作业 3:用

词袋

概述

这个项目的目标是向你介绍图像识别。具体来说，我们将从非常简单的方法——微小图像和最近邻分类——开始研究场景识别的任务，然后继续介绍更高级的方法——量化局部特征包和支持向量机学习的线性分类器。

词袋模型是一种流行的图像分类技术，其灵感来源于自然语言处理中使用的模型。该模型忽略或淡化单词排列(图像中的空间信息)，并根据视觉单词频率的直方图进行分类。通过对大量局部特征语料库进行聚类，建立视觉单词“词汇表”。有关使用量化特征进行类别识别的更多细节，请参见 Szeliski 第 14.4.1 章。此外，14.3.2 讨论了词汇的创建，14.1 涵盖了分类技术。

对于这个项目，你将实现一个基本的词袋模型，有很多额外学分的机会。您将通过在 15 个场景数据库上进行训练和测试将场景分类为 15 个类别之一(在 Lazebnik 等人 2006 年介绍，尽管建立在先前发布的数据集之上)。**Lazebnik et al. 2006 是一篇非常值得阅读的论文，尽管我们将实现论文中讨论的基线方法(相当于零级金字塔)，而不是更复杂的空间金字塔(这是额外的学分)。**关于词袋模型的预深度学习特征编码方法的优秀综述，请参阅 Chatfield et al, 2011。

15 场景数据集中每个类别的示例场景。数字来自 Lazebnik et al. 2006。

细节和入门代码

这个项目的主要脚本是 student_code_SID.py。ipython notebook 作业 3。Ipynb 运行实现项目所需的函数调用。你的任务是在主脚本中实现这些方法，这些方法将允许你在 notebook 上实现所需的精度。

你需要实现 2 种不同的图像表示——微小的图像和 SIFT 特征的包——和 2 种不同的分类技术——最近邻和线性 SVM。在写作中，你被特别要求报告以下组合的性能，并且也强烈建议你按照这个顺序实现它们:

- 微小图像表示和最近邻分类器(精度约为 18-25%)。
- Bag of SIFT 表示和最近邻分类器(精度约 50-60%)。
- 袋 SIFT 表示和线性 SVM 分类器(精度约 60- 70%)。

你将从实现微小图像表示和最近邻分类器开始。它们易于理解，易于实现，对于我们的实验设置来说运行速度非常快(少于 10 秒)。

“微小图像”功能的灵感来自 Torralba、Fergus 和 Freeman 的同名作品，是可能的最简单的图像表示之一。一个简单地将每张图像调整为一个小的、固定的分辨率(我们推荐 16x16)。如果将微小的图像设置为零均值和单位长度，效果会稍微好一些。这不是一个特别好的表示，因为它丢弃了所有的高频图像内容，对空间或亮度变化不是特别不变。**Torralba，Fergus 和 Freeman 提出了几种对齐方法来缓解后一个缺点，但我们在这个项目中不会担心对齐问题。**我们只是使用微小的图像作为基线。更多细节请参阅起步代码中的 get_tiny_images()。

最近邻分类器同样易于理解。当任务是将测试特征分类到特定类别时，只需找到“最近的”训练示例(L2 距离是一个足够的度量)，并将该最近训练示例的标签分配给测试用例。最近邻分类器有许多理想的特征——它不需要训练，它可以学习任意复杂的决策边界，并且它可以简单地支持多类问题。虽然它很容易受到训练噪声的影响，但可以通过基于 K 个最近邻的投票来缓解(但并不要求你这么做)。随着特征维度的增加，最近邻分类器也会受到影响，因为分类器没有机制来学习哪些维度

与决策无关。对于高维数据和许多训练示例，最近邻计算也会变得缓慢。

详情参见 nearest_neighbor_classify()。

在 15 个场景数据库上，微型图像表示和最近邻分类器将获得 15% ~ 25%的准确率。作为比较，chance 性能约为 7%。

在你实现了基线场景识别管道之后，是时候转向更复杂的图像表示了——量化的 SIFT 特征包。在我们可以将我们的训练和测试图像表示为特征直方图包之前，我们首先需要建立一个视觉单词词汇表。我们将从我们的训练集(10 或 100)中采样许多局部特征，然后用 kmeans 对它们进行聚类，从而形成这个词汇表。kmeans 聚类的数量就是我们词汇量的大小和我们特征的大小。例如，你可能首先将许多 SIFT 描述子聚类成 k=50 个簇。这将连续的、128 维的 SIFT 特征空间划分为 50 个区域。对于我们观察到的任何新的 SIFT 特征，只要我们保存原始聚类的质心，我们就可以弄清楚它属于哪个区域。那些质心就是我们的视觉词汇表。因为采样和聚类许多局部特征的速度可能很慢，启动代码保存了聚类质心，并避免在未来运行时重新计算它们。

更多细节请参见 build_vocabulary()。

现在我们已经准备好将我们的训练和测试图像表示为视觉单词的直方图。对于每个图像，我们将密集采样许多 SIFT 描述符。我们不是存储数百个 SIFT 描述符，而是简单地计算在我们的视觉单词词汇表中，每个簇中有多少个 SIFT 描述符。这是通过为每个 SIFT 特征找到最近邻的 kmeans 质心来实现的。因此，如果我们有一个包含 50 个视觉单词的词汇表，并且我们在一幅图像中检测到 220 个 SIFT 特征，我们的 SIFT 表示包将是一个 50 维的直方图，其中每个箱子统计 SIFT 描述子被分配给该聚类的次数，总和为 220。直方图应该被归一化，这样图像大小不会显著改变特征量级的包。详情参见 get_bags_of_sifts()。

你现在应该测量你的 SIFT 包表示与最近邻分类器配对时的工作情况。SIFT 表示包有许多设计决策和自由参数(聚类数量、采样密度、采样尺度、SIFT 参数等)，因此精度可能在 50%到 60%之间变化。

最后一个任务是训练 1-vs-all 线性 svm，使其在 SIFT 特征空间的包中运行。线性分类器是可能的最简单的学习模型之一。特征空间由学习的超平面和测试用例进行划分

根据它们落在超平面的哪一边进行分类。尽管这个模型的表达能力远不如最近邻分类器，但它通常会表现得更好。例如，也许在我们的 SIFT 表示包中，50 个视觉单词中有 40 个是无信息量的。它们根本不能帮助我们决定一张图像是“森林”还是“卧室”。也许它们代表了在所有类型的场景中出现的平滑补丁、梯度或阶梯边缘。来自最近邻分类器的预测仍然会受到这些频繁出现的视觉单词的严重影响，而线性分类器可以学习到特征向量的这些维度不太相关，因此在做出决策时降低它们的权重。有许多方法可以学习线性分类器，但我们将通过支持向量机找到线性决策边界。你不需要实现支持向量机。然而，线性分类器本质上是二分类的，我们有一个 15 路分类问题。为了决定一个测试用例属于 15 个类别中的哪一个，你将训练 15 个二元的、一对全的 svm。1-vs-all 意味着每个分类器将被训练来识别“森林”与“非森林”、“厨房”与“非厨房”等。所有 15 个分类器将在每个测试用例上进行评估，最有信心的正例分类器将“获胜”。例如，如果“kitchen”分类器返回-0.2 的分数(其中 0 是在决策边界上)，而“forest”分类器返回-0.3 的分数，而所有其他分类器甚至更负，即使没有一个分类器将测试用例放在决策边界的正侧，测试用例也将被分类为厨房。当学习一个 SVM 时，你有一个自由参数`lambda`，它控制模型的正则化程度。你的准确率会对 lambda 非常敏感，所以一定要测试多个值。详情参见 svm_classify()。

现在，我们可以使用一对多线性 svm 对 SIFT 表示进行评估。根据参数的不同，准确率应该在 60%到 70%之间。如果你实施下面的额外学分建议，你还可以做得更好。

从 student_code_SID.py 开始的入门代码包含了更多关于输入、输出的具体指导，以及你将使用的五个函数的建议策略

实现:get_tiny_images()， nearest_neighbor_classify()， build_voca bulary()，get_bags_of_sifts()，和 svm_classify()。utils 文件夹包含在 notebook 上可视化结果所需的代码，不需要你编辑。

实验设计

机器学习的一个重要方面是估计“好的”超参数。作为这个项目的一部分，你将执行以下三个实验，并在你的写作中分析结果:

- 使用交叉验证来衡量性能，而不是由起步代码提供的固定测试/训练分割。每次迭代随机选择 100 张训练和 100 张测试图像，并报告平均性能和标准偏差。
- 在你的训练过程中添加一个验证集来调整学习参数。这个验证集可以是训练集的子集，也可以是其他未使用的测试集的一部分。
- 使用不同的词汇量进行实验并报告性能。例如 10、20、50、100、200、400、1000、10000。

评价和可视化

初始代码通过在笔记本中每次运行 student_code_SID.py 时，在真实测试标签和预测测试标签之间产生一个混淆矩阵来构建一个混淆矩阵并可视化你的分类决策。

有用的功能

入门代码包含从 Scikit-Learn 实用程序到 VLFeat 命令的有用函数的更完整的讨论。

交

这是非常重要的，如果你不按照说明做，你会被扣分。每一次你不按照说明做，你就会被扣 5 分。上交的文件夹必须包含以下内容:

mycode/—包含本次赋值的所有代码的目录

将 student_code_SID_.py 中的“SID”替换为你的学生证。

- mycode /词汇。PKL -请确保你想让我们运行你的代码的词汇文件名为 vocab.pkl

SID.pdf—本次作业的报告。

标题

- +20 分:构建微小的图像特征用于场景识别。(get_tiny_images ())
- +10 分:最近邻分类器。(nearest_neighbor_classify ())
- +20 分:从一组随机训练特征中构建一个词汇表。(build_vocabulary ())
- +20 分:构建视觉单词直方图，用于训练和测试图像。(get_bags_of_sifts ())
- +10 分:在你的词袋模型上训练 1 对全部 svm。(svm_classify ())
- +10 pts:加速生成 sift 特征的过程。
- +10 pts:添加交叉验证，三种组合的准确率超过平均值。21%， 55%和 65%)
- -10 分:糟糕的代码风格。
- -10 分:没有分配报告。

最后的建议

- 提取特征、聚类以构建通用字典以及从特征构建直方图可能很慢。一个好的实现可以在不到 10 分钟的时间内运行整个管道，但这可能是以准确性为代价的(例如，视觉词汇表太小或采样率太稀疏)。

建议您使用我们提供的配置创建一个 conda 环境(命令:conda env create -f environment_assignment3.)。yml)。记得在打开笔记本的时候先激活环境。如果你的笔记本内核有一些错误，可以参考下面的配置。