一、SVC的使用

SVM模型有两个非常重要的参数C与gamma。其中 C是惩罚系数，即对误差的宽容度。c越高，说明越不能容忍出现误差,容易过拟合。C越小，容易欠拟合。C过大或过小，泛化能力变差

gamma是选择RBF函数作为kernel后，该函数自带的一个参数。隐含地决定了数据映射到新的特征空间后的分布，gamma越大，支持向量越少，gamma值越小，支持向量越多。支持向量的个数影响训练与预测的速度。

$$k(x,z) = \exp(-\frac{d(x,z)^2}{2*\sigma^2}) = \exp(-gamma \cdot d(x,z)^2) \Rightarrow gamma = \frac{1}{2\cdot\sigma^2}$$

这里面大家需要注意的就是gamma的物理意义，大家提到很多的RBF的幅宽，它会影响每个支持向量对应的高斯的作用范围，从而影响泛化性能。我的理解：**如果gamma设的太大，σ会很小，σ很小的高斯分布长得又高又瘦， 会造成只会作用于支持向量样本附近**，对于未知样本分类效果很差，存在训练准确率可以很高，(如果让下σ无穷小，则理论上，高斯核的SVM可以拟合任何非线性数据，但容易过拟合)而测试准确率不高的可能，就是通常说的过训练；而如果设的过小，则会造成平滑效应太大，无法在训练集上得到特别高的准确率，也会影响测试集的准确率。

> 就是说C大过拟合 gamma大过拟合

PYTHON

```python
sklearn.svm._classes.SVC
def __init__(self,
             *,
             C: Any = 1.0,
             kernel: Any = "rbf",
             degree: Any = 3,
             gamma: Any = "scale",
             coef0: Any = 0.0,
             shrinking: Any = True,
             probability: Any = False,
             tol: Any = 1e-3,
             cache_size: Any = 200,
             class_weight: Any = None,
```

```
                 verbose: Any = False,
                 max_iter: Any = -1,
                 decision_function_shape: Any = "ovr",
                 break_ties: Any = False,
                 random_state: Any = None) -> None
```

线性

```python
model = SVC(C=100,kernel='linear')

w = model.coef_
b = model.intercept_
```

高斯核函数

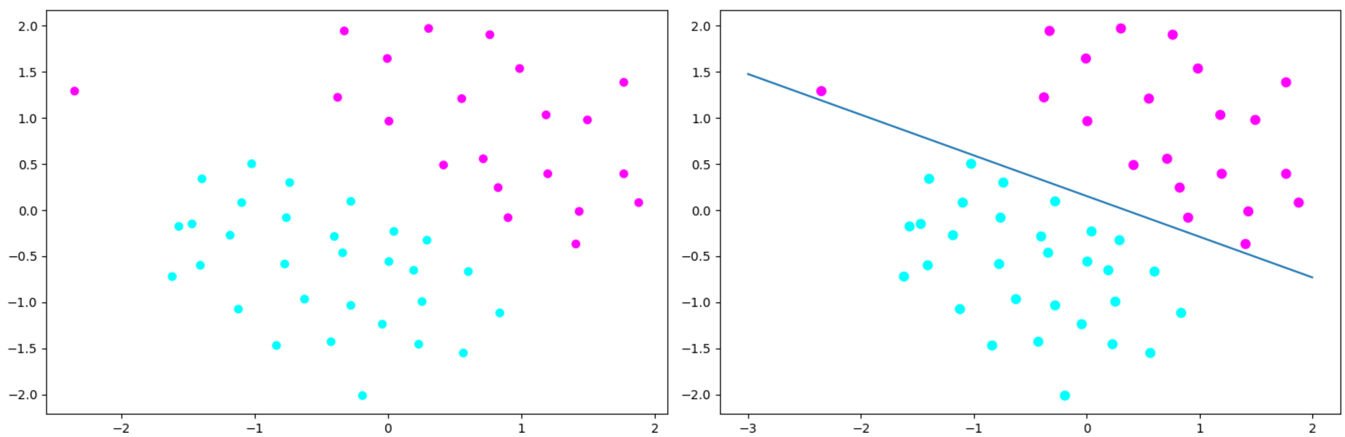
```python
model = SVC(C=1,kernel='rbf')
model.fit(data[[0,1]],data['y'])
plt.subplot(2,2,i+1)
# plt.title(f'c = {cs[i]}')
# 然后预测再画图
x = y = np.linspace(-2,2,50)
xx,yy = np.meshgrid(x,y)
# Z = np.vstack([xx.ravel(), yy.ravel()]).T
Z = np.c_[xx.ravel(), yy.ravel()]

length = model.decision_function(Z).reshape(xx.shape)

targets = model.predict(data[[0,1]])
# 画图
plt.contour(xx,yy,length,[-1,0,1],cmap='coolwarm')
plt.scatter(data[0], data[1], s=5, c=data['y'] , cmap='cool')
```

decision_function() # SVM中每个点到猜测的超平面的距离

二、线性SVM

```python
# 导入包
import numpy as np
from scipy.io import loadmat
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.svm import SVC

# 数据展示
data1 = loadmat('/root/pycharmDemo/ML/data/MLData/6/ex6data1.mat')
X = data1['X']
Y = data1['y']
datapdX = pd.DataFrame(X)
datapdX = datapdX.apply(lambda x: (x - x.mean()) / (x.std()))
X = np.array(datapdX)
plt.figure(figsize=(15,5))
# isZero = (Y == 0).reshape(-1)
plt.subplot(1,2,1)
# plt.scatter(X[isZero, 0], X[isZero, 1], c='r', label='class 0')
# plt.scatter(X[~isZero, 0], X[~isZero, 1], c='b', label='class 1')
plt.scatter(X[:, 0], X[:, 1], c=Y, cmap='cool')


# 使用sklearn中的SVC函数进行

model = SVC(C=100,kernel='linear')
model.fit(X,Y)



target = model.predict(X)
plt.subplot(1,2,2)
plt.scatter(X[:,0], X[:,1], s=50, c=Y, cmap='cool')

w = model.coef_
b = model.intercept_

x = np.linspace(-3,2,50)
y = -1 * (w[0,0] * x + b) / w[0,1]
```

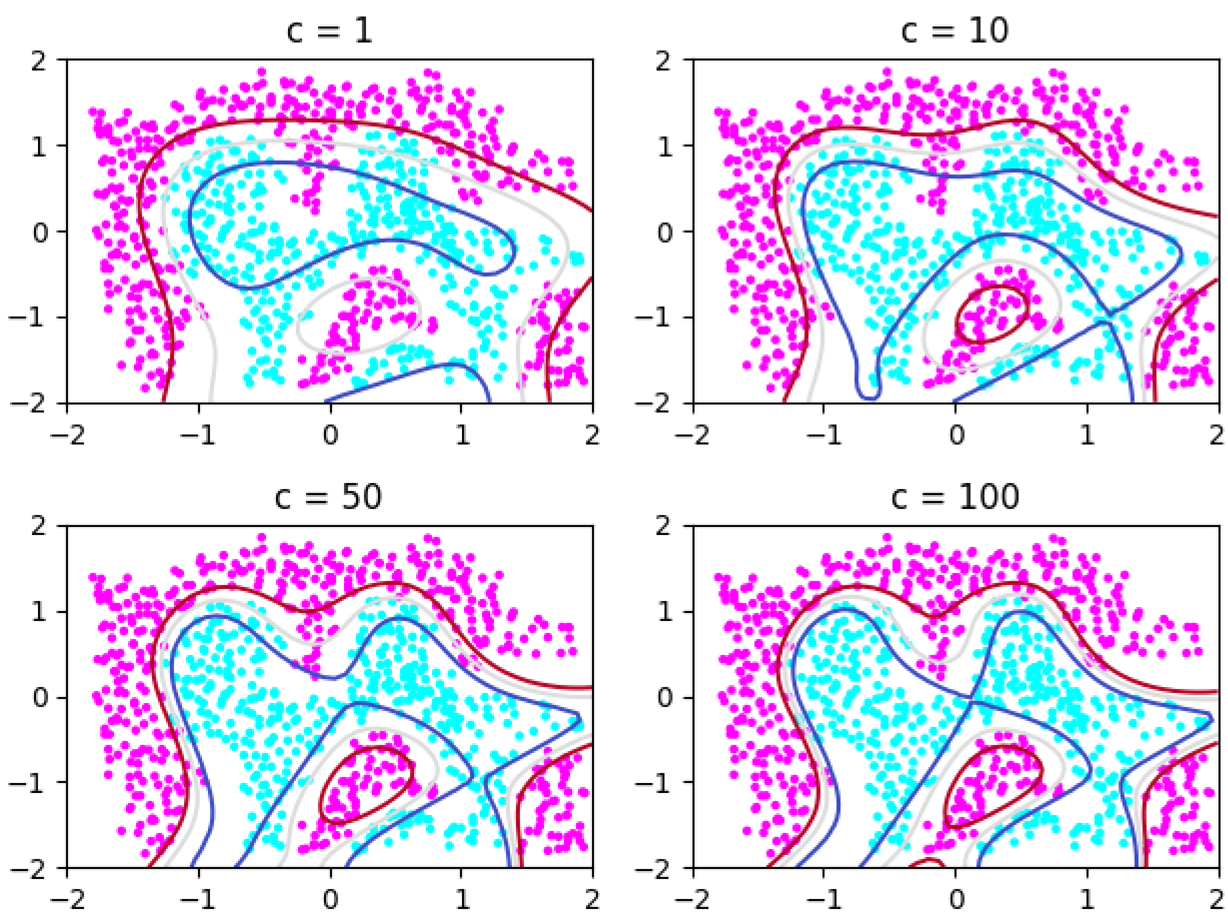
```
plt.plot(x,y)

plt.show()
```



三、核函数

```python
# 导入包
import numpy as np
from scipy.io import loadmat
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.svm import SVC
data2 = loadmat('/root/pycharmDemo/ML/data/MLData/6/ex6data2.mat')
data2['X'].shape ,data2['y'].shape
data = pd.DataFrame(data2['X'],dtype=float)
data['y'] = data2['y']
all_float64 = data.dtypes[data.dtypes == 'float'].index

data[all_float64] = data[all_float64].apply(lambda x: (x - x.mean()) /

cs = [1,10,50,100]
for i in range(len(cs)):
    model = SVC(C=cs[i],kernel='rbf')
    model.fit(data[[0,1]],data['y'])
    plt.subplot(2,2,i+1)
    plt.title(f'c = {cs[i]}')
    # 然后预测再画图
```

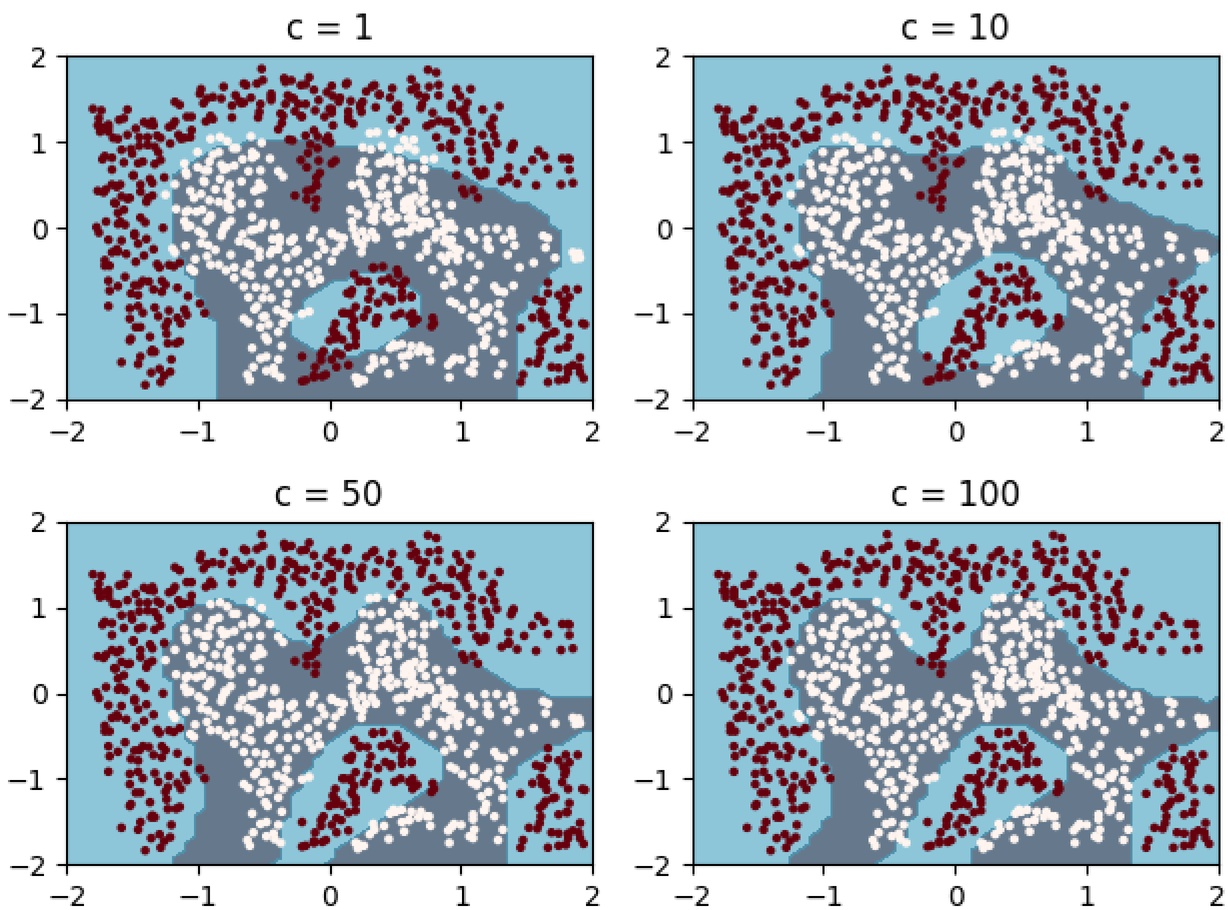
```
 x = y = np.linspace(-2,2,50)
    xx,yy = np.meshgrid(x,y)
    # Z = np.vstack([xx.ravel(), yy.ravel()]).T
 Z = np.c_[xx.ravel(), yy.ravel()]

    length = model.decision_function(Z).reshape(xx.shape)

    targets = model.predict(data[[0,1]])
    # 画图
 plt.contour(xx,yy,length,[-1,0,1],cmap='coolwarm')
    plt.scatter(data[0], data[1], s=5, c=data['y'] , cmap='cool')
plt.show()
```



3.2 contourf画图

```python
# 导入包
import numpy as np
```

```python
from scipy.io import loadmat
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.svm import SVC
data2 = loadmat('/root/pycharmDemo/ML/data/MLData/6/ex6data2.mat')
data2['X'].shape ,data2['y'].shape
data = pd.DataFrame(data2['X'],dtype=float)
data['y'] = data2['y']
all_float64 = data.dtypes[data.dtypes == 'float'].index
data[all_float64] = data[all_float64].apply(lambda x: (x - x.mean()) /

cs = [1,10,50,100]
for i in range(len(cs)):
    model = SVC(C=cs[i],kernel='rbf')
    model.fit(data[[0,1]],data['y'])
    plt.subplot(2,2,i+1)
    plt.title(f'c = {cs[i]}')
    # 然后预测再画图
 x = y = np.linspace(-2,2,50)
    xx,yy = np.meshgrid(x,y)
    # Z = np.vstack([xx.ravel(), yy.ravel()]).T
 Z = np.c_[xx.ravel(), yy.ravel()]

    length = model.decision_function(Z).reshape(xx.shape)
    all_targets = model.predict(Z).reshape(xx.shape)
    targets = model.predict(data[[0,1]])
    # 画图
 # plt.contour(xx,yy,length,[-1,0,1],cmap='coolwarm')
 plt.contourf(xx,yy,all_targets,[-1,0,1],cmap=plt.cm.ocean, alpha=0.6)
    plt.scatter(data[0], data[1], s=5, c=data['y'] , cmap='Reds')
plt.show()
```



四、最优参数

4.1 手动调参

PYTHON

```python
# 导入包
import numpy as np
from scipy.io import loadmat
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.svm import SVC
from sklearn import metrics
# 数据展示
data3 = loadmat('/root/pycharmDemo/ML/data/MLData/6/ex6data3.mat')
X = data3['X']
Y = data3['y']
datapd = pd.DataFrame(X,columns=['X1','X2'])
# datapd = datapd.apply(lambda x: (x - x.mean()) / (x.std()))
```

```
datapd['y'] = data3['y']
plt.scatter(datapd['X1'], datapd['X2'], c=datapd['y'],s=10)
# 验证集
cv = pd.DataFrame(data3.get('Xval'), columns=['X1', 'X2'])
cv['y'] = data3.get('yval')
plt.scatter(cv['X1'], cv['X2'], c=cv['y'],s=10)
candidate = [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30, 100]
combination = [(C,gamma) for gamma in candidate for C in candidate]
score_list = []
for C,gamma in combination:
    model = SVC(C=C,gamma=gamma)
    model.fit(datapd[['X1','X2']] , datapd['y'])
    score_list.append(model.score(cv[['X1','X2']] , cv['y']))
idx = np.argmax(score_list)
score_list[idx]
best_model = SVC(C=30,gamma=10)
best_model.fit(datapd[['X1','X2']] , datapd['y'])
ypred = best_model.predict(cv[['X1', 'X2']])

print(metrics.classification_report(cv['y'], ypred))
```

```
[152]: best_model = SVC(C=30,gamma=10)
best_model.fit(datapd[['X1','X2']] , datapd['y'])
ypred = best_model.predict(cv[['X1', 'X2']])

print(metrics.classification_report(cv['y'], ypred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.96      0.97       113
           1       0.95      0.97      0.96        87

    accuracy                           0.96       200
   macro avg       0.96      0.97      0.96       200
weighted avg       0.97      0.96      0.97       200
```

4.2 GridSearchCV

> 接着上面有关代码走

```python
parameters = {'C': candidate, 'gamma': candidate}
svc = SVC()
clf = GridSearchCV(svc, parameters, n_jobs=-1)
clf.fit(datapd[['X1', 'X2']], datapd['y'])
clf.best_params_
clf.best_score_
ypred = clf.predict(cv[['X1', 'X2']])
print(metrics.classification_report(cv['y'], ypred))
```

```
[161]: GridSearchCV(estimator=SVC(), n_jobs=-1,
                    param_grid={'C': [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30, 100],
                                'gamma': [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30, 100]})

[162]: clf.best_params_

[162]: {'C': 30, 'gamma': 3}

[163]: clf.best_score_

[163]: 0.9194905869324475

[164]: ypred = clf.predict(cv[['X1', 'X2']])
       print(metrics.classification_report(cv['y'], ypred))

              precision    recall  f1-score   support

           0       0.95      0.96      0.96       113
           1       0.95      0.93      0.94        87

    accuracy                           0.95       200
   macro avg       0.95      0.95      0.95       200
weighted avg       0.95      0.95      0.95       200
```

参考文档

decision_function
Python中np.c*和np.r*的区别
SVM基础用法以及可视化