机器学????告

姓名??崇?

学号?U201610531

班???信中英1601

??内容?SVM

二.??任??
任?一?分?用?性SVM和高斯核SVM???数据?行分?
任?二?使用高斯核SVM??定数据集?行分?
任?三?使用?性SVM???垃圾?件分?

三.??原理?

1. SVM求解:

根据拉格朗日?偶性??成如下形式?再由KKT条件求出?足α*?由α*得到最?的w*?b*

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^{\top} \boldsymbol{x}_j$$

$$s.t. \quad \sum_{i=1}^{m} \alpha_i y_i = 0,$$

$$\alpha_i \geq 0, i = 1, 2, ..., m.$$

再得

$$w_* = \sum_{i=1}^{N} \alpha_i^* y_i x_i$$

$$b_* = y_j - \sum_{i=1}^{N} \alpha_i^* y_i (x_i^T x_j)$$

2. ??隔SVM:

当数据集不是?性可分???本点不?足函数?隔大于等于1的?束条件??解决???引入非?松弛?量ξ?

使?束条件??

$$y_i(w_Tx_i+b)\geq 1-\xi_i$$

引入??系数C>0,?目?函数??

$$\frac{1}{2}$$

C越大越接近于硬?隔
??又?成

$$\min_{w,b,\xi}\frac{1}{2}$$

s.t.

$$y_i(w_Tx_i+b)\geq 1-\xi_i$$

$$\xi_i\geq 0$$

3. 核函数:

若数据不?性可分??引入核函数?将其映射到新的空?
核函数??X??入空?(欧式空?R_n的子集)?H?特征空?(希?伯特空?)?存在一个X到H的映射

$$\Phi(x):X\rightarrow H$$

?核函数?

$$x,z\in X$$

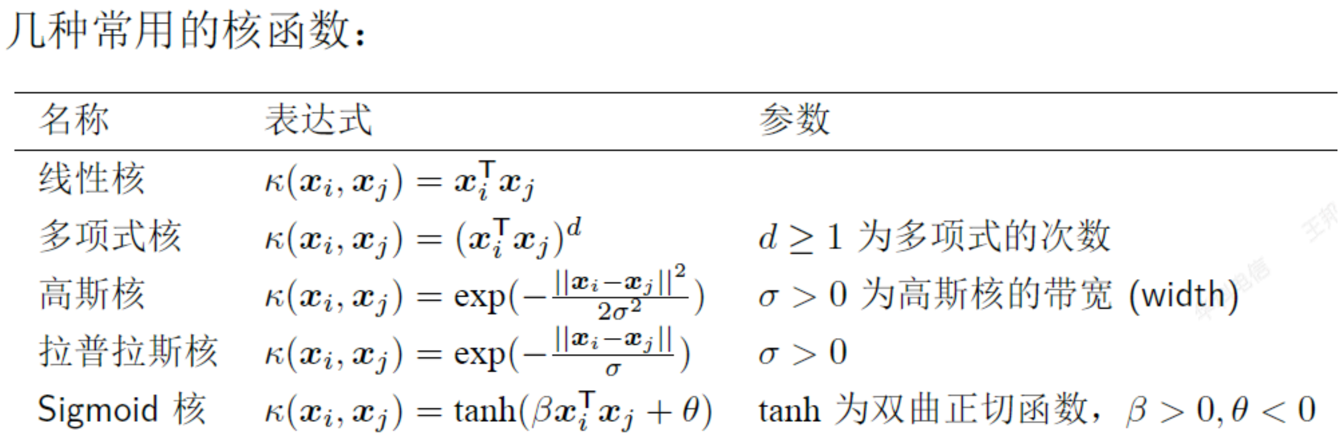
$$K(x,z) = \Phi(x)_T \Phi(z)_T$$

SVM求解又?成

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

$$s.t. \quad \sum_{i=1}^{m} \alpha_i y_i = 0,$$

$$\alpha_i \geq 0, i = 1, 2, ..., m.$$



几种常用的核函数：

名称	表达式	参数
线性核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j$	
多项式核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j)^d$	$d \geq 1$ 为多项式的次数
高斯核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2})$	$\sigma > 0$ 为高斯核的带宽 (width)
拉普拉斯核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{\sigma})$	$\sigma > 0$
Sigmoid 核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(\beta \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j + \theta)$	tanh 为双曲正切函数，$\beta > 0, \theta < 0$

4. SMO

?快速求解SVM?采用启?式算法SMO(sequential minimal optimization),包括两个部分?求解两个?量二次?划的解析方法和???量的启?方法

基本思路

> 1. 若所有?量**α**都?足最?化的KKT条件?最后化??的解就得到
> 2. 否??取两个?量?固定其他?量????两个?量构建一个二次?划????化?两个?量

两个?量α的更新?

$$???\alpha_{2}^{new,unc} = \alpha_{2}^{old} + \frac{y_2(E_1 - E_2)}{\eta}$$

$$\alpha_{2}^{old} = \begin{cases} H, \alpha_{2}^{new,unc} > H \\ \alpha_{new,unc}^{2}, L \le \alpha_{new,unc} \le H \\ L_{;}^{2} \alpha_{2}^{new,unc} > L^2 \end{cases}$$

$$\alpha_{1}^{new} = \alpha^{old} + y_1 y_2 (\alpha_{2}^{old} - \alpha_{2}^{old})$$

其中?

$$\eta = ||\Phi(x_1) - \Phi(x_2)||_2$$

$$??E_i = (\sum_{j=1}^{N} \alpha_j y_j K(x_j, x_i) + b) - y_i$$

$$L = \begin{cases} \\ \max(0, \alpha^{old} - \alpha^{new}), y_1 = y_2 \end{cases}$$
$$\max(0, \alpha_{2}^{2}{}_{old} - \alpha_{1}^{1}{}_{new} - C), y_1 \neq y_2$$

$$H = \begin{cases} \\ \min(C, C + \alpha_{\substack{old\\2}} + \alpha_{\substack{new\\1}}), y_1 = y_2 \end{cases}$$
$$\min(C, \alpha_{\substack{old\\2}} + \alpha_{\substack{new\\1}}), y_1 \neq y_2$$

??b的更新?

$$b_{\substack{new\\1}} = -E_1 - y_1 K_{11}(\alpha_{\substack{new\\1}} - \alpha_{\substack{old\\1}}) - y_2 K_{21}(\alpha_{\substack{new\\2}} - \alpha_{\substack{old\\2}}) + b_{old}$$

$$b_{\substack{new\\2}} = -E21 - y_1 K_{12}(\alpha_{\substack{new\\1}} - \alpha_{\substack{old\\1}}) - y_2 K_{22}(\alpha_{\substack{new\\2}} - \alpha_{\substack{old\\2}}) + b_{old}$$

如果$\alpha_{\substack{new\\1}}$,$\alpha_{\substack{new\\2}}$,同??足0<$\alpha_{\substack{new\\i}}$<C,?$b_{\substack{new\\1}} = b_{\substack{new\\2}}$
如果$\alpha_{\substack{new\\1}}$,$\alpha_{\substack{new\\2}}$,是0或C??取中点?$b_{new}$
?量的???

> 1. 称第一个?量的???外?循??在外?循?中?取?反KKT条件最?重的?本点
> 2. 称第二个?量的???内?循??第二个?量的???准是希望能使α_2有足?大的?化?使得$|E_1 - E_2|$最大?
> 3. ?算更新??b和?差E

四.???程?

任?一?分?用?性SVM和高斯核SVM???数据?行分?

1. **利用SMO算法??SVM**
   参考??原理的???程?
   ?里用?化版SMO?跳??找最佳α?的步?

```python
def svmTrain_SMO(X, y, C, kernelFunction='linear', tol=1e-3, max_iter=5, **kargs):
    """
    ??????SMO????SVM
    ????????????

    ???
    X, y?loadData??????
    C?????
    kernelFunction???????, ????????????????????K
    tol????
    max_iter???????

    ???
    model['kernelFunction']??????
    model['X']?????
    model['y']??????
    model['alpha']??????????
    model['w'], model['b']?????
    """

    start = time.clock()

    m,n = X.shape
    X = np.mat(X)
    y = np.mat(y, dtype='float64')
    #?????-1??
    y[np.where(y==0)] = -1

    alphas = np.mat(np.zeros((m,1)))
    b = 0.0
    E = np.mat(np.zeros((m,1)))
    iters = 0
    eta = 0.0
    L = 0.0
    H = 0.0

    if kernelFunction =='linear':
        K = X*X.T
    elif kernelFunction == 'gaussian':
        K = kargs['K_matrix']
    else :
        print('Kernel Error')
        return None


    print('Training ...', end='')
```

```
dots = 12
while iters < max_iter:
    # ????
    num_changed_alphas = 0
    for i in range(m):
        # ?? E_i
        E[i] = b + np.sum(np.multiply(np.multiply(alphas, y), K[:,i])) - y[i]
        # ????KKT??
        if (y[i]*E[i] < -tol and alphas[i] < C) or (y[i]*E[i] > tol and alphas[i] > 0):
            #???????????α
            j = np.random.randint(m)
            while j == i:
                j = np.random.randint(m)
            # ??E_j
            E[j] = b + np.sum(np.multiply(np.multiply(alphas, y), K[:,j])) - y[j]

            # ?? alpha^old
            alpha_i_old = alphas[i].copy()
            alpha_j_old = alphas[j].copy()

            # ??L,H
            if y[i] == y[j]:
                L = max(0, alphas[j] + alphas[i] - C)
                H = min(C, alphas[j] + alphas[i])
            else:
                L = max(0, alphas[j] - alphas[i])
                H = min(C, C + alphas[j] - alphas[i])

            if L == H:
                continue

            eta = 2*K[i,j] - K[i,i] -K[j,j]
            if eta >= 0:
                continue
            # ??alpha^new_j
            alphas[j] = alphas[j] - (y[j]*(E[i] - E[j]))/eta

            alphas[j] = min(H, alphas[j])
            alphas[j] = max(L, alphas[j])

            # ?????????????????????????????????????
            if abs(alphas[j] - alpha_j_old) < tol:
                alphas[j] = alpha_j_old
                continue
            # ??alpha^new_i
            alphas[i] = alphas[i] + y[i]*y[j]*(alpha_j_old - alphas[j])
```

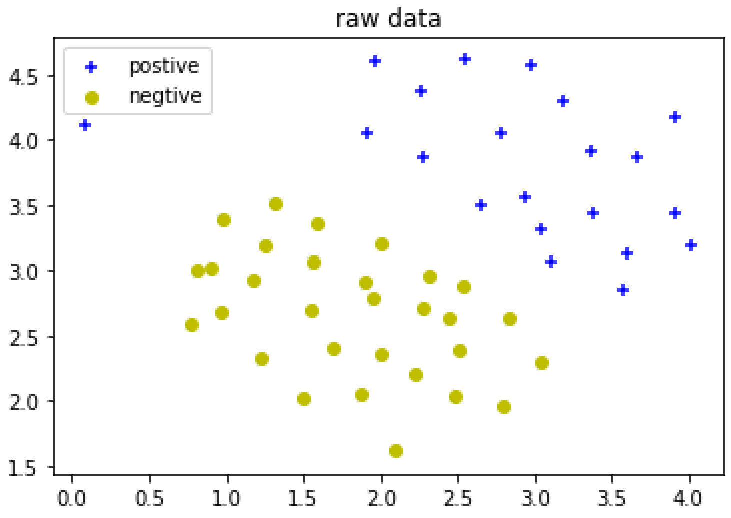
```
                # ?????b
                b1 = b - E[i]\
                 - y[i] * (alphas[i] - alpha_i_old) *  K[i,j]\
                 - y[j] * (alphas[j] - alpha_j_old) *  K[i,j]

                b2 = b - E[j]\
                 - y[i] * (alphas[i] - alpha_i_old) *  K[i,j]\
                 - y[j] * (alphas[j] - alpha_j_old) *  K[j,j]

                if (0 < alphas[i] and alphas[i] < C):
                    b = b1
                elif (0 < alphas[j] and alphas[j] < C):
                    b = b2
                else:
                    b = (b1+b2)/2.0

                num_changed_alphas = num_changed_alphas + 1
        # ?????????α?????????
        if num_changed_alphas == 0:
            iters = iters + 1
        else:
            iters = 0

        print('.', end='')
        dots = dots + 1
        if dots > 78:
            dots = 0
            print()

    print('Done',end='')
    end = time.clock()
    print('( '+str(end-start)+'s )')
    print()

    idx = np.where(alphas > 0)
    #????
    model = {'X':X[idx[0],:], 'y':y[idx], 'kernelFunction':str(kernelFunction), \
             'b':b, 'alphas':alphas[idx], 'w':(np.multiply(alphas,y).T*X).T}
    return model
```

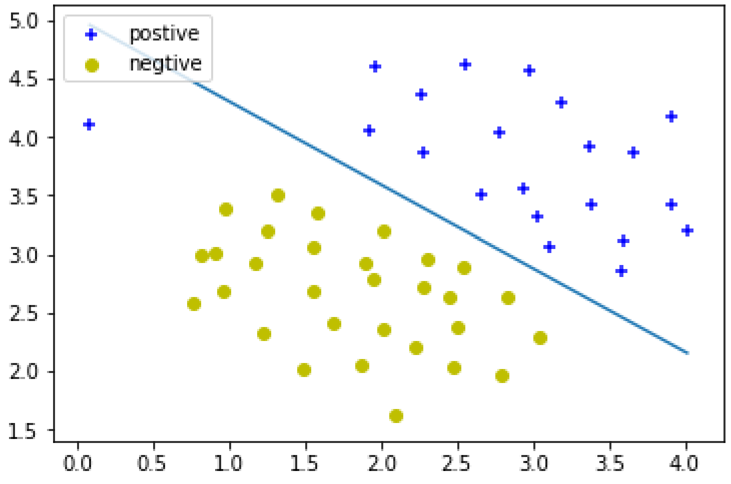
2. **plot**

   linear kernel

```
X,y = SVM_Functions.loadData('task1_linear.mat')
SVM_Functions.plotData(X,y,title='raw data')
model = SVM_Functions.svmTrain_SMO(X,
            y, 1 , kernelFunction='linear', tol=1e-3, max_iter=5)
SVM_Functions.visualizeBoundaryLinear(X,y,model)
```

得?果

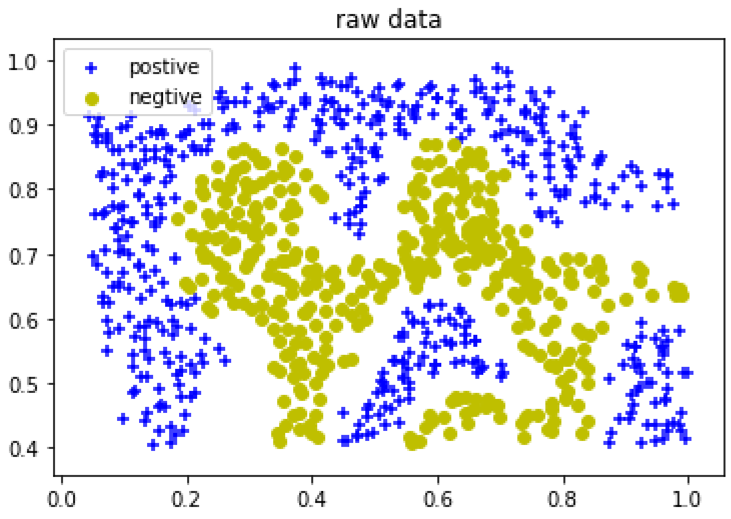


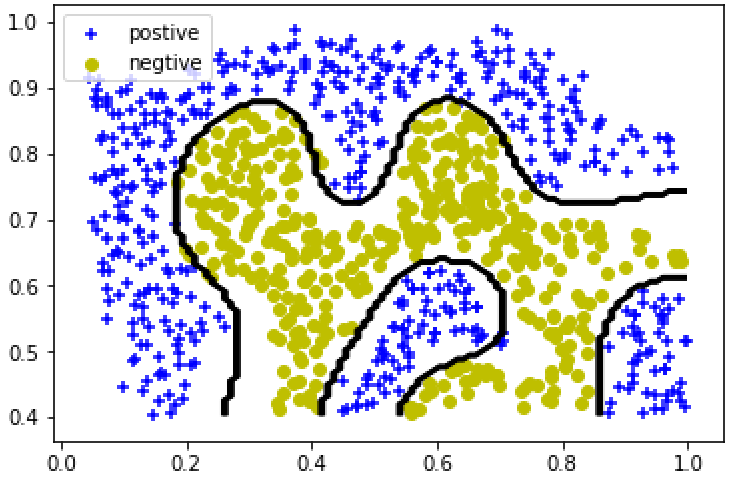


gaussian kernel

```
X1,y1 = SVM_Functions.loadData('task1_gaussian.mat')
SVM_Functions.plotData(X1,y1,title='raw data')
model_1 = SVM_Functions.svmTrain_SMO(X1,
            y1, 1 , kernelFunction='gaussian',
            K_matrix=SVM_Functions.gaussianKernel(X1,sigma=0.1))
SVM_Functions.visualizeBoundaryGaussian(X1,y1,model_1,sigma=0.1)
```

得?果





任?二?使用高斯核SVM??定数据集?行分?

?程?利用任?一中得svmTrain_SMO()?通?迭代来改?C?sigma?来?行分???得精度

```
#?????
def accu(a,b):
    count = 0
    a = np.array(a)
    b = np.array(b)
    for  i in range(a.shape[0]):
        if a[i] == b[i]:
            count+=1
    return count/a.shape[0]
```

```
X,y = SVMF.loadData('task2.mat')
acc=[]
for C in [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]:
    for sigma in [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]:
        model = SVMF.svmTrain_SMO(X,
                y, C , kernelFunction='gaussian',
                K_matrix=SVM_Functions.gaussianKernel(X,sigma))
        y_pred = SVMF.svmPredict(model, X, sigma)
        acc.extend[accu(y_pred,y]
        SVMF.visualizeBoundaryGaussian(X,y,model,sigma)
```

得?果

sigma	0.01	0.03	0.1	0.3	1	3	10	30
C								
0.01	0.49763033	0.49763033	0.50236967	0.49763033	0.81990521	0.8056872	0.50236967	0.49763033
0.03	0.50236967	0.49763033	0.49763033	0.86729858	0.50236967	0.50236967	0.49763033	0.50236967
0.1	0.50236967	0.49763033	0.9478673	0.86255924	0.83886256	0.82464455	0.50236967	0.50236967
0.3	0.50236967	0.97630332	0.9478673	0.90521327	0.87203791	0.71563981	0.81042654	0.49763033
1	1	0.99526066	0.9478673	0.93364929	0.90521327	0.84834123	0.78672986	0.50236967
3	1	1	0.94312796	0.94312796	0.92890995	0.87203791	0.80094787	0.50236967
10	1	1	0.96208531	0.93838863	0.91943128	0.90047393	0.84834123	0.50236967
30	1	1	0.9478673	0.94312796	0.91943128	0.91469194	0.8436019	0.81990521

任?三?使用?性SVM???垃圾?件分?

?程?先?入??集?划分???集和??集?在??集上??SVM????果并??

1. ?入??集

```
train_feat ,labels = SVMF.loadData('task3_train.mat')
print('number of sample:{}'.format(train_feat.shape[0]))
print('dims of sample:{}'.format(train_feat.shape[1]))
```

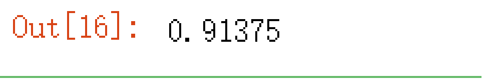
2. 划分数据集

```
num_train = int(0.8 * train_feat.shape[0]) # ????????????
indices = np.arange( train_feat.shape[0])
np.random.shuffle(indices)  # shuffle ??
train_indices = indices[:num_train]
valid_indices = indices[num_train:]
train_data = train_feat[train_indices,:]
train_label = labels[train_indices,:]
valid_data = train_feat[valid_indices,:]
valid_label = labels[valid_indices,:]
```

3. ??

```
model = SVMF.svmTrain_SMO(train_data,
                train_label, C=1 , kernelFunction='linear')
y_pred = SVMF.svmPredict(model, valid_data)
accu(y_pred,valid_label)
```

得??精度



```
Out[16]:  0.91375
```

4. ?入??集

```
from scipy.io import loadmat
def loadData_1(filename):
    """
    ??:
        ?????
    ??:
        numpy.array???X, y??array
        X?m×n???array, m????, n?????
        y?m×1???array, 1????, 0????
    """

    dataDict = loadmat(filename)

    return dataDict['X']
test_feat = loadData_1('task3_test.mat')
```

5. ??

```
ans = SVMF.svmPredict(model, test_feat)
np.savetxt('ans.txt',ans,fmt='%d')
```

五.?????

通??次???我更加理解了SVM相关理?及其?化??方法?