

crazy_sparrow的专栏

目录视图

摘要视图

RSS 订阅

学院APP首次下载，可得
代码英才

试读一增长黑客，创业公司必知的“黑科技” 免费学技术，就是如此任性 CSDN 诚征

Opencv学习笔记（十）高斯混合模型

标签: matlab function statistics comments 算法 测试

2012-03-30 21:28

24548人阅读

评论(99)

收藏 举报

分类: OpenCV (12)

版权声明：本文为博主原创文章，未经博主允许不得转载。

原创文章，转载请注明：http://blog.csdn.net/crazy_sparrow/article/details/7413019

好吧，我承认这个题目有点噱头，其实本文要讲的一般的高斯混合模型，基于matlab实现，没有涉及到opencv。之所以作为opencv的学习笔记之一是因为之后打算讲下基于高斯混合模型的背景建模（实现目标跟踪），所以把这个也放上来了。

混合高斯模型的原理说白了就一句话：任意形状的概率分布都可以用多个高斯分布函数去近似。换句话说，高斯混合模型（GMM）可以平滑任意形状的概率分布。其参数求解方法一般使用极大似然估计求解，但使用极大似然估计法往往不能获得完整数据（比如样本已知，但样本类别（属于哪个高斯分布）未知），于是出现了EM（最大期望值）求解方法。

虽然上面说的简单，但是混合高斯模型和EM求解的理论还是比较复杂的，我把我所找到的我认为能够快速掌握高斯混合模型的资料打包到了附件中，大家可以去下载，了解混合高斯模型以及EM的完整推导过程。

附件下载地址：

http://download.csdn.net/detail/crazy_sparrow/4187994

大致抽取下高斯混合模型的重要概念：

1) 任意数据分布可用高斯混合模型（M个单高斯）表示（（1）式）

$$p(\mathbf{x}_i) = \sum_{j=1}^M \alpha_j N_j(\mathbf{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \quad (1)$$

其中：

$$\sum_{j=1}^M \alpha_j = 1, \quad (2)$$

$$N_j(\mathbf{x}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) = \frac{1}{\sqrt{(2\pi)^m |\boldsymbol{\Sigma}_j|}} \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) \right]$$

(3) 表示第j个高斯混合模型

2) N个样本集X的log似然函数如下：

$$\begin{aligned} l(X|\Theta) &= \log \prod_{i=1}^N \sum_{j=1}^M \alpha_j N_j(\mathbf{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \\ &= \sum_{i=1}^N \log \sum_{j=1}^M \alpha_j N_j(\mathbf{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \end{aligned} \quad (4)$$

其中：

参数: $\theta_j = (\alpha_j, \mu_j, \Sigma_j)$, $\Theta = (\theta_1, \dots, \theta_M)^T$ (5)

下面具体讲讲基于**EM**迭代的混合高斯模型参数求解算法流程:

1) 初始参数由**k-means** (其实也是一种特殊的高斯混合模型) 决定:

[plain]

```
01. function [Priors, Mu, Sigma] = EM_init_kmeans(Data, nbStates)
02. % Inputs -----
03. %   o Data:      D x N array representing N datapoints of D dimensions.
04. %   o nbStates:  Number K of GMM components.
05. % Outputs -----
06. %   o Priors:     1 x K array representing the prior probabilities of the
07. %                K GMM components.
08. %   o Mu:        D x K array representing the centers of the K GMM components.
09. %   o Sigma:     D x D x K array representing the covariance matrices of the
10. %                K GMM components.
11. % Comments -----
12. %   o This function uses the 'kmeans' function from the MATLAB Statistics
13. %     toolbox. If you are using a version of the 'netlab' toolbox that also
14. %     uses a function named 'kmeans', please rename the netlab function to
15. %     'kmeans_netlab.m' to avoid conflicts.
16.
17. [nbVar, nbData] = size(Data);
18.
19. %Use of the 'kmeans' function from the MATLAB Statistics toolbox
20. [Data_id, Centers] = kmeans(Data', nbStates);
21. Mu = Centers';
22. for i=1:nbStates
23.     idtmp = find(Data_id==i);
24.     Priors(i) = length(idtmp);
25.     Sigma(:,:,i) = cov([Data(:,idtmp) Data(:,idtmp)]');
26.     %Add a tiny variance to avoid numerical instability
27.     Sigma(:,:,i) = Sigma(:,:,i) + 1E-5.*diag(ones(nbVar,1));
28. end
29. Priors = Priors ./ sum(Priors);
```

[plain]

```
01. function [Priors, Mu, Sigma] = EM_init_kmeans(Data, nbStates)
02. % Inputs -----
03. %   o Data:      D x N array representing N datapoints of D dimensions.
04. %   o nbStates:  Number K of GMM components.
05. % Outputs -----
06. %   o Priors:     1 x K array representing the prior probabilities of the
07. %                K GMM components.
08. %   o Mu:        D x K array representing the centers of the K GMM components.
09. %   o Sigma:     D x D x K array representing the covariance matrices of the
10. %                K GMM components.
11. % Comments -----
12. %   o This function uses the 'kmeans' function from the MATLAB Statistics
13. %     toolbox. If you are using a version of the 'netlab' toolbox that also
14. %     uses a function named 'kmeans', please rename the netlab function to
15. %     'kmeans_netlab.m' to avoid conflicts.
16.
17. [nbVar, nbData] = size(Data);
18.
19. %Use of the 'kmeans' function from the MATLAB Statistics toolbox
20. [Data_id, Centers] = kmeans(Data', nbStates);
21. Mu = Centers';
22. for i=1:nbStates
23.     idtmp = find(Data_id==i);
24.     Priors(i) = length(idtmp);
25.     Sigma(:,:,i) = cov([Data(:,idtmp) Data(:,idtmp)]');
26.     %Add a tiny variance to avoid numerical instability
27.     Sigma(:,:,i) = Sigma(:,:,i) + 1E-5.*diag(ones(nbVar,1));
28. end
29. Priors = Priors ./ sum(Priors);
```

2) **E**步 (求期望)

求取:

$$\beta_j = E(\alpha_j | \mathbf{x}_i; \Theta) = \frac{\alpha_j N_j(\mathbf{x}_i; \Theta)}{\sum_l \alpha_l N_l(\mathbf{x}_i; \Theta)}, \quad 1 \leq i \leq n, \quad 1 \leq j \leq M \quad (7)$$

其实，这里最贴切的式子应该是log似然函数的期望表达式

事实上，3）中参数的求取也是用log似然函数的期望表达式求偏导等于0解得的简化后的式子，而这些式子至于（7）式有关，于是E步可以只求（7）式，以此简化计算，不需要每次都求偏导了。

[plain]

```
01. %% E-step %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
02. for i=1:nbStates
03.     %Compute probability p(x|i)
04.     Pxi(:,i) = gaussPDF(Data, Mu(:,i), Sigma(:,i));
05. end
06. %Compute posterior probability p(i|x)
07. Pix_tmp = repmat(Priors,[nbData 1]).*Pxi;
08. Pix = Pix_tmp ./ repmat(sum(Pix_tmp,2),[1 nbStates]);
09. %Compute cumulated posterior probability
10. E = sum(Pix);
```

[plain]

```
01. %% E-step %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
02. for i=1:nbStates
03.     %Compute probability p(x|i)
04.     Pxi(:,i) = gaussPDF(Data, Mu(:,i), Sigma(:,i));
05. end
06. %Compute posterior probability p(i|x)
07. Pix_tmp = repmat(Priors,[nbData 1]).*Pxi;
08. Pix = Pix_tmp ./ repmat(sum(Pix_tmp,2),[1 nbStates]);
09. %Compute cumulated posterior probability
10. E = sum(Pix);
```

[plain]

```
01. function prob = gaussPDF(Data, Mu, Sigma)
02. % Inputs -----
03. %   o Data: D x N array representing N datapoints of D dimensions.
04. %   o Mu:   D x K array representing the centers of the K GMM components.
05. %   o Sigma: D x D x K array representing the covariance matrices of the
06. %             K GMM components.
07. % Outputs -----
08. %   o prob: 1 x N array representing the probabilities for the
09. %             N datapoints.
10.
11. [nbVar,nbData] = size(Data);
12.
13. Data = Data' - repmat(Mu',nbData,1);
14. prob = sum((Data*inv(Sigma)).*Data, 2);
15. prob = exp(-0.5*prob) / sqrt((2*pi)^nbVar * (abs(det(Sigma))+realmin));
```

[plain]

```
01. function prob = gaussPDF(Data, Mu, Sigma)
02. % Inputs -----
03. %   o Data: D x N array representing N datapoints of D dimensions.
04. %   o Mu:   D x K array representing the centers of the K GMM components.
05. %   o Sigma: D x D x K array representing the covariance matrices of the
06. %             K GMM components.
07. % Outputs -----
08. %   o prob: 1 x N array representing the probabilities for the
09. %             N datapoints.
10.
11. [nbVar,nbData] = size(Data);
12.
13. Data = Data' - repmat(Mu',nbData,1);
14. prob = sum((Data*inv(Sigma)).*Data, 2);
15. prob = exp(-0.5*prob) / sqrt((2*pi)^nbVar * (abs(det(Sigma))+realmin));
```

3）M步（最大化步骤）

更新权值：

$$\alpha'_j = \frac{\sum_{i=1}^N \beta_{ij}}{N}$$

更新均值：

$$\mu'_j = \frac{\sum_{i=1}^n \beta_{ij} x_i}{\sum_{i=1}^n \beta_{ij}}$$

更新方差矩阵：

$$\Sigma'_j = \frac{\sum_{i=1}^n \beta_{ij} (x_i - \mu'_j)(x_i - \mu'_j)^T}{\sum_{i=1}^n \beta_{ij}}$$

[cpp]

```
01. %% M-step %%%%%%%%%%%%%%
02. for i=1:nbStates
03.     %Update the priors
04.     Priors(i) = E(i) / nbData;
05.     %Update the centers
06.     Mu(:,i) = Data*Pix(:,i) / E(i);
07.     %Update the covariance matrices
08.     Data_tmp1 = Data - repmat(Mu(:,i),1,nbData);
09.     Sigma(:,i) = (repmat(Pix(:,i)',nbVar, 1) .* Data_tmp1*Data_tmp1') / E(i);
10.     %% Add a tiny variance to avoid numerical instability
11.     Sigma(:,i) = Sigma(:,i) + 1E-5.*diag(ones(nbVar,1));
12. end
```

[cpp]

```
01. %% M-step %%%%%%%%%%%%%%
02. for i=1:nbStates
03.     %Update the priors
04.     Priors(i) = E(i) / nbData;
05.     %Update the centers
06.     Mu(:,i) = Data*Pix(:,i) / E(i);
07.     %Update the covariance matrices
08.     Data_tmp1 = Data - repmat(Mu(:,i),1,nbData);
09.     Sigma(:,i) = (repmat(Pix(:,i)',nbVar, 1) .* Data_tmp1*Data_tmp1') / E(i);
10.     %% Add a tiny variance to avoid numerical instability
11.     Sigma(:,i) = Sigma(:,i) + 1E-5.*diag(ones(nbVar,1));
12. end
```

4) 截止条件

不断迭代EM步骤，更新参数，直到似然函数前后差值小于一个阈值，或者参数前后之间的差（一般选择欧式距离）小于某个阈值，终止迭代，这里选择前者。附上结合EM步骤的代码：

[cpp]

```
01. while 1
02.     %% E-step %%%%%%%%%%%%%%
03.     for i=1:nbStates
04.         %Compute probability p(x|i)
05.         Pxi(:,i) = gaussPDF(Data, Mu(:,i), Sigma(:,i));
06.     end
07.     %Compute posterior probability p(i|x)
08.     Pix_tmp = repmat(Priors,[nbData 1]).*Pxi;
09.     Pix = Pix_tmp ./ repmat(sum(Pix_tmp,2),[1 nbStates]);
10.     %Compute cumulated posterior probability
11.     E = sum(Pix);
12.     %% M-step %%%%%%%%%%%%%%
13.     for i=1:nbStates
14.         %Update the priors
15.         Priors(i) = E(i) / nbData;
16.         %Update the centers
17.         Mu(:,i) = Data*Pix(:,i) / E(i);
18.         %Update the covariance matrices
19.         Data_tmp1 = Data - repmat(Mu(:,i),1,nbData);
20.         Sigma(:,i) = (repmat(Pix(:,i)',nbVar, 1) .* Data_tmp1*Data_tmp1') / E(i);
21.         %% Add a tiny variance to avoid numerical instability
22.         Sigma(:,i) = Sigma(:,i) + 1E-5.*diag(ones(nbVar,1));
23.     end
24.     %% Stopping criterion %%%%%%%%%%%%%%
25.     for i=1:nbStates
```

```

26.     %Compute the new probability p(x|i)
27.     Pxi(:,i) = gaussPDF(Data, Mu(:,i), Sigma(:,i,i));
28. end
29. %Compute the log likelihood
30. F = Pxi*Priors';
31. F(find(F<realmin)) = realmin;
32. loglik = mean(log(F));
33. %Stop the process depending on the increase of the log likelihood
34. if abs((loglik/loglik_old)-1) < loglik_threshold
35.     break;
36. end
37. loglik_old = loglik;
38. nbStep = nbStep+1;
39. end

```

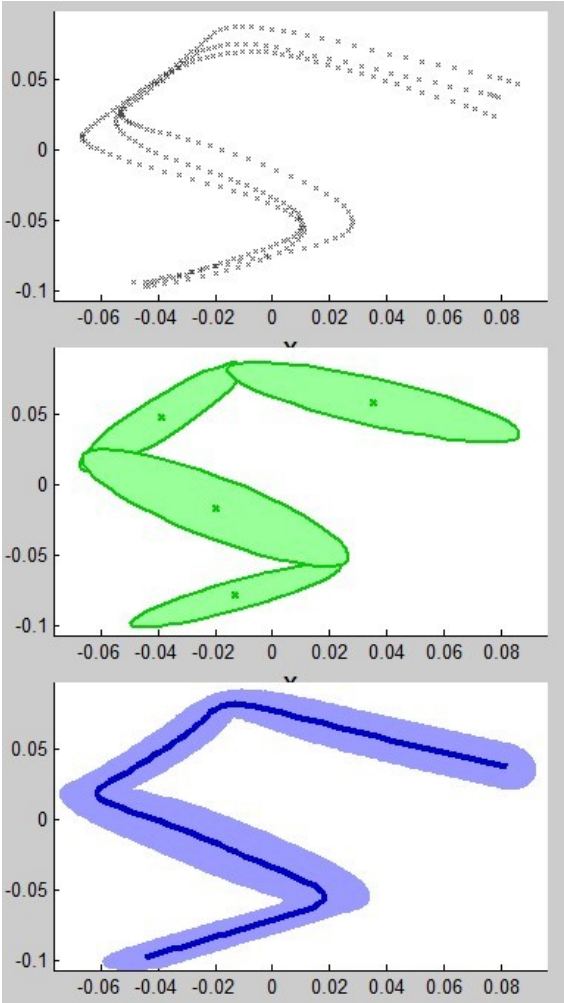
[cpp]

```

01. while 1
02.     %% E-step %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
03.     for i=1:nbStates
04.         %Compute probability p(x|i)
05.         Pxi(:,i) = gaussPDF(Data, Mu(:,i), Sigma(:,i,i));
06.     end
07.     %Compute posterior probability p(i|x)
08.     Pix_tmp = repmat(Priors,[nbData 1]).*Pxi;
09.     Pix = Pix_tmp ./ repmat(sum(Pix_tmp,2),[1 nbStates]);
10.     %Compute cumulated posterior probability
11.     E = sum(Pix);
12.     %% M-step %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
13.     for i=1:nbStates
14.         %Update the priors
15.         Priors(i) = E(i) / nbData;
16.         %Update the centers
17.         Mu(:,i) = Data*Pix(:,i) / E(i);
18.         %Update the covariance matrices
19.         Data_tmp1 = Data - repmat(Mu(:,i),1,nbData);
20.         Sigma(:,i,i) = (repmat(Pix(:,i)',nbVar, 1) .* Data_tmp1*Data_tmp1') / E(i);
21.         %% Add a tiny variance to avoid numerical instability
22.         Sigma(:,i,i) = Sigma(:,i,i) + 1E-5.*diag(ones(nbVar,1));
23.     end
24.     %% Stopping criterion %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
25.     for i=1:nbStates
26.         %Compute the new probability p(x|i)
27.         Pxi(:,i) = gaussPDF(Data, Mu(:,i), Sigma(:,i,i));
28.     end
29.     %Compute the log likelihood
30.     F = Pxi*Priors';
31.     F(find(F<realmin)) = realmin;
32.     loglik = mean(log(F));
33.     %Stop the process depending on the increase of the log likelihood
34.     if abs((loglik/loglik_old)-1) < loglik_threshold
35.         break;
36.     end
37.     loglik_old = loglik;
38.     nbStep = nbStep+1;
39. end

```

结果测试（第一幅为样本点集，第二幅表示求取的高斯混合模型，第三幅为回归模型）：



忘了说了，传统的GMM和k-means一样，需要给定K值（即:要有几个高斯函数）。

上一篇 [c++中const与c语言当中的区别zz](#)

下一篇 [Opencv学习笔记（十一）目标跟踪](#)

顶

17

踩

0

我的同类文章

OpenCV（12）
更多

猜你在找

- JAVA性能测试项目实战之真实OA系统【小强测试出品】
- MySQL入门到精通（偏性能调优方向）【小强测试出品】
- Jmeter性能测试全程实战
- 互联网单元测试零基础
- 移动APP测试基础到进阶
- JAVA性能测试项目实战之真实OA系统【小强测试出品】

- MySQL入门到精通（偏性能调优方向）【小强测试出品】
- Jmeter性能测试全程实战
- 互联网单元测试零基础
- 移动APP测试基础到进阶


查看评论

暂无评论


您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

个人资料



crzy_sparrow



访问：306763次

积分：3274

等级：

BLOG

5

排名：第5620名

原创：46篇


转载：25篇

译文：0篇

评论：308条

文章搜索


博客专栏



内存管理

文章：3篇

阅读：14527



Opencv学习笔记

文章：11篇

阅读：196138

文章分类

android(3)

file:///F:/Dropbox/share%20(1)/GMM/Opencv%E5%AD%A6%E4%B9%A0%E7%AC%94%E8%AE%B0%EF%BC%88%E5%8D%81%EF%BC%89%E9... 7/10

linux(8)
linux驱动分析(1)
QT开发(7)
嵌入式开发(0)
图像处理(6)
达芬奇架构(5)
生活点滴(9)
琐碎(5)
面试相关(13)
OpenCV(13)
LDD3读书笔记(2)
计算广告学(0)
项目工具(1)
Hadoop,Storm,Hbase,Rainbow,Pig
Web开发(0)

文章存档
2013年11月(2)
2012年05月(1)
2012年04月(16)
2012年03月(20)
2012年02月(4)
展开

阅读排行
Opencv学习笔记（五）Harris角点检测(47963)
Opencv学习笔记（九）光流法(39998)
Opencv学习笔记（六）SURF学习笔记(30179)
Opencv学习笔记（十）高斯混合模型(24543)
Opencv学习笔记（十一）目标跟踪(19504)
Opencv学习笔记（二）meanshift之我见(11340)
初窥内存管理（三）伙伴算法(8242)
Opencv学习笔记（四）霍夫变换(6789)
canny算子四部曲之一（高斯滤波）(6204)
Opencv学习笔记（一）Ubuntu + QT + Opencv环境搭建(5845)

评论排行
Opencv学习笔记（十）高斯混合模型(99)
Opencv学习笔记（九）光流法(43)
Opencv学习笔记（十一）目标跟踪(35)
Opencv学习笔记（五）Harris角点检测(29)
Opencv学习笔记（六）SURF学习笔记(14)
初窥内存管理（三）伙伴算法(10)
新出炉作息时间表，必须做到！！!(10)
初窥内存管理（一）(7)
Opencv学习笔记（四）霍夫变换(6)

初窥内存管理（二）(5)

推荐文章

- *Hadoop节点"慢磁盘"监控
- *假如你想成为全栈工程师...
- *没有躲过的坑--正则表达式截取字符串
- *CardView完全解析与RecyclerView结合使用(三十二)
- *And roid 高仿微信发朋友圈浏览图片效果
- *通过Ajax的方式执行GP服务

最新评论

Opencv学习笔记（十）高斯混合模型

ozhan73412: 博主，可以给我发一份高斯混合模型用EM算法实现的完整代码吗？谢谢了，邮箱是：935626315@qq...

Opencv学习笔记（十）高斯混合模型

ozhan73412: 老师您好，看了您的博客写的很有水平，我最近也在学习图像处理的知识，但是在实现mrf中的GMM模型时， ...

Opencv学习笔记（十一）目标跟踪

lsicyzo: 大神 求完整代码 邮箱 1246266859@qq.com 谢谢

Opencv学习笔记（十）高斯混合模型

renyunshamen754: 你好楼主~~新手正在学习这部分内容，能否发一份完整的代码，谢谢~~414982884@qq.com

Opencv学习笔记（五）Harris角点检测

鸢尾0709: @sunjunlishi:改进的和fast算法为啥运行不出来

canny算子四部曲之一（高斯滤波）

jjdbear: 感谢博主，试验过了可以运行，不过一般是以R为参数，这个是先方差再求窗口半径有点不习惯

Opencv学习笔记（十）高斯混合模型

Downloadtheprogram: 我正在学习这个，能把这个程序发个给我吗649674796@qq.com

Opencv学习笔记（十）高斯混合模型

saygoodbyels: 楼主，正在学习，求完整代码。1559914865@qq.com 谢谢

Opencv学习笔记（十）高斯混合模型

归缘憾: 真是感谢楼主分享了，话说你这是c语言的代码吗？能否花点时间给发份代码。邮箱：1016181343@qq...

Opencv学习笔记（十）高斯混合模型

skyng_li: 谢谢分享，我最近也在学习利用高斯模型来提取相应特征，请楼主发个完整代码，jzyslg888@126....

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服

杂志客服

微博客服

webmaster@csdn.net

400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏乐知网络技术有限公司 提供商务支持

京 ICP 证 09002463 号 | Copyright © 1999-2014, CSDN.NET, All Rights Reserved

file:///F:/Dropbox/share%20(1)/GMM/Opencv%E5%AD%A6%E4%B9%A0%E7%AC%94%E8%AE%B0%EF%BC%88%E5%8D%81%EF%BC%89%E9... 9/10

☐