寒假第四周读书报告 2.10-2.16

61518426 周之遥

·读书进度

统计学习方法完成第9章：EM算法及其推广

在所求的模型中有变量无法观测，即“无法判断选择了哪一枚硬币”时使用。EM算法的核心是Q函数的求解与极大化。Q函数是完全数据的对数似然函数关于给定参数情况下隐变量对观测数据的后向概率的期望。

EM算法可用于高斯混合分布。

EM算法可用于无监督学习，隐变量即为数据未标注的部分。

EM算法可解释为F函数的极大-极大，在实际使用中，在每一轮可以不求Q函数的极大化，满足Q函数增大即可或极大参数的某一维（其他维固定）。

·提出问题

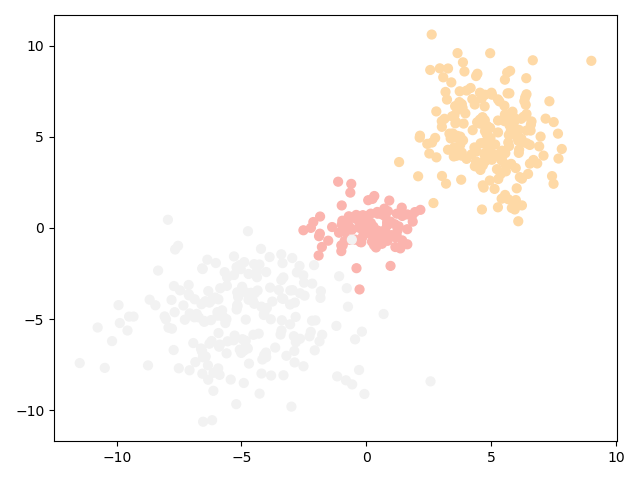
·解决问题

·代码实现

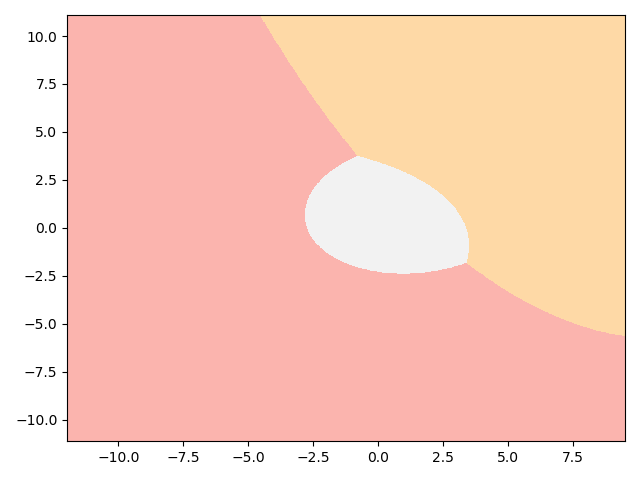
1. GMM模型

**import** matplotlib.pyplot **as** plt  
**from** sklearn.mixture **import** GaussianMixture  
**import** numpy **as** np  
**import** time  
  
start=time.clock()  
  
*#产生三个高斯分布的数据*mean1 = [0, 0]  
cov1 = [[1, 0], [0, 1]]  
x, y = np.random.multivariate\_normal(mean1, cov1, 100).T  
  
mean2 = [5, 5]  
cov2 = [[2, 0], [0, 3]]  
x1, y1 = np.random.multivariate\_normal(mean2, cov2, 200).T  
x=np.concatenate([x,x1],0)  
y=np.concatenate([y,y1],0)  
  
mean3 = [-5, -5]  
cov3 = [[5, 0], [0, 5]]  
x1, y1 = np.random.multivariate\_normal(mean3, cov3, 200).T  
x=np.concatenate([x,x1],0)  
y=np.concatenate([y,y1],0)  
a=np.array([0 **for** \_ **in** range(100)])  
b=np.array([1 **for** \_ **in** range(200)])  
c=np.array([2 **for** \_ **in** range(200)])  
labels=np.r\_[a,b,c] *#labels————数据产生于哪一个高斯分布*X=np.c\_[x,y] *#X————数据点集  
  
#画出原始点*plt.scatter(X[:,0], X[:,1], c=labels, s=40, cmap=plt.cm.Pastel1);  
plt.figure()  
plt.show()  
  
*#使用GMM模型拟合数据。事实上是一种无监督学习*gmm=GaussianMixture(n\_components=3).fit(X)  
classify = gmm.predict(X) *#预测分类结果*print(**'权重：'**,gmm.weights\_)  
print(**'均值：'**,gmm.means\_)  
print(**'协方差：'**,gmm.covariances\_ )  
*#画出聚类结果*plt.scatter(x, y, c=classify, s=40, cmap=plt.cm.Pastel1);  
x\_min, x\_max = X[:, 0].min() - .5, X[:, 0].max() + .5  
y\_min, y\_max = X[:, 1].min() - .5, X[:, 1].max() + .5  
xx, yy = np.meshgrid(np.arange(x\_min, x\_max, .02), np.arange(y\_min, y\_max, .02))  
Z1 = gmm.predict(np.c\_[xx.ravel(), yy.ravel()])  
Z1 = Z1.reshape(xx.shape)  
plt.pcolormesh(xx, yy, Z1, cmap=plt.cm.Pastel1)  
plt.show()  
plt.figure()  
  
*#可视化生成的概率模型*x\_min, x\_max = X[:, 0].min() - .5, X[:, 0].max() + .5  
y\_min, y\_max = X[:, 1].min() - .5, X[:, 1].max() + .5  
xx, yy = np.meshgrid(np.arange(x\_min, x\_max, .02), np.arange(y\_min, y\_max, .02))  
Z = gmm.score\_samples(np.c\_[xx.ravel(), yy.ravel()])  
Z= 2\*\*Z  
Z = Z.reshape(xx.shape)  
plt.pcolormesh(xx, yy, Z, cmap=**'Blues'**)  
plt.show()  
  
end=time.clock()  
print(**'Running time: %s Seconds'**%(end-start))

结果：



原始点



聚类结果

权重： [0.40622539 0.39881405 0.19496056]

均值： [[-4.88217997 -4.91434435]

[ 4.91885696 5.12517228]

[ 0.15901886 0.0760412 ]]

协方差： [[[ 5.52503614 0.14405031]

[ 0.14405031 4.69086171]]

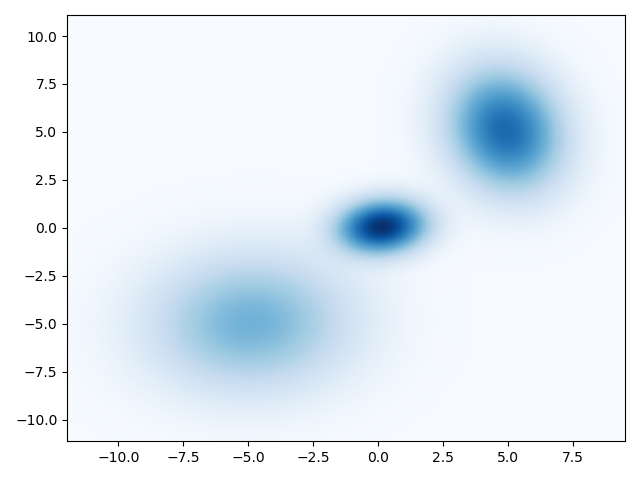
[[ 1.75419291 -0.31661999]

[-0.31661999 3.69233428]]

[[ 1.05601302 0.07902202]

[ 0.07902202 0.71318274]]]

Running time: 2.6423014999999994 Seconds



概率密度图