电子科技大学信息与软件工程学院

**实 验 报 告**

学 号 2019091604016

姓 名 夏野

（实验） 课程名称 人工智能

实验时间 11月7日

实验教师 周帆

**电 子 科 技 大 学**

**实 验 报 告**

1. **实验名称：**

基于sklearn工具包进行线性回归和逻辑斯蒂回归

1. **实验学时：**

4学时

1. **实验目的：**

初步了解机器学习中的SVM，学习如何使用python重要模块sklearn实现线性SVM和核方法的SVM。其中，线性SVM通常用于线性可分的分类任务；核方法的SVM则通常用于非线性可分的分类任务。通过这次实验，了解SVM的分类原理，以及针对线性不可分的任务时SVM如果结合核函数解决问题。

1. **实验原理：**
2. SVM：支持向量机（support vector machines，SVM）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。
3. SVM的工作原理：就是在两类数据点上，确定一个超平面，使得该平面能够很好的划分这两类数据。在线性可分问题上，自然使用SVM可以行得通，只需找到划分超平面即可，但是如果是非线性可分的情况，我们就需要用到核函数来将数据转换到跟高维度，从而使用线性划分的方法进行分类。
4. 核方法：在线性 SVM 中转化为最优化问题时求解的公式计算都是以内积(dot product)形式出现的，其中 ϕ ( X ) \phi(X)ϕ(X) 是把训练集中的向量点转化到高维的非线性映射函数，因为内积的算法复杂度非常大，所以我们利用核函数来取代计算非线性映射函数的内积。
5. **实验内容：**
6. 线性SVM：
   1. 生成数据集。使用sklearn.datasets.make\_blobs函数为聚类任务生成数据集，输出数据集和相应的标签。
      1. 特征数设置为2 (n\_features)，标签数设置为2 (centers);
      2. 划分数据集，80%为训练集、其余为测试集。
   2. 数据可视化。(参考使用matplotlib)。
      1. 以第一个样本特征为x轴，第二个样本特征为y轴，绘制散点图。(根据标签着色)
   3. 搭建模型。svm.LinearSVC。
   4. train and test。训练集注入模型，随后将训练好的模型用于测试集预测。
      1. 测试结果衡量指标(metrics)：accuacy。
   5. 分析讨论。调整模型参数C，对不同结果进行分析。
7. 带核函数的SVM：
   1. 数据集：iris。
      1. 特征采用,数据集前两个特征值;
      2. 划分数据集，前130为训练集，其余为测试集。
   2. 搭建模型。svm.SVC ( kernel = ‘ rbf ’ )，选择核函数。
   3. train and test。训练集注入模型，随后将训练好的模型用于测试集预测
      1. 测试结果衡量指标(metrics)：accuacy，precision and recall。
   4. 可视化。绘制支持向量机分类边界。
8. **实验器材（设备、元器件）：**

1. 硬件：mac笔记本电脑

2. 软件：ios系统，pycharm

1. **实验步骤：**
2. 线性SVM：
   1. 生成数据集：使用sklearn.datasets.make\_blobs()方法为该任务生成数据集，传入参数指定数据量为1000，特征有2个，标签值设定为2；
   2. 分割数据集：根据实验要求，需要将刚刚生成的数据分割为训练集和测试集，比例为4:1。这里，我采用sklearn中的train\_test\_split()方法进行处理，其中最重要的参数train\_size设置为0.8；
   3. 数据可视化：使用matplotlib实现数据可视化。首先导入matplotlib.pyplot，并缩写为plt。根据实验的要求，需要画出散点图，因此我们使用scatter()方法实现，其中需要传入X的值和y轴的值。这样，我们就创建了一个figure用于存放scatter这个图片，但是还无法显示这个图，还需要使用show()方法将figure上的图显示出来；
   4. 数据预处理：这里我依然使用StandardScaler来对数据进行标准化，首先使用fit()方法获得元数据的均值和方差，然后在训练和测试数据上分别使用transform方法进行数据标准化处理；
   5. 搭建模型：使用sklearn包中的svm模块搭建模型。其中包含LinearSVC，同时首先设置内部参数C，它表示一个浮点数，惩罚参数。我先设置值为1，也就是默认值；
   6. 训练模型：使用fit()方法，将训练数据和对应target作为参数，开始训练模型。之后，进行模型评估，svc具有score()方法，将测试数据和起对应target作为参数，计算得到accuracy。需要注意的是，这里的准确度不是针对训练数据的acc，而是模型从未见过的数据的acc，这样更能反应模型的泛化能力；
   7. 调整C：利用循环，每次修改C的值为10的次方，这样每次都打印出来acc的值，进行比较。
3. 核函数的SVM：
   1. 加载数据：使用load\_iris()函数，加载数据。根据实验要求，只取前2个特征，因此使用切片[0:2]获得数据的前两个特征；同时，在切割数据集的时候，根据要求，训练数据130比，其余为测试数据，也利用切片进行分割操作；
   2. 数据可视化：利用matplotlib模块进行数据可视化操作，利用scatter()进行散点图的绘制，最后通过show()方法进行数据的显示；
   3. 搭建模型：使用sklearn包中的svm模块搭建模型。其中包含SVC类，我定义了一个svc对象，同时设置参数kernel=‘rbf’，也就是定义核函数为rbf；
   4. 训练模型：使用fit()方法，将训练数据和对应target作为参数，开始训练模型。之后，进行模型评估，svc具有score()方法，将测试数据和起对应target作为参数，计算得到accuracy。
   5. 可视化：在进行了模型训练后，使用matplotlib绘制出向量机的分类边界。
4. **实验结果与分析（含重要数据结果分析或核心代码流程分析）**

1. 实验一 - 线性SVM：

1. 实验结果及分析：

（1）数据分布可视化：

**Chart, scatter chart

Description automatically generated**

结果分析：

获取数据后，为数据进行分类后的到的可视化散点图数据。

（2）C的参数变化：

Text

Description automatically generated

结果分析：

由于数据量太小，我们可以看到虽然我们的惩罚参数C的值以10的指数级变化，但是训练结果的值一直是没有改变的。

b. 核心代码流程分析：

（1）生成数据集代码

X, y = make\_blobs(n\_samples=1000, n\_features=2, centers=2)

代码分析：

调用make\_bolbs()函数生成数据集，并设置n\_samples,n\_features,centers三个参数。他们对应的含义是：样本数量为1000个， 样本的特征个数为2个， 样本的标签个数为2个。

（2）分割数据集代码

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, train\_size=0.8)

代码分析：

调用train\_test\_split()函数将刚刚生成的数据集分为训练数据集以及测试数据集，并且根据要求需要将训练数据集和测试数据集的比例设置为4：1，所以这里我们还需要设置train\_size参数为0.8。

（3）数据可视化代码

plt.scatter(X[:, 0], X[:, 1], c=y)  
plt.show()

代码分析：

首先我们需要导入matplotlib.pyplot包，并将它的别名设置为plt。接着调用plt的scatter()方法。这样就可以将plt对象中存入我们的相关数据。最后调用plt的show()方法,就可以在运行代码时生成数据可视化后的散点图。

（4）数据预处理（标准化）代码

standardScaler = StandardScaler()  
standardScaler.fit(X\_train)  
X\_train = standardScaler.transform(X\_train)  
X\_test = standardScaler.transform(X\_test)

代码分析：

首先我们调用StandardScaler()函数创建standardScaler对象来对数据进行标准化，接着我们调用fit()方法获得元数据的均值和方差，然后我们在训练和测试数据上分别使用transform()方法进行数据标准化处理,并将结果存入X\_train、X\_test两个变量中。

（5）搭建模型代码

svc = LinearSVC(C=1)

代码分析：

直接调用LinearSVC()函数创造svc对象，来搭建svc模型。这里我们向方法中传入参数`C=1`作为惩罚参数，并且设置为默认值1。

（6）训练模型代码

svc.fit(X\_train, y\_train)  
y\_predict = svc.predict(X\_test)

代码分析：

首先，我们调用svc的fit()方法，将训练数据和对应target作为参数，开始训练模型。之后，进行模型评估，svc具有score()方法，将测试数据和起对应target作为参数，计算得到accuracy。需要注意的是，这里的准确度不是针对训练数据的acc，而是模型从未见过的数据的acc，这样更能反应模型的泛化能力。

（7）调整惩罚参数C代码

for i in range(1, 4):  
 svc = LinearSVC(C=10\*\*i)  
 svc.fit(X\_train, y\_train)  
 y\_predict = svc.predict(X\_test)

代码分析：

使用for循环，每次修改C的值为10的次方，这样每次都打印出来acc的值，进行比较。

2. 实验二 – 核方法的SVM：

a.实验结果及分析：

（1）运行结果

Text

Description automatically generated

分析：

经过模型分析，我们可以得到和方法的SVM的acc值为0.8383615...可以看到是比较高的，能够被我们所接受。

（2）散点图、边界划分图

Chart, scatter chart

Description automatically generated

散点图

Chart, scatter chart

Description automatically generated

边界划分图

分析：

成功得到数据的散点图以及边界划分的图像。

b.核心代码流程分析

（1）装在数据集代码

iris = load\_iris()  
X = iris.data[:, 0:2]  
y = iris.target[:]

代码分析：

调用load\_iris()函数将我们的iris的数据集装在进入iris对象中。接着将我们的iris数据集的data和target分别划分给X，y两个变量

（2）划分数据集代码

X\_train = X[:130, :]  
y\_train = y[:130]  
X\_test = X[130:, :]  
y\_test = y[130:]

代码分析：

分别将X，y两个变量划分为训练集和测试集两个部分。

（3）数据可视化，做出数据的散点图

plt.scatter(X[:, 0], X[:, 1], c=y)  
plt.show()

代码分析：

方法和上一个代码类似，这里不做详细解释

（4）模型搭建代码

svc = SVC(kernel='rbf')

代码分析：

调用SVC()函数构建svc模型对象。这里传入参数`kernel=’rbf’`表明定义核函数为’rbf’。

（5）训练模型代码

svc.fit(X\_train, y\_train)  
y\_predict = svc.predict(X\_train)

print("y\_predict.shape: ", y\_predict.shape)  
print('The acc is', svc.score(X\_train, y\_train))

代码分析：

首先，我们调用svc模型对象的fit()函数训练模型；接着，我们调用svc的predict()函数模型来验证预测性。进行模型评估，svc具有score()方法，将测试数据和起对应target作为参数，计算得到accuracy。

（6）边界可视化代码

assert isinstance(y, object)  
plt.scatter(X[:, 0], X[:, 1], c=y)  
  
def plot\_hyperplane(clf, X, y, h=0.02, draw\_sv=True, title='hyperplane'):  
 x\_min, x\_max = X[:, 0].min() - 0.1, X[:, 0].max() + 0.1  
 y\_min, y\_max = X[:, 1].min() - 0.1, X[:, 1].max() + 0.1  
 xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h),  
 np.arange(y\_min, y\_max, h))  
  
 plt.title(title)  
 plt.xlim(xx.min(), xx.max())  
 plt.ylim(yy.min(), yy.max())  
 plt.xticks(())  
 plt.yticks(())  
  
 Z = clf.predict(np.c\_[xx.ravel(), yy.ravel()]) # SVM的分割超平面  
  
 Z = Z.reshape(xx.shape)  
 plt.contourf(xx, yy, Z, cmap='hot', alpha=0.5)  
 plt.scatter(X[:, 0], X[:, 1], c=y)  
  
 if draw\_sv:  
 sv = clf.support\_vectors\_  
 plt.scatter(sv[:, 0], sv[:, 1], c='r', marker='.', s=1)  
  
 plt.show()  
  
  
plot\_hyperplane(svc, X\_train, y\_train)

代码分析：

首先调用plt对象的scatter()函数，将我们的数据集转载进入对象中。接着定义plot\_hyperplane。接着为plt模型输入一些默认的定义值。接着调用clf的predict()方法来进行SVM分割超平面，并创建Z对象。接着调用Z的reshape()函数。并通过contourf()函数设置相关颜色。最后调用plt的show()函数来展示我们的数据图。

1. **总结及心得体会：**

完成实验二后，我对于机器学习中的SVM有了初步的了解；并且通过该实验我更加熟悉了python中重要模块sklearn中如何实现线性SVM和核方法SVM。其中包含了两类非常经典的的任务：线性可分割任务、线性不可分割任务。线性SVM常用于线性可分割的分类任务，而核方法SVM则通常用语非线性可分割的分类任务。

通过此次实验我了解了SVM的分类原理，以及针对线性不可分的任务时如何结合核函数来解决问题。

同时对于python语言的语法更加熟悉。