西安电子科技大学人工智能学院

模式识别 课程实践报告

成　绩

实践课题名称 SVM算法验证练习

班级 2020039 姓名 刘焕宇 学号 20009200770

实践日期 2022 年 11 月

指导教师评语：
指导教师： 年 月 日

一、实验内容

支持向量机($Support\ Vector\ Machine, SVM$)是一类按监督学习方式对数据进行二元分类的广义线性分类器($generalized\ linear\ classifier$)，其决策边界是对学习样本求解的最大边距超平面($maximum-margin\ hyperplane$)。

SVM提出于1964年，在二十世纪90年代后得到快速发展并衍生出一系列改进和扩展算法，在人像识别、文本分类等模式识别问题中有得到应用。最基础的SVM算法是一种二分类模型，它将实例的特征向量映射为空间中的一些点，SVM的目的就是想要画出一条线，以“最好地”区分这两类点，以至如果以后有了新的点，这条线也能做出很好的分类。SVM 适合中小型数据样本、非线性、高维的分类问题。

非线性SVM通过核函数，将特征空间映射到高维，采用非线性变换，将线性问题转化为非线性问题，称之为“核技巧”，使得在输入空间中的超曲面模型对应于特种空间H中的超平面模型。

本实验聚焦于探究SVM算法在鸢尾花与声呐数据集上的表现，并加以学习与验证，探究不同核函数对结果的影响。

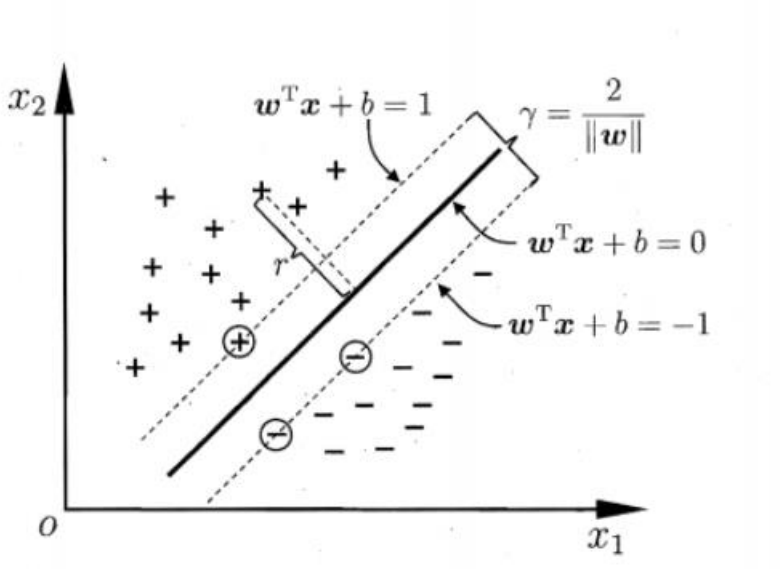


图1 SVM原理图

二、实验原理

1. 线性可分线性SVM算法介绍

设数据集中两类样本的标签分别为$+1,-1$，则给出的训练集若存在法向量ω,b使得所有样本满足$y_i*(\omega^T x+b)>0$成立，则称数据集线性可分，否则称为线性不可分。

先要找出一个线性分类器将两种样本类别的样本分开，即在n维空间中找到一个划分超平面分割两种类别的样本。划分超平面满足$\omega^T x+b=0$，由于特种空间中任意一点x到超平面的距离可以定义为：

$$r=\frac{|\omega^T x+b|}{||\omega||} \tag{1}$$

同时，我们需要让两个样本尽可能分得开，即使超平面到两类样本的最小距离尽可能大，由于ω,b同时扩大或缩小若干倍，不会影响超平面，ω的模长会影响距离，因此我们有函数间隔、几何间隔的定义，二者相差$||\omega||$倍。

不妨分类超平面到两个类别的$|\omega^T x+b|$最小值为1，显然这是可以同时扩大缩小参数，而不改变分类超平面做到的。因此我们要让最小距离最大，同时所有的点距离应该大于这个最小距离，即可得到下式：

$$max\ \frac{1}{||\omega||}$$

$$st.\ \ y_i*(\omega^T x+b)\geq 1$$

求解该凸优化问题，利用拉格朗日函数法求解即可。

2. 线性不可分线性SVM算法

若此时数据集中存在一些特异点，将这些特异点除去后，剩下大部分的样本点组成的集合是线性可分的。显然此时由于某些样本点无法再满足函数间隔大于的条件，此时我们增加松弛变量$\delta\geq 0$，与惩罚常数C，即可将线性不可分数据集转化成如下凸二次规划问题：

$$min 0.5||\omega||+C\sum\delta_i$$

$$st.\ \ y_i*(\omega^T x+b)\geq 1\text{ - }\delta_i$$

$$\delta_i\geq 0$$

最小化目标函数，即在最大化间隔的同时，使得误分类点的数量尽量小，误差尽量小。此时问题的解分类面称为软间隔的支持向量，实际结果可以通过δ_i,C,ω共同判定。

3.非线性SVM算法

对于非线性分类问题的求解，我们将特征变换到空间H，通过映射函数$\varphi(\mathrm{x})$，将映射后的特征用于支持向量机分类，而不是用原来的特征。将公式$\omega^T x+b$中的内积映射到变换空间H中。使用映射的特征原因是其能更好的拟合数据，并且对于低维空间中不可分数据，将其映射到高维空间中往往就可分了。

非线性的问题很难求解，所以我们希望可以将非线性的问题转化为线性问题，然后解线性分类问题的方法解决它，我们采用非线性变换，将其转化为线性问题，进而求解原来的非线性问题。核技巧就是采用这样的方法。如果函数$K(x,z)$，满足条件：

$$K(x,z)=\varphi(x)*\varphi(z)$$

则称$K(x,z)$为核函数。核技巧并不显式定义映射函数φ，它通过学习和预测中定义的核函数。核函数的选择对分类问题意义重大，现实生活中往往通过实验验证核函数的有效性。

常用的和核函数有：

线性核函数	$K(x,z)=x\cdot z$
多项式核函数	$K(x,z)=(x\cdot z+1)^p$
高斯核函数	$K(x,z)=\exp(-\frac{\|x-z\|^2}{2\sigma^2})$

本实验旨在探究不同核函数的选择，对分类效果的影响。

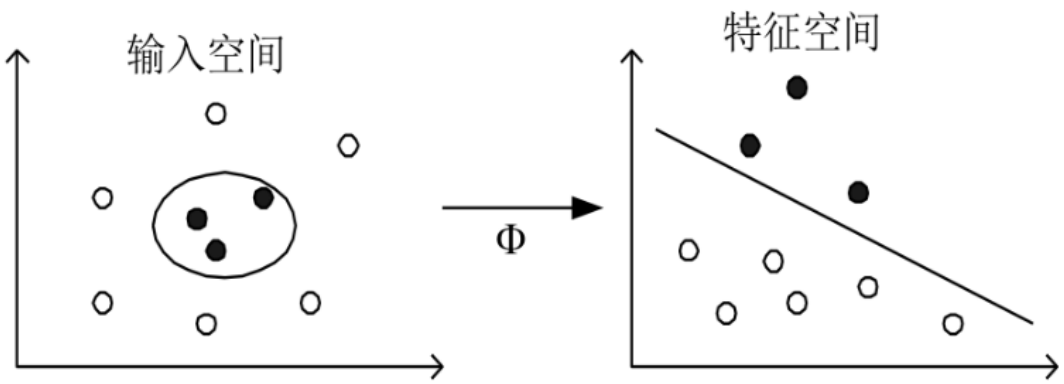
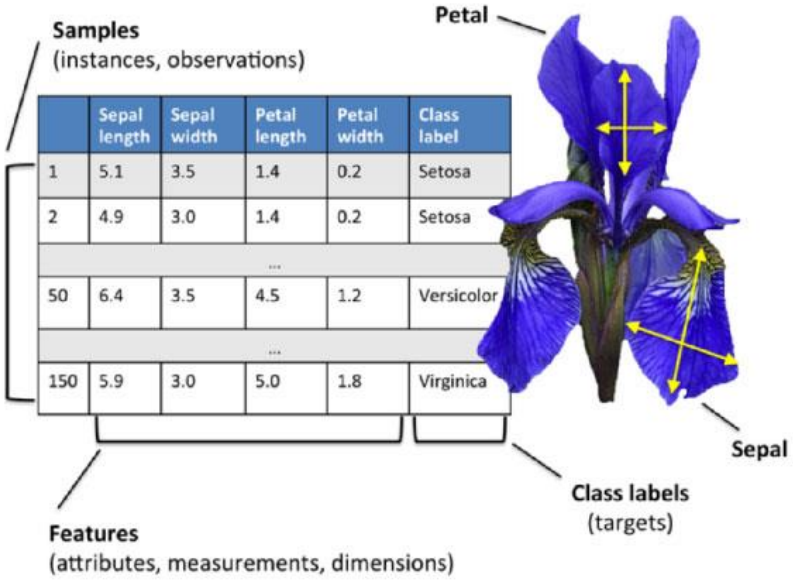


图2 核函数映射示意图

三、实验结果与分析

Iris也称鸢尾花卉数据集，是一类多重变量分析的数据集。通过花萼长度，花萼宽度，花瓣长度，花瓣宽度4个属性预测鸢尾花卉属于（Setosa(山鸢尾)，Versicolour(杂色鸢尾)，Virginica(维吉尼亚鸢尾)）三个种类中的哪类，每类50个样本，共计150个样本。



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

图3 鸢尾花数据集

分别采用高斯核函数、线性核函数、多项式核函数，选用样本的40%训练，60%作为验证，惩罚参数为1，可作出如下结果图：

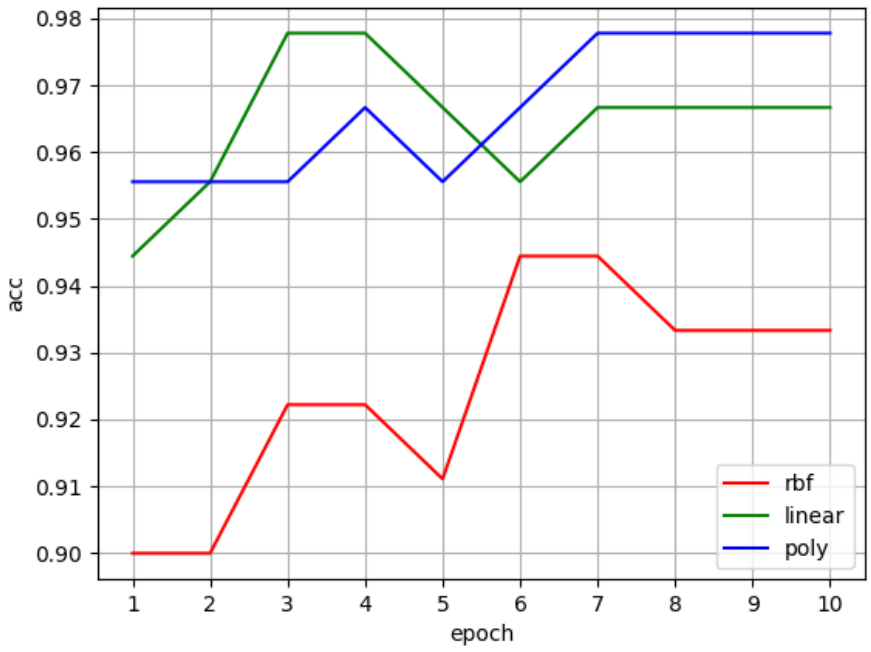


图4 鸢尾花实验结果图

*Sonar*声呐数据集来源于*UCI*，是初学机器学习常用的数据集之一。共有208行60列特征，数据分为两类，标签为R/M。表示208个观察对象，60个不同角度返回的力度值，二分类结果是岩石/金属。

分别采用高斯核函数、线性核函数、多项式核函数，选用样本的60%训练，40%作为验证，惩罚参数为1，可作出如下结果图：

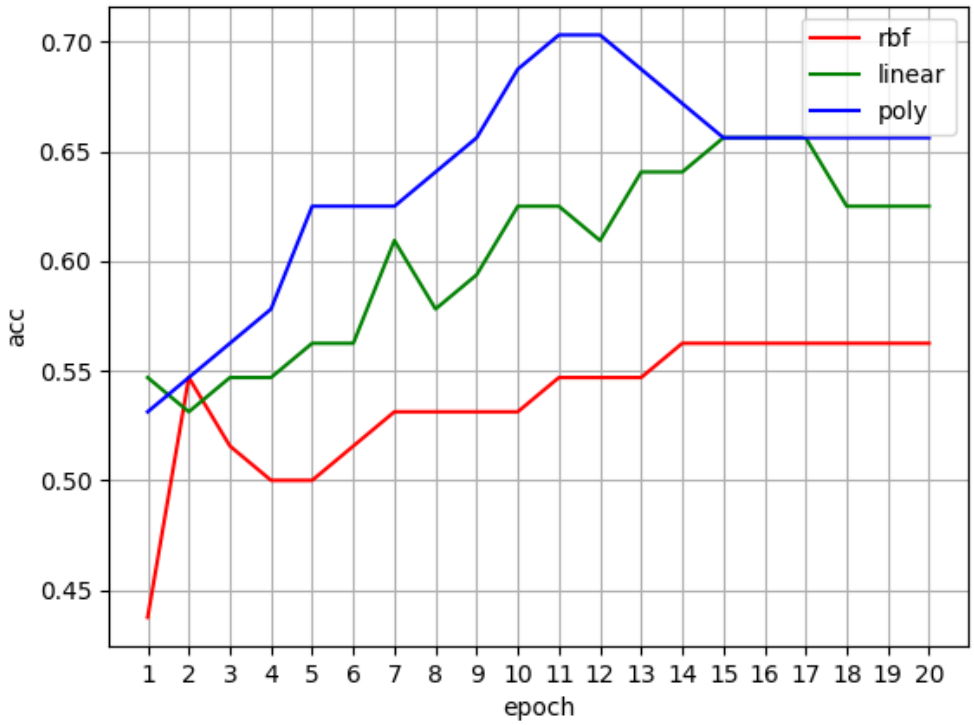


图5 声呐实验结果图

四、源程序代码

```
# Iris
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm


def work(num, typ):
    ker = 'rbf'
    if typ == 2:
        ker = 'linear'
    elif typ == 3:
```

```
        ker = 'poly'
    clf = svm.SVC(C=num / 10, kernel=ker)  # C:惩罚函数，默认是1
    # kernel：核函数，默认是rbf高斯核，可以是'linear', 'poly', 'rbf',
'sigmoid', 'precomputed'
    clf.fit(train, train_label)
    c = clf.score(test, test_label)
    return c


if __name__ == "__main__":
    df = pd.read_csv("iris.data", header=None)

    df.replace('Iris-setosa', 0, inplace=True)
    df.replace('Iris-versicolor', 1, inplace=True)
    df.replace('Iris-virginica', 2, inplace=True)
    data = np.array(df.values, dtype='float')

    siz = 20
    train = data[range(0, 0+siz), :]
    train = np.vstack((train, data[range(50, 50+siz), :]))
    train = np.vstack((train, data[range(100, 100+siz), :]))

    test = data[range(0+siz, 50), :]
    test = np.vstack((test, data[range(50+siz, 100), :]))
    test = np.vstack((test, data[range(100+siz, 150), :]))

    train_label = train[:, -1]
    train = train[:, :-1]

    test_label = test[:, -1]
    test = test[:, :-1]
    print(train.shape, train_label.shape)
    print(test.shape, test_label.shape)
    ACC1 = list()
    ACC2 = list()
    ACC3 = list()
    epoch = 10
    for num in range(1, epoch + 1):
        ACC1.append(work(num, 1))
        ACC2.append(work(num, 2))
        ACC3.append(work(num, 3))
    px = np.arange(1, epoch + 1).astype(dtype='str')
    plt.plot(px, ACC1, c='r')
    plt.plot(px, ACC2, c='g')
```

```
    plt.plot(px, ACC3, c='b')
    plt.grid()
    plt.xlabel('epoch')
    plt.ylabel('acc')
    labels = ['rbf', 'linear', 'poly']
    plt.legend(labels, loc='best', fancybox=True)
    # plt.savefig('SVM-IRIS.png')
    plt.show()

# Sonar
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm


def work(num, typ):
    ker = 'rbf'
    if typ == 2:
        ker = 'linear'
    elif typ == 3:
        ker = 'poly'
    clf = svm.SVC(C=num / 10, kernel=ker)  # C:惩罚函数，默认是 1
    # kernel: 核函数，默认是 rbf 高斯核，可以是‘linear’, ‘poly’, ‘rbf’,
‘sigmoid’, ‘precomputed’
    clf.fit(train, train_label)
    c = clf.score(test, test_label)
    return c


if __name__ == "__main__":
    df = pd.read_csv("sonar.all-data", header=None)
    df.replace('R', 0, inplace=True)
    df.replace('M', 1, inplace=True)
    Sonar = np.array(df.values, dtype='float')


    # 第一类取 0：60 为训练集 ，第二类取 97：180 为训练集
    train = Sonar[range(0, 61), :]
    train = np.vstack((train, Sonar[range(97, 180), :]))

    test = Sonar[range(61, 97), :]
    test = np.vstack((test, Sonar[range(180, 208), :]))
```

```
train_label = train[:, -1]
train = train[:, :-1]

test_label = test[:, -1]
test = test[:, :-1]

print(train.shape, train_label.shape)
print(test.shape, test_label.shape)

ACC1 = list()
ACC2 = list()
ACC3 = list()
epoch = 20
for num in range(1, epoch + 1):
    ACC1.append(work(num, 1))
    ACC2.append(work(num, 2))
    ACC3.append(work(num, 3))


px = np.arange(1, epoch + 1).astype(dtype='str')
plt.plot(px, ACC1, c='r')
plt.plot(px, ACC2, c='g')
plt.plot(px, ACC3, c='b')
plt.grid()
plt.xlabel('epoch')
plt.ylabel('acc')
labels = ['rbf', 'linear', 'poly']
plt.legend(labels, loc='best', fancybox=True)
# plt.savefig('SVM-SONAR.png')
plt.show()
```