: 0.91

Testing Score: 0.90



3. 朴素贝叶斯

3.5 BernoulliNB

伯努利分布又名“两点分布”、“二项分布”或“0-1分布”，适用于数据集中每个特征只有0和1两个数值。



3. 朴素贝叶斯

3.5 BernoulliNB

Sklearn提供BernoulliNB用于伯努利分布

`BernoulliNB(alpha=1.0, binarize=0.0, fit_prior=True, class_prior=None)`

`alpha`: 平滑因子, 与多项式中的`alpha`一致。

`fit_prior` : 是否去学习类的先验概率, 默认是`True`。

`class_prior` : 各个类别的先验概率, 与多项式中的`class_prior`一致。

`binarize` : 样本特征二值化的阈值, 默认是0。如果不输入, 模型认为所有特征都已经二值化; 如果输入具体的值, 模型把大于该值的归为一类, 小于的归为另一类。



3. 朴素贝叶斯

3.5 BernoulliNB

【例4】 BernoulliNB举例

```
# 例4: BernoulliNB举例
import numpy as np
from sklearn.naive_bayes import BernoulliNB
from sklearn.datasets import make_blobs
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

X, y = make_blobs(n_samples = 500, centers = 5, random_state = 8)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 8)
nb = BernoulliNB()
nb.fit(X_train, y_train)
print('模型得分: {:.3f}'.format(nb.score(X_test, y_test)))

x_min, x_max = X[:,0].min()-0.5, X[:,0].max() + 0.5
y_min, y_max = X[:,1].min()-0.5, X[:,1].max() + 0.5
xx,yy = np.meshgrid(np.arange(x_min, x_max,.02),np.arange(y_min, y_max, .02))
```



3. 朴素贝叶斯

3.5 BernoulliNB

【例4】 BernoulliNB举例

```
z = nb.predict(np.c_[xx.ravel(),yy.ravel()]).reshape(xx.shape)
plt.pcolormesh(xx,yy,z,cmap = plt.cm.Pastel1)
plt.scatter(X_train[:,0],X_train[:,1],c = y_train,cmap = plt.cm.cool,edgecolor = 'k')
plt.scatter(X_test[:,0],X_test[:,1],c = y_test,cmap = plt.cm.cool,marker = '*',edgecolor = 'k')
plt.xlim(xx.min(),xx.max())
plt.ylim(yy.min(),yy.max())
plt.title('Classifier: BernoulliNB')
plt.show()
```

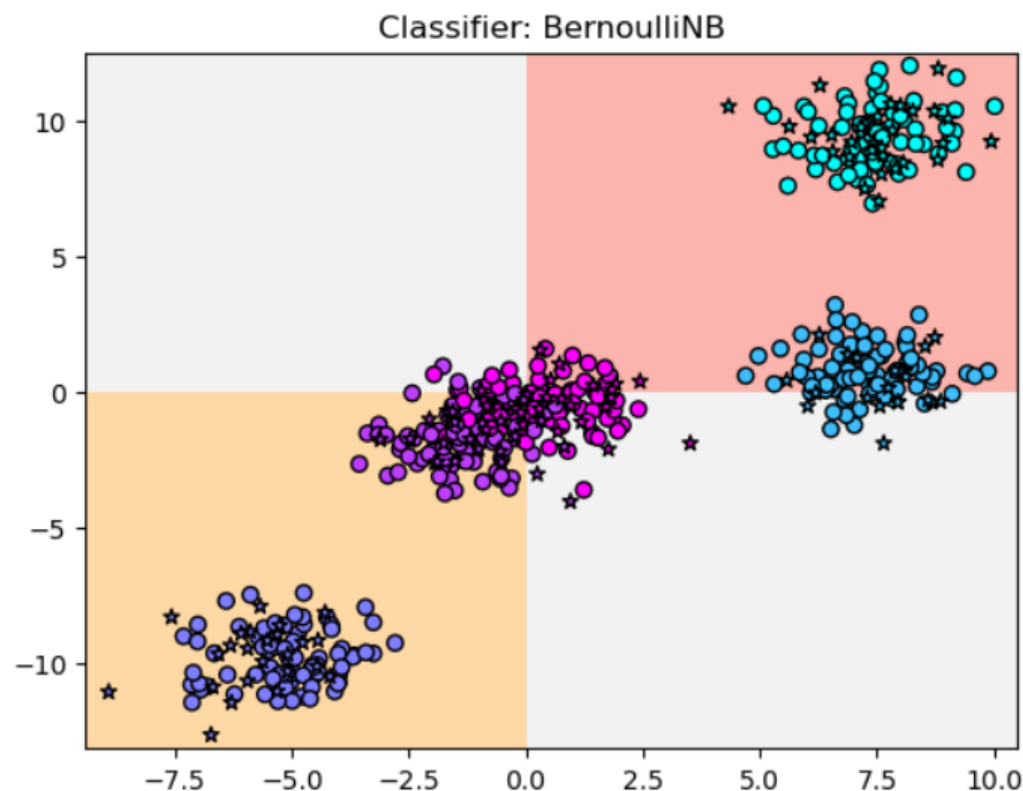


3. 朴素贝叶斯

3.5 BernoulliNB

【例4】BernoulliNB举例

模型得分: 0.544



4. 朴素贝叶斯进行新闻分类

新闻分类数据来源于Sklearn的20newsgroups数据集，属于非负离散数值，适合用于多项式离散分布朴素贝叶斯进行分类。

【例5】MultinomialNB应用于20newsgroups数据集

```
# 例5: MultinomialNB应用于20newsgroups数据集
from sklearn.datasets import fetch_20newsgroups
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB # 多项式朴素贝叶斯模型
from sklearn.metrics import classification_report

# 步骤1. 数据获取
news = fetch_20newsgroups(subset = 'all')
print('输出数据的条数:', len(news.data)) # 输出数据的条数: 18846
```



4. 朴素贝叶斯进行新闻分类

【例5】MultinomialNB应用于20newsgroups数据集

```
#步骤2. 数据预处理
#分割训练集和测试集, 随机采样25%的数据样本作为测试集
X_train,X_test,y_train,y_test = train_test_split(news.data,news.target,test_size = 0.25,random_state = 33)
#文本特征向量化
vec = CountVectorizer()
X_train = vec.fit_transform(X_train)
X_test = vec.transform(X_test)

#步骤3. 使用多项式朴素贝叶斯进行训练
mnb = MultinomialNB()
mnb.fit(X_train,y_train) # 利用训练数据对模型参数进行估计
y_predict = mnb.predict(X_test) # 对参数进行预测

#步骤4. 获取结果报告
print('准确率:', mnb.score(X_test,y_test))
print(classification_report(y_test, y_predict, target_names = news.target_names))
```



4. 朴素贝叶斯进行新闻分类

【例5】MultinomialNB应用于20newsgroups数据集

输出数据的条数: 18846

准确率: 0.8397707979626485

	precision	recall	f1-score	support
alt.atheism	0.86	0.86	0.86	201
comp.graphics	0.59	0.86	0.70	250
comp.os.ms-windows.misc	0.89	0.10	0.17	248
comp.sys.ibm.pc.hardware	0.60	0.88	0.72	240
comp.sys.mac.hardware	0.93	0.78	0.85	242
comp.windows.x	0.82	0.84	0.83	263
misc.forsale	0.91	0.70	0.79	257
rec.autos	0.89	0.89	0.89	238
rec.motorcycles	0.98	0.92	0.95	276
rec.sport.baseball	0.98	0.91	0.95	251
rec.sport.hockey	0.93	0.99	0.96	233
sci.crypt	0.86	0.98	0.91	238
sci.electronics	0.85	0.88	0.86	249
sci.med	0.92	0.94	0.93	245
sci.space	0.89	0.96	0.92	221
soc.religion.christian	0.78	0.96	0.86	232
talk.politics.guns	0.88	0.96	0.92	251
talk.politics.mideast	0.90	0.98	0.94	231
talk.politics.misc	0.79	0.89	0.84	188
talk.religion.misc	0.93	0.44	0.60	158
accuracy			0.84	4712
macro avg	0.86	0.84	0.82	4712
weighted avg	0.86	0.84	0.82	4712



4. 朴素贝叶斯进行新闻分类

新闻分类数据来源于Sklearn的20newsgroups数据集，属于非负离散数值，适合用于多项式离散分布朴素贝叶斯进行分类。

【例5】 MultinomialNB应用于20newsgroups数据集



4. 朴素贝叶斯进行垃圾邮件分类

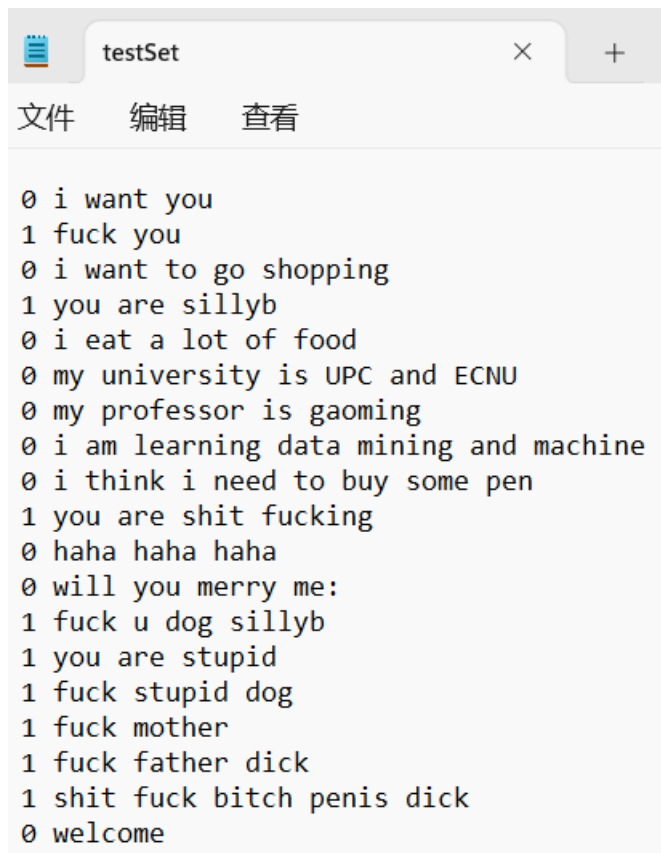
朴素贝叶斯算法在垃圾邮件过滤中，应用及其广泛，下面采用朴素贝叶斯定理和Sklearn的函数两种方式实现



5. 朴素贝叶斯进行垃圾邮件分类

5.1 朴素贝叶斯定理实现

语料textSet.txt的内容如下



```
0 i want you
1 fuck you
0 i want to go shopping
1 you are sillyb
0 i eat a lot of food
0 my university is UPC and ECNU
0 my professor is gaoming
0 i am learning data mining and machine
0 i think i need to buy some pen
1 you are shit fucking
0 haha haha haha
0 will you merry me:
1 fuck u dog sillyb
1 you are stupid
1 fuck stupid dog
1 fuck mother
1 fuck father dick
1 shit fuck bitch penis dick
0 welcome
```



5. 朴素贝叶斯进行垃圾邮件分类

5.1 朴素贝叶斯定理实现

```
# -*- coding: utf-8 -*-
from numpy import *
import matplotlib.pyplot as plt
import time
import math
import re

def loadTrainDataSet(): #读取训练集
    fileIn = open("C:\\Users\\Administrator\\自然语言处理\\testSet.txt")
    postingList = [] # 邮件表, 二维数组
    classVec = []
    i = 0
    for line in fileIn.readlines():
        lineArr = line.strip().split()
        temp = []
        for i in range(len(lineArr)):
            if i == 0:
                classVec.append(int(lineArr[i]))
            else:
                temp.append(lineArr[i])
        postingList.append(temp)
        i = i + 1
    return postingList, classVec
```

```
def createVocabList(dataSet): # 创建词典
    vocabSet = set([]) # 定义list型的集合
    for document in dataSet:
        vocabSet = vocabSet | set(document)
    return list(vocabSet)

def setOfWords2Vec(vocabList, inputSet): # 每一个训练样本的特征向量
    returnVec = [0] * len(vocabList)
    for word in inputSet:
        if word in vocabList:
            returnVec[vocabList.index(word)] = 1
        else:
            pass
    # print("\'%s\' 不存在于词典中"%word)
    return returnVec
```



5. 朴素贝叶斯进行垃圾邮件分类

5.1 朴素贝叶斯定理实现

```
def createTrainMatrix(vocabList,postingList): # 生成训练矩阵
    trainMatrix=[]
    for i in range(len(postingList)):
        curVec=setOfWords2Vec(vocabList,postingList[i])
        trainMatrix.append(curVec)
    return trainMatrix

def trainNB0(trainMatrix,trainCategory):
    numTrainDocs = len(trainMatrix) # 样本数量
    numWords = len(trainMatrix[0]) # 样本特征数
    pAbusive = sum(trainCategory)/float(numTrainDocs) #p(y=1)
    # 分子赋值为1, 分母赋值为2 (拉普拉斯平滑)
    p0Num=ones(numWords); # 初始化向量, 代表所有0类样本中词j出现次数
    p1Num=ones(numWords); # 初始化向量, 代表所有1类样本中词j出现次数
    p0Denom=p1Denom=2.0 # 代表0类1类样本的总词数
    for i in range(numTrainDocs):
        if trainCategory[i] == 1:
            p1Num+=trainMatrix[i]
            p1Denom+=sum(trainMatrix[i])
        else:
            p0Num+=trainMatrix[i]
            p0Denom+=sum(trainMatrix[i])
```

```
p1Vect = p1Num/p1Denom # 概率向量(p(x0=1|y=1),p(x1=1|y=1),...p(xn=1|y=1))
p0Vect = p0Num/p0Denom # 概率向量(p(x0=1|y=0),p(x1=1|y=0),...p(xn=1|y=0))
#取对数, 之后的乘法就可以改为加法, 防止数值下溢损失精度
p1Vect = log(p1Vect)
p0Vect = log(p0Vect)
return p0Vect,p1Vect,pAbusive

def classifyNB(vocabList,testEntry,p0Vec,p1Vec,pClass1): # 朴素贝叶斯分类
    # 先将输入文本处理成特征向量
    regEx = re.compile('\s\w*') #正则匹配分割, 以字母数字的任何字符为分隔符
    testArr = regEx.split(testEntry)
    testVec = array(setOfWords2Vec(vocabList,testArr))

    # 此处的乘法并非矩阵乘法, 而是矩阵相同位置的2个数分别相乘
    # 矩阵乘法应当 dot(A,B) 或者 A.dot(B)
    # 原式子取对数, 因此原本的连乘变为连加
    p1 = sum(testVec * p1Vec) + log(pClass1)
    p0=sum(testVec * p0Vec) + log(1.0 - pClass1)
    # 比较大小
    if p1 > p0:
        return 1
    else:
        return 0
```



5. 朴素贝叶斯进行垃圾邮件分类

5.1 朴素贝叶斯定理实现

```
# 测试方法
def testingNB():
    postingList,classVec = loadTrainDataSet()
    vocabList = createVocabList(postingList)
    trainMatrix = createTrainMatrix(vocabList,postingList)
    p0V,p1V,pAb = trainNB0(trainMatrix,classVec)
    #输入测试文本, 单词必须用空格分开
    testEntry = input()
    print('测试文本为: ' + testEntry)
    if classifyNB(vocabList,testEntry,p0V,p1V,pAb):
        print("-----侮辱性邮件-----")
    else:
        print("-----正常邮件-----")

testingNB()
```

```
fuck
测试文本为:  fuck
-----侮辱性邮件-----

hello
测试文本为:  hello
-----正常邮件-----
```



5. 朴素贝叶斯进行垃圾邮件分类

5.2 Sklearn朴素贝叶斯实现

语料垃圾邮件数据spam.csv百度链接:

https://pan.baidu.com/s/1ncgjQe_FQMIRgL5aSu00Uw

提取码: k9po



5. 朴素贝叶斯进行垃圾邮件分类

5.2 Sklearn朴素贝叶斯实现

步骤1. 读取数据

步骤2. 语料数据划分训练集和测试集

步骤3. 进行无量纲化，使用CountVectorizer将句子转化为向量

步骤4. 采用朴素贝叶斯算法训练预测

步骤5. 模型评估



5. 朴素贝叶斯进行垃圾邮件分类

5.2 Sklearn朴素贝叶斯实现

```
# -*- coding: utf-8 -*-  
from sklearn.feature_extraction.text import CountVectorizer  
from sklearn.model_selection import train_test_split  
import pandas as pd  
  
# 步骤1. 读取数据  
spam_file = r"C:\\Users\\Administrator\\自然语言处理\\spam.csv"  
to_drop = ['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4']  
df = pd.read_csv(spam_file, engine = 'python', encoding = 'ISO-8859-1')  
df.drop(columns = to_drop, inplace = True)  
df['encoded_label'] = df.v1.map({'spam':0, 'ham':1})  
print(df.head())
```



5. 朴素贝叶斯进行垃圾邮件分类

5.2 Sklearn朴素贝叶斯实现

```
# 步骤2. 语料数据划分训练集和测试集
train_data, test_data, train_label, test_label = train_test_split(
    df.v2, df.encoded_label, test_size = 0.7, random_state = 0)
# df.v2是邮件内容, df.v1是邮件标签 (ham和spam)

# 步骤3. 进行无量纲化, 使用CountVectorizer将句子转化为向量
c_v = CountVectorizer(decode_error = 'ignore')
train_data = c_v.fit_transform(train_data)
test_data = c_v.transform(test_data)

# plt.matshow(train_data.toarray())
# plt.show()

# 步骤4. 采用朴素贝叶斯算法训练预测
from sklearn import naive_bayes as nb
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

clf = nb.MultinomialNB()
model = clf.fit(train_data, train_label)
```



5. 朴素贝叶斯进行垃圾邮件分类

5.2 Sklearn朴素贝叶斯实现

```
#步骤5. 模型评估
predicted_label = model.predict(test_data)
print("train score:", clf.score(train_data, train_label))
print("test score:", clf.score(test_data, test_label))
print("Classifier Accuracy:", accuracy_score(test_label, predicted_label))
print("Classifier Report:\n", classification_report(test_label, predicted_label))
print("Confusion Matrix:\n", confusion_matrix(test_label, predicted_label))
```



5. 朴素贝叶斯进行垃圾邮件分类

5.2 Sklearn朴素贝叶斯实现

	v1	v2	encoded_label
0	ham Go until jurong point, crazy.. Available only ...		1
1	ham Ok lar... Joking wif u oni...		1
2	spam Free entry in 2 a wkly comp to win FA Cup fina...		0
3	ham U dun say so early hor... U c already then say...		1
4	ham Nah I don't think he goes to usf, he lives aro...		1

train score: 0.9934171154997008

test score: 0.9794924378364522

Classifier Accuracy: 0.9794924378364522

Classifier Report:

	precision	recall	f1-score	support
0	0.97	0.87	0.92	532
1	0.98	1.00	0.99	3369
accuracy			0.98	3901
macro avg	0.98	0.93	0.95	3901
weighted avg	0.98	0.98	0.98	3901

Confusion Matrix:

```
[[ 464  68]
 [  12 3357]]
```



6. 支持向量机

支持向量机(Support Vector Machine, 缩写 SVM)的基本思想是在N维数据找到N-1维的超平面(hyperplane)作为分类的决策边界。确定超平面的规则是找到离超平面最近的那些点，使它们离分隔超平面的距离尽可能远。离超平面最近的实心圆和空心圆称为支持向量(support vector)，其超平面的距离之和称为“间隔距离”，“间隔距离”越大，分类的准确率越高。

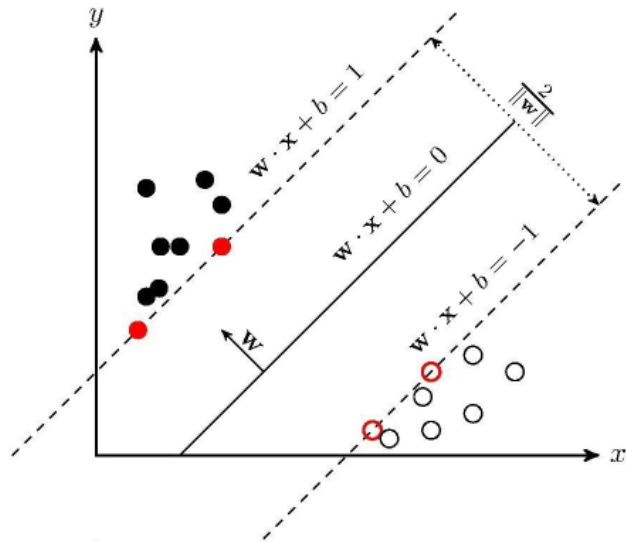


6. 支持向量机

任意超平面都可以用如下的线性方程描述：

$$wx + b = 0$$

超平面的效果如图，两侧的数据好比是两条河，SVM就是将离河岸最远点的集合绘制成线，行走的地方越宽越好，这样掉入河里的概率就低。



6. 支持向量机

6.1 核函数

核函数用于将非线性问题转化为线性问题。通过特征变换增加新的特征，使得低维度空间中的线性不可分问题变为高维度空间中的线性可分问题，从而进行升维变换。



6. 支持向量机

6.1 核函数

支持向量机用于分类(Support Vector Classification, 缩写SVC)的具体语法如下所示:

SVC(kernel)

kernel取值有RBF (Radial Basis Function), Linear, Poly等核函数, 默认取值是"RBF", 即径向基核 (高斯核函数), Linear是线性核函数, Poly是多项式核函数。



6. 支持向量机

6.2 线性核函数

线性核函数（Linear Kernel）是指不通过核函数进行维度提升，仅在原始维度空间中寻求线性分类边界。线性核函数的kernel参数取值为linear。

`SVC(kernel = 'linear', C)`

C: 惩罚系数，用来控制损失函数的惩罚系数。C 越大，相当于惩罚松弛变量，松弛变量接近0。即对误分类的惩罚增大，趋向于对训练集全分对的情况，这样会出现训练集测试时准确率很高，但泛化能力弱，容易导致过拟合。C 值小，对误分类的惩罚减小，容错能力增强，泛化能力较强，但也可能导致欠拟合。



6. 支持向量机

6.2 线性核函数

【例6】线性核函数

```
# 例6: 线性核函数
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.datasets import make_blobs

# 先创建50个数据点, 让它们分为两类
X, y = make_blobs(n_samples = 50, centers = 2, random_state = 6)

# 创建一个线性内核的支持向量机模型
clf = svm.SVC(kernel='linear', C = 1000)
clf.fit(X, y)

# 把数据点画出来
plt.scatter(X[:, 0], X[:, 1], c = y, s = 30, cmap = plt.cm.Paired)
```



6. 支持向量机

6.2 线性核函数

【例6】线性核函数

```
#建立图像坐标
ax = plt.gca()
xlim = ax.get_xlim()
ylim = ax.get_ylim()
xx = np.linspace(xlim[0], xlim[1], 30)
yy = np.linspace(ylim[0], ylim[1], 30)
YY, XX = np.meshgrid(yy, xx)
xy = np.vstack([XX.ravel(), YY.ravel()]).T
Z = clf.decision_function(xy).reshape(XX.shape)

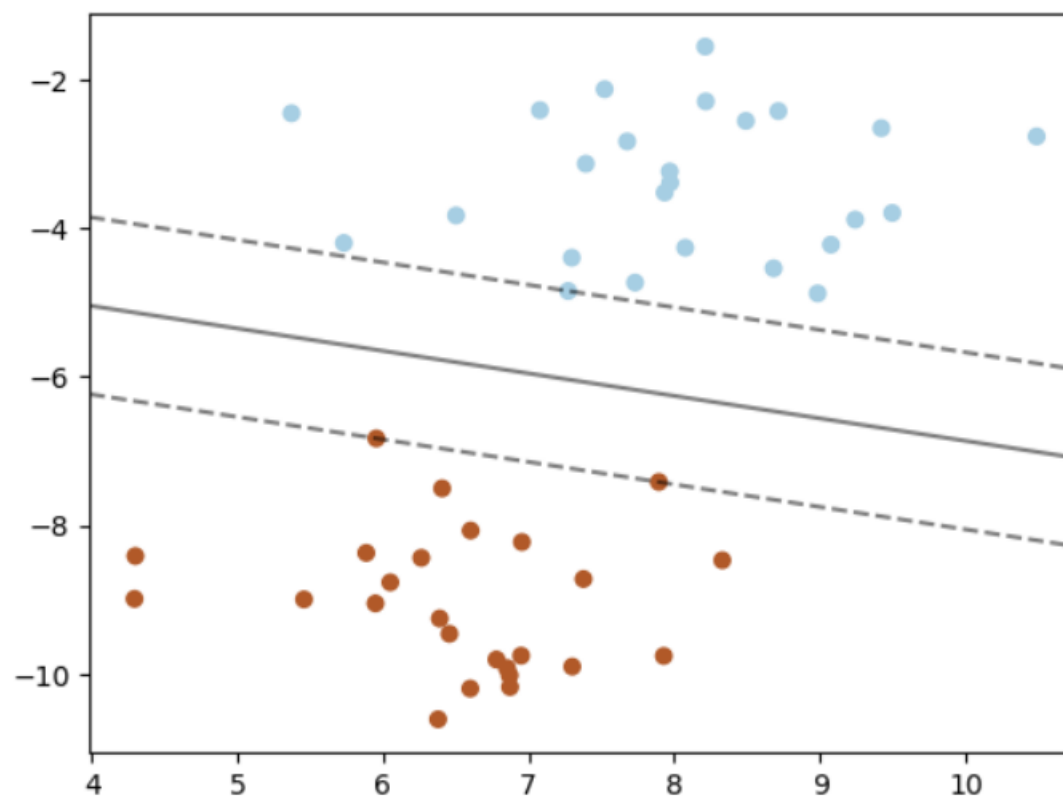
# 把分类的决定边界画出来
ax.contour(XX, YY, Z, colors = 'k', levels = [-1, 0, 1], alpha = 0.5, linestyles = ['--', '-', '--'])
ax.scatter(clf.support_vectors_[0], clf.support_vectors_[1], s = 100,
           linewidth=1, facecolors='none')
plt.show()
```



6. 支持向量机

6.2 线性核函数

【例6】线性核函数



6. 支持向量机

6.3 多项式核函数

多项式核函数（Polynomial Kernel）是指通过多项式函数增加原始样本特征的高次方幂。通过把样本原始特征进行乘方把数据投射到高维空间。多项式核函数的kernel参数取值为ploy。

```
SVC(kernel = 'ploy', degree=3)
```

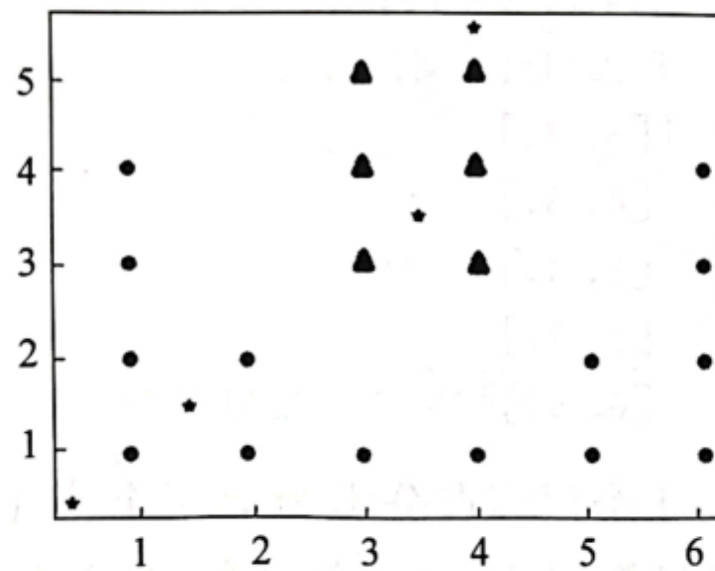
degree: 表示选择的多项式的最高次数，默认为三次多项式



6. 支持向量机

6.3 多项式核函数

【例7】区分点。圆点是正类，三角是负类，五星是预测样本点



6. 支持向量机

6.3 多项式核函数

【例7】区分点。圆点是正类，三角是负类，五星是预测样本点

```
# 例7: 区分点
# -*- coding:utf-8 -*-
from sklearn.svm import SVC
import numpy as np
X = np.array([[1,1],[1,2],[1,3],[1,4],[2,1],[2,2],[3,1],[4,1],[5,1],
              [5,2],[6,1],[6,2],[6,3],[6,4],[3,3],[3,4],[3,5],[4,3],
              [4,4],[4,5]])
Y = np.array([1] * 14 + [-1] * 6)
T = np.array([[0.5,0.5],[1.5,1.5],[3.5,3.5],[4,5.5]])
```



6. 支持向量机

6.3 多项式核函数

【例7】区分点。圆点是正类，三角是负类，五星是预测样本点

```
#X为训练样本, Y为训练样本标签(1和-1), T为测试样本
svc = SVC(kernel = 'poly', degree = 2, gamma = 1, coef0 = 0)
svc.fit(X,Y)
pre = svc.predict(T)
print("预测结果\n",pre)          #输出预测结果
print("正类和负类支持向量总个数\n",svc.n_support_)    #输出正类和负类支持向量总个数
print("正类和负类支持向量索引\n",svc.support_)        #输出正类和负类支持向量索引
print("正类和负类支持向量\n",svc.support_vectors_)    #输出正类和负类支持向量
```



6. 支持向量机

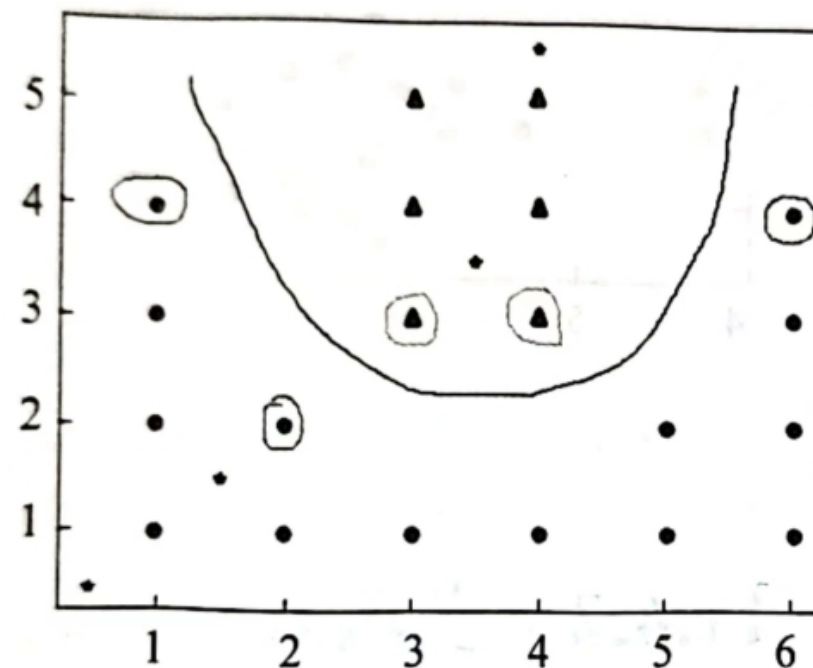
6.3 多项式核函数

【例7】区分点。圆点是正类，三角是负类，五星是预测样本点

预测结果

[1 1 -1 -1]
正类和负类支持向量总个数
[2 3]
正类和负类支持向量索引
[14 17 3 5 13]
正类和负类支持向量
[[3. 3.]
[4. 3.]
[1. 4.]
[2. 2.]
[6. 4.]]

4个预测点分类为前两个为1，后两个为-1。负类支持向量有两个，在样本集中索引为14, 17, 分别为 (3,3)、(4,3)。正类支持向量有三个，在样本集中索引为3, 5,, 3, 分别为 (1,4)、(2,2)、(6,4)。



6. 支持向量机

6.4 高斯核函数

高斯核函数也叫径向基函数，是通过高斯分布函数衡量样本与样本间的“相似度”，进而线性可分。kernel参数的取值为rbf。

```
model = svm.SVC(kernel='rbf', C)
```



6. 支持向量机

6.3 高斯核函数

【例8】高斯核函数

```
# 例8: 高斯核函数
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.datasets import make_blobs

# 先创建50个数据点, 让它们分为两类
X, y = make_blobs(n_samples = 50, centers = 2, random_state = 6)

# 创建一个RBF内核的支持向量机模型
clf_rbf = svm.SVC(kernel = 'rbf', C = 1000)
clf_rbf.fit(X, y)

# 把数据点画出来
plt.scatter(X[:, 0], X[:, 1], c = y, s = 30, cmap = plt.cm.Paired)
```



6. 支持向量机

6.3 高斯核函数

【例8】高斯核函数

```
#建立图像坐标
ax = plt.gca()
xlim = ax.get_xlim()
ylim = ax.get_ylim()

xx = np.linspace(xlim[0], xlim[1], 30)
yy = np.linspace(ylim[0], ylim[1], 30)
YY, XX = np.meshgrid(yy, xx)
xy = np.vstack([XX.ravel(), YY.ravel()]).T
Z = clf_rbf.decision_function(xy).reshape(XX.shape)

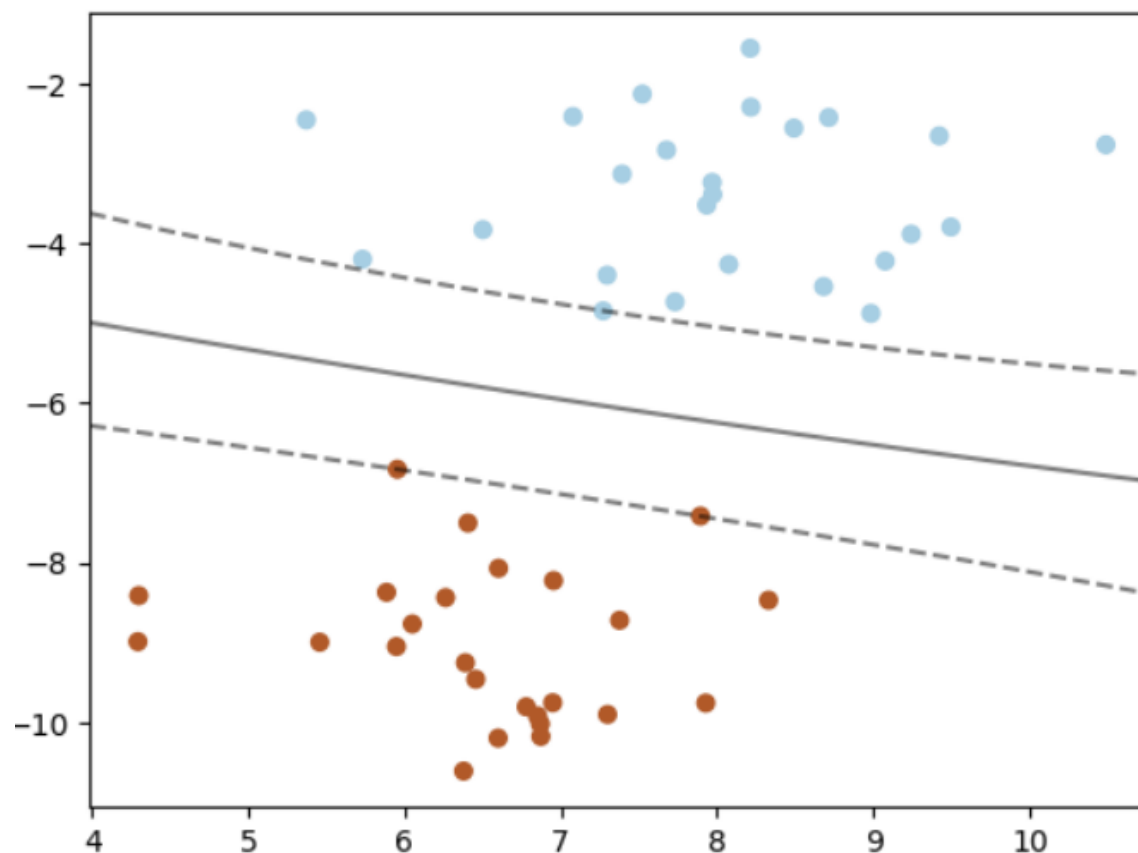
# 把分类的决定边界画出来
ax.contour(XX, YY, Z, colors = 'k', levels = [-1, 0, 1], alpha = 0.5,linestyles = ['--', '-', '--'])
ax.scatter(clf_rbf.support_vectors_[0], clf_rbf.support_vectors_[1], s = 100,
           linewidth=1, facecolors = 'none')
plt.show()
```



6. 支持向量机

6.3 高斯核函数

【例8】高斯核函数



7. 支持向量机进行鸢尾花分类

鸢尾花(Iris)数据集由Fisher 在1936收集整理，是一类多重变量分析的数据集。数据集包含150个数据样本，分为3类，分别是山鸢尾、变色鸢尾和维吉尼亚鸢尾。

鸢尾花数据集每类50个数据，每个数据包含花萼长度(sepal length)、花萼宽度(sepal width)、花瓣长度(petal length)、花瓣宽度(petal width)4个属性。通过鸢尾花的4个属性去预测鸢尾花卉属于属于三个种类中的哪一类，常用在分类操作。



7. 支持向量机进行鸢尾花分类

【例9】支持向量机对鸢尾花分类

```
# 例9: 支持向量机对鸢尾花分类
from sklearn import datasets
import sklearn.model_selection as ms
import numpy as np
import sklearn.svm as svm
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report

iris = datasets.load_iris()
x = iris.data[:, :2]
y = iris.target

# 数据集分为训练集和测试集
train_x, test_x, train_y, test_y = ms.train_test_split(x, y, test_size = 0.25, random_state = 5)
```



7. 支持向量机进行鸢尾花分类

【例9】支持向量机对鸢尾花分类

```
# 基于线性核函数
model = svm.SVC(kernel = 'linear')
model.fit(train_x, train_y)

## 基于多项式核函数, 三阶多项式核函数
# model = svm.SVC(kernel = 'poly', degree = 3)
# model.fit(train_x, train_y)

## 基于径向基(高斯)核函数
# model = svm.SVC(kernel = 'rbf', C = 600)
# model.fit(train_x, train_y)

# 预测
pred_test_y = model.predict(test_x)
```



7. 支持向量机进行鸢尾花分类

【例9】支持向量机对鸢尾花分类

```
# 计算模型精度
bg = classification_report(test_y, pred_test_y)
print('基于线性核函数 的分类报告: ', bg, sep = '\n')
# print('基于多项式核函数 的分类报告: ', bg, sep = '\n')
# print('基于径向基(高斯)核函数 的分类报告: ', bg, sep = '\n')

# 绘制分类边界线
l, r = x[:, 0].min() - 1, x[:, 0].max() + 1
b, t = x[:, 1].min() - 1, x[:, 1].max() + 1
n = 500
grid_x, grid_y = np.meshgrid(np.linspace(l, r, n), np.linspace(b, t, n))
bg_x = np.column_stack((grid_x.ravel(), grid_y.ravel()))
bg_y = model.predict(bg_x)
grid_z = bg_y.reshape(grid_x.shape)

# 画图显示样本数据
plt.title('kernel = linear ', fontsize = 16)
# plt.title('kernel = poly ', fontsize = 16)
# plt.title('kernel = rbf', fontsize = 16)
```



7. 支持向量机进行鸢尾花分类

【例9】支持向量机对鸢尾花分类

```
plt.xlabel('X', fontsize = 14)
plt.ylabel('Y', fontsize = 14)
plt.tick_params(labelsize = 10)
plt.pcolormesh(grid_x, grid_y, grid_z, cmap = 'gray')
plt.scatter(test_x[:, 0], test_x[:, 1], s = 80, c = test_y, cmap = 'jet', label = 'Samples')

plt.legend()
plt.show()
```

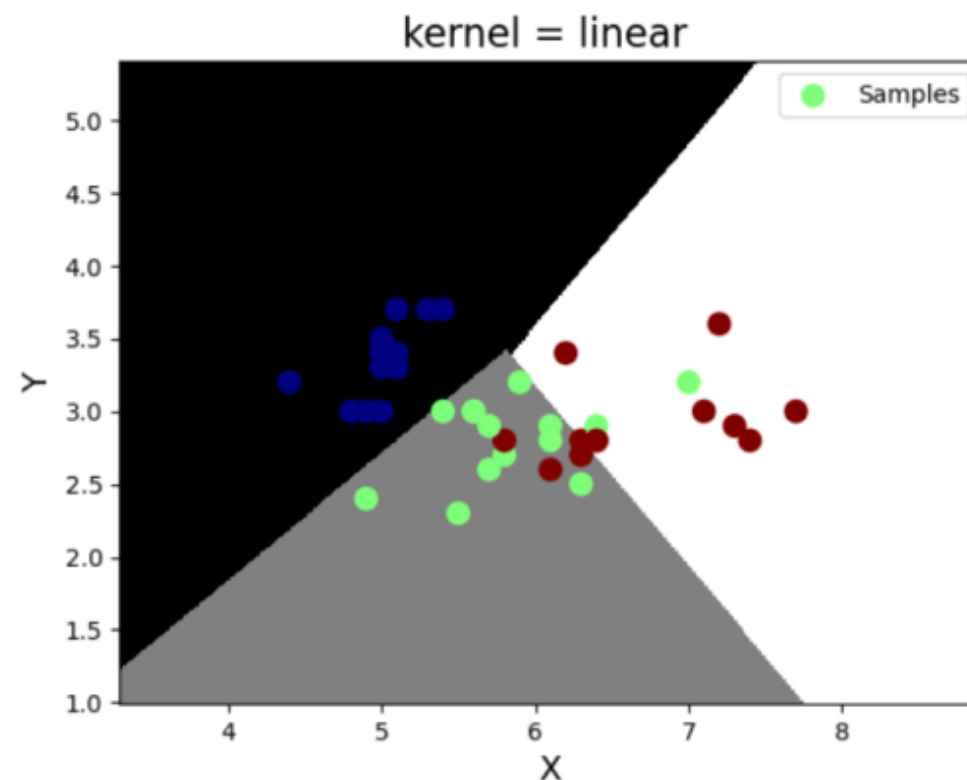


7. 支持向量机进行鸢尾花分类

【例9】支持向量机对鸢尾花分类

基于线性核函数 的分类报告:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	12
1	0.75	0.86	0.80	14
2	0.80	0.67	0.73	12
accuracy			0.84	38
macro avg	0.85	0.84	0.84	38
weighted avg	0.84	0.84	0.84	38

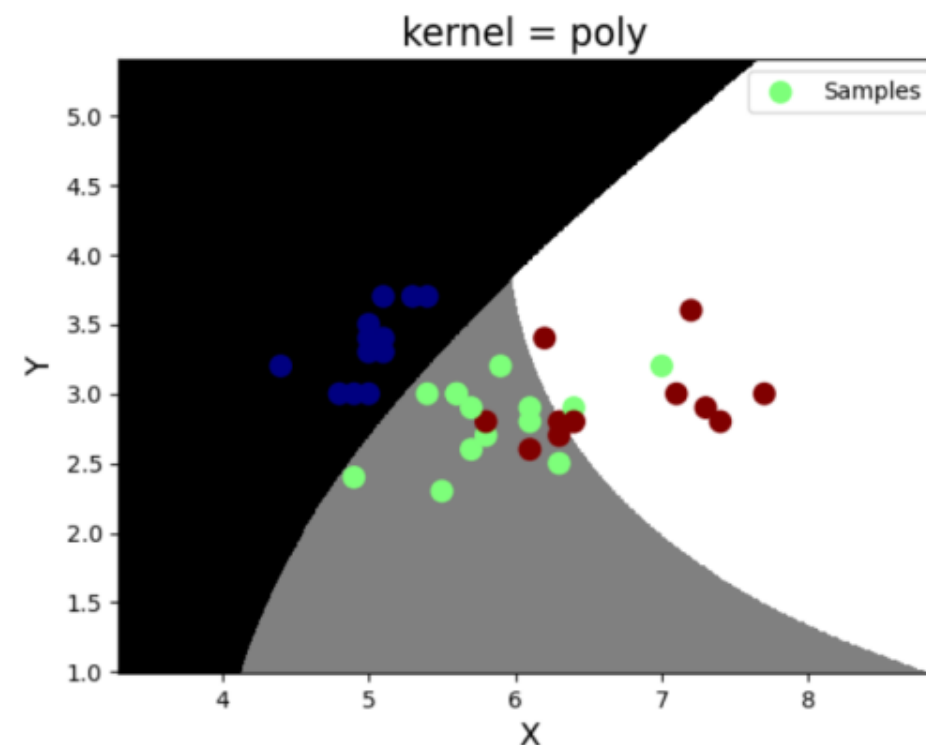


7. 支持向量机进行鸢尾花分类

【例9】支持向量机对鸢尾花分类

基于多项式核函数 的分类报告:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	12
1	0.75	0.86	0.80	14
2	0.80	0.67	0.73	12
accuracy			0.84	38
macro avg	0.85	0.84	0.84	38
weighted avg	0.84	0.84	0.84	38



7. 支持向量机进行鸢尾花分类

【例9】支持向量机对鸢尾花分类

基于径向基（高斯）核函数 的分类报告：

	precision	recall	f1-score	support
0	1.00	1.00	1.00	12
1	0.86	0.86	0.86	14
2	0.83	0.83	0.83	12
accuracy			0.89	38
macro avg	0.90	0.90	0.90	38
weighted avg	0.89	0.89	0.89	38

