**GMM模型做分类**

**一、核心思想**

1. 对每个动作建GMM，数量不一定是按照scene来的，而是应该有个学习的过程。

GMM的学习过程：

普通vl\_gmm，人工调每个动作的模型数；

普通vl0gmm，寻找可靠的搜索算法；

**近邻传播算法！Science affinity appropagation**

2. 分类的方法：**到哪个团团近，就算那个团团的class；**

或这到**哪个类的所有团团的总距离近，就算哪个类的**；

或：利用GMM的概率模型，算属于每个c**lass的总概率**；

训练负样本，加入-分策略；

3. improvement: iterator gmm（hierarchy model)

**二、问题**

1、 这个真的能用高斯模型拟合吗？（每个GMM2、4、8，16的类好像都不是特别好）linear 一定符合高斯吗？

2、 PCA是不是可以实验下？（减少co-occurrence）

4、我们训练的样本是不是不够？（回想bag-of-words聚类的过程，提了多少描述子啊）

5、试一下affinity appropagation？

6、聚类的其他算法？（因为有些exp表明，每次准确率浮动的范围非常大，而vl\_gmm的过程是随机的！）

**三、实验**

2014/11/30： 归一化数据本身和GMM模型测试：发现sift+bow（不含spatial，spatial已经归一化过了）下，handclapping的准确率稳定在70%+，但是整体准确率30+

2014/11/30 归一化的dense更差了（EXP2 稳定在20%）EXP3 出现严重的问题：全部会识别成4！！！！

Experiment 1:（SIFT的bag-of-words dim=18结果)

1. 每个动作训练8的GMM（使用vl\_gmm)做，全场景测试，（每个动作训练集：40个(hist\_trainSet) 随机选择，测试集：60个(hist\_testSet）；

2. 判别方法：找最小分布所在的region对应的class

代码：project\_sift\GMM

3. 实验结果（10次平均值）

0.430000000000000

0.405263157894737

0.330000000000000

0.535000000000000

0.404166666666667

avg=0.420885964912281;

归一化距离模型后：

0.215000000000000

0.447368421052632

0.145000000000000

0.445000000000000

0.766666666666667

Avg=0.40

归一化后我们发现，总体准确率有一定的提升，但是部分特别动作类型下降（jogging，很容易被误判成walking）

Experiment 1\_2:（SIFT的bag-of-words dim=18结果 GMM=4)

1. 每个动作训练4的GMM（使用vl\_gmm)做，全场景测试，（每个动作训练集：40个(hist\_trainSet) 随机选择，测试集：60个(hist\_testSet）；

2. 判别方法：找最小分布所在的region对应的class

代码：project\_sift\GMM

3. 实验结果：

0.305000000000000

0.463157894736842

0.345000000000000

0.345000000000000

0.425000000000000

avg=0.376631578947368

归一化距离模型后：

0.150000000000000

0.468421052631579

0.195000000000000

0.465000000000000

0.716666666666667

AVG=0.3990

Experiment 1\_3:（SIFT的bag-of-words dim=18结果 GMM=15)

1. 每个动作训练15的GMM（使用vl\_gmm)做，全场景测试，（每个动作训练集：40个(hist\_trainSet) 随机选择，测试集：60个(hist\_testSet）；

2. 判别方法：找最小分布所在的region对应的class

代码：project\_sift\GMM

3. 实验结果：

0.350000000000000

0.363157894736842

0.295000000000000

0.410000000000000

0.495833333333333

AVG=0.3828

归一化距离模型后：

0.250000000000000

0.484210526315790

0.0450000000000000

0.215000000000000

0.791666666666667

AVG=35.72%

spatial-sift test

Experiment 1\_4: (spatial-bow sift dim=100+ gmm=2,8,15)

. 每个动作训练8的GMM（使用vl\_gmm)做，全场景测试，（每个动作训练集：40个(hist\_trainSet) 随机选择，测试集：60个(hist\_testSet）；

2. 判别方法：找最小分布所在的region对应的class

代码：project\_sift\GMM

3. 实验结果：(gmm=8)

0.325000000000000

0.357894736842105

0.295000000000000

0.540000000000000

0.208333333333333

AVG=34.52

(实验发现 GMM=1: 25% GMM=4：AVG=38.8(highest)，GMM=15：33.32的结果和该结果差不多）

归一化之后：

0.385000000000000

0.694736842105263

0.270000000000000

0.205000000000000

0.0750000000000000

AVG=32.59%

===========================================================================================================================

对dense trajectory做。

Experiment 2(dense trajectory+fisher vector test gmm=8)

. 每个动作训练8的GMM（使用vl\_gmm)做，全场景测试，（每个动作训练集：40个(hist\_trainSet) 随机选择，测试集：60个(hist\_testSet）；

2. 判别方法：找最小分布所在的region对应的class

代码：DenseTrajectory\GMM

GMM=8:

0.193333333333333

0.345762711864407

0.300000000000000

0.258333333333333

0.170000000000000

AVG=0.2535

归一化：

0.0916666666666667

0.193220338983051

0.356666666666667

0.255000000000000

0.296666666666667

AVG=0.2386

GMM=15:

0.166666666666667

0.227118644067797

0.251666666666667

0.351666666666667

0.271666666666667

avg=0.2538

归一化：

0.138333333333333

0.208474576271186

0.258333333333333

0.380000000000000

0.276666666666667

AVG=0.2524

GMM=4:

0.205000000000000

0.249152542372881

0.296666666666667

0.221666666666667

0.191666666666667

AVG=0.2328

归一化：

0.196666666666667

0.223728813559322

0.263333333333333

0.231666666666667

0.165000000000000

AVG=0.21

============================================================================================================================

策略改进实验，到每个class的距离和最小！（在dense+归一化的情况下，好像对class的数量特别敏感）

============================================================================================================================

对dense trajectory做。

Experiment 3(dense trajector+fv improved test)

. 每个动作训练8的GMM（使用vl\_gmm)做，全场景测试，（每个动作训练集：40个(hist\_trainSet) 随机选择，测试集：60个(hist\_testSet）；

2. 判别方法：找到分布距离之和最小的对应的class

代码：DenseTrajectory\GMM

GMM=2(running的效果最差）

归一化后much worser！（walking经常被认为是jogging和running）

归一化后：

0.161666666666667

0.303389830508475

0.211666666666667

0.501666666666667

0

AVG=0.2357

GMM=4(特殊现象：虽然10次的avg不理想，但是有些时候我们的结果会比较好：over85%，看来，我们每次实验的结果和聚类密切相关）

0.0516666666666667

0.418644067796610

0.326666666666667

0.0283333333333333

0.341666666666667

AVG=28左右；

归一化后：（running经常被错误检测成jogging）

0.423333333333333

0.188135593220339

0.420000000000000

0.00333333333333333

0.0550000000000000

0.2108

GMM=8：

0.0666666666666667

0.0779661016949153

0.580000000000000

0

0.401666666666667

AVG=0.225259887005650

归一化后：（walking的检测大部分出奇的好，但是其他动作也会检测成walking）

0.0133333333333333

0.0661016949152542

0.298333333333333

0.0683333333333333

0.618333333333333

Avg=0.2129