

[CS231N - Assignment1 - Q5 - Image features exercises]

我们已经看到，通过用输入图像的像素训练的线性分类器对图像的分类问题已经取得了不错的成绩。在这个练习中我们会用对图像像素进一步计算得来的特征来训练线性分类器从而提高性能。

抽取特征 (Extract Features)

对于每张图，我们计算**梯度方向直方图(HOG)**特征和用**HSV (Hue色调, Saturation饱和度, Value明度)**颜色空间的**色调**特征。把每张图的梯度方向直方图和颜色直方图特征合并形成我们最后的特征向量。

粗略的讲呢，HOG应该可以捕捉到图像的纹理特征而忽略了颜色信息，颜色直方图会表示图像的颜色特征而忽略了纹理特征(详细见[这篇](#))。所以我们预估把两者结合起来得到的效果应该是比用其中一种得到的效果好。对于后面的bonus，验证一下这个设想是不错的选择。

`hog_feature` 和 `color_histogram_hsv` 两个函数都只对一张图做操作并返回这张图片的特征向量。`extract_features` 函数接收一堆图片和一个list的特征函数，然后用每个特征函数在每张图片上过一遍，把结果存到一个矩阵里面，矩阵的每一行都是一张图片的所有特征的合并。【注：题目中写的column,但从实际结果上来看应该是行】

1. 代码解析

```

1. from cs231n.features import *
2.
3. num_color_bins = 10 # Number of bins in the color histogram
4. feature_fns = [hog_feature, lambda img: color_histogram_hsv(img, nbin=
5. X_train_feats = extract_features(X_train, feature_fns, verbose=True)
6. X_val_feats = extract_features(X_val, feature_fns)
7. X_test_feats = extract_features(X_test, feature_fns)
8.
9. print(X_train_feats.shape, X_val_feats.shape, X_test_feats.shape)
10.
11. # 预处理: 减掉每一列特征的平均值
12. mean_feat = np.mean(X_train_feats, axis=0, keepdims=True)
13. X_train_feats -= mean_feat
14. X_val_feats -= mean_feat
15. X_test_feats -= mean_feat
16.
17.
18. # 预处理: 每一列除以标准差, 这确保了每个特征都在一个数值范围内
19. std_feat = np.std(X_train_feats, axis=0, keepdims=True)
20. X_train_feats /= std_feat
21. X_val_feats /= std_feat
22. X_test_feats /= std_feat
23.
24.
25. # 多加一个bias列
26. X_train_feats = np.hstack([X_train_feats, np.ones((X_train_feats.shape[0], 1
27. X_val_feats = np.hstack([X_val_feats, np.ones((X_val_feats.shape[0], 1
28. X_test_feats = np.hstack([X_test_feats, np.ones((X_test_feats.shape[0]

```

在features.py中写了两个特征的计算方法，HOG是改写了scikit-image的hog接口，并且首先要转换成灰度图。颜色直方图是实现用matplotlib.colors.rgb_to_hsv的接口把图片从RGB变成HSV，再提取明度(value)，把value投射到不同的bin当中去。关于HOG的原理请谷歌百度。

可能会踩的坑

1. "*Import Error: cannot import name imread*" 安装或者重装Pillow或PIL(pip intall PIL)。
如果安装提示已经安装了，可以重装，下载地址[在这里](#)，下载后，pip install xxxx.whl
2. `orientation_histogram[:, :, i] = uniform_filter(temp_mag, size=(cx, cy))[cx/2::cx, cy/2::cy].T` 这行报错,"*TypeError: slice indices must be integers or None or have an __index__ method*",可以把代码改成：
`orientation_histogram[:, :, i] = uniform_filter(temp_mag, size=(cx, cy))[int(cx/2)::cx, int(cy/2)::cy].T`

SVM训练(Train SVM on Features)

用前面作业中的多分类SVM来对上面抽取到的特征进行训练。这次应该会比之前直接在像

素上训练的结果更好。

```
1. from cs231n.classifiers.linear_classifier import LinearSVM
2.
3. #learning_rates = [1e-9, 1e-8, 1e-7]
4. #regularization_strengths = [5e4, 5e5, 5e6]
5.
6. results = {}
7. best_val = -1
8. best_svm = None
9.
10. #pass
11. learning_rates =[5e-9, 7.5e-9, 1e-8]
12. regularization_strengths = [(5+i)*1e6 for i in range(-3,4)]
13.
14. #####
15. # TODO:
16. # 用验证集来调整learning rate和regularization的强度。
17. # 这个应该和你之前做SVM做验证是一样的，把最好的结果保存在best_svm。你也许想试
18. # 试不同的颜色直方图的bin的个数。如果你调的够仔细，应该可以在验证集上得到
19. # 差不多0.44的正确率。
20. #####
21. for rs in regularization_strengths:
22.     for lr in learning_rates:
23.         svm = LinearSVM()
24.         loss_hist = svm.train(X_train_feats, y_train, lr, rs, num_iter)
25.         y_train_pred = svm.predict(X_train_feats)
26.         train_accuracy = np.mean(y_train == y_train_pred)
27.         y_val_pred = svm.predict(X_val_feats)
28.         val_accuracy = np.mean(y_val == y_val_pred)
29.         if val_accuracy > best_val:
30.             best_val = val_accuracy
31.             best_svm = svm
32.             results[(lr,rs)] = train_accuracy, val_accuracy
33. #####
34. #                                     END OF YOUR CODE
35. #####
36. # Print out results.
37. for lr, reg in sorted(results):
38.     train_accuracy, val_accuracy = results[(lr, reg)]
39.     print('lr %e reg %e train accuracy: %f val accuracy: %f' %
40.           lr, reg, train_accuracy, val_accuracy))
41. print('best validation accuracy achieved during cross-validation: %f'
```

建议这里可以把learning rate和regularization_strengths开个大一点的区间，多试试。

```
1. # 想知道算法是如何运作的很重要的方法是把它的分类错误可视化。在这里的可视化中，我们
2. # 展示了我们系统错误分类的图片。比如第一列展示的是实际label是飞机，但是被我们系统误
3. # 标注成其它label的图片。
4. examples_per_class = 8
5. classes = ['plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse']
6. for cls, cls_name in enumerate(classes):
7.     idxs = np.where((y_test != cls) & (y_test_pred == cls))[0]
8.     idxs = np.random.choice(idxs, examples_per_class, replace=False)
9.     for i, idx in enumerate(idxs):
10.         plt.subplot(examples_per_class, len(classes), i * len(classes))
11.         plt.imshow(X_test[idx].astype('uint8'))
12.         plt.axis('off')
13.         if i == 0:
14.             plt.title(cls_name)
15. plt.show()
```

Inline question 1:

描述一下你模型分错类的结果。他们看起来有合理吗？

Answer: 笔者表示看起来不怎么合理....LOL。

用神经网络训练图片特征

在前面的作业中我们看到直接在像素上训练两层的神经网络结果比线性分类器的效果好。在这个notebook中我们已经看到线性分类器在图片特征上比在像素上效果好。

为了完整性，我们应该试试在图片特征上用神经网络。这个方法应该比前面的方法都好：你应该在测试集上很容易得到一个55%的分类结果。我们最好的模型达到了60%的准确率。

```

1. from cs231n.classifiers.neural_net import TwoLayerNet
2.
3. input_dim = X_train_feats.shape[1]
4. hidden_dim = 500
5. num_classes = 10
6.
7. net = TwoLayerNet(input_dim, hidden_dim, num_classes)
8. best_net = None
9.
10. #####
11. # TODO: Train a two-layer neural network on image features. You may wa
12. # cross-validate various parameters as in previous sections. Store you
13. # model in the best_net variable.
14. #####
15. learning_rates = [1e-2, 1e-1, 5e-1, 1, 5]
16. regularization_strengths = [1e-3, 5e-3, 1e-2, 1e-1, 0.5, 1]
17.
18. for lr in learning_rates:
19.     for reg in regularization_strengths:
20.         net = TwoLayerNet(input_dim, hidden_dim, num_classes)
21.         # Train the network
22.         stats = net.train(X_train_feats, y_train, X_val_feats, y_val,
23.                         num_iters=1500, batch_size=200,
24.                         learning_rate=lr, learning_rate_decay=0.95,
25.                         reg=reg, verbose=False)
26.         val_acc = (net.predict(X_val_feats) == y_val).mean()
27.         if val_acc > best_val:
28.             best_val = val_acc
29.             best_net = net
30.             results[(lr, reg)] = val_acc
31.
32. # Print out results.
33. for lr, reg in sorted(results):
34.     val_acc = results[(lr, reg)]
35.     print 'lr %e reg %e val accuracy: %f' % (
36.             lr, reg, val_acc)
37.
38. print 'best validation accuracy achieved during cross-validation: %f'
39. #pass
40. #####
41. #                                     END OF YOUR CODE
42. #####

```

这部分基本上复用前面的代码就可以了，参数还是要多调试。

Bonus: Design your own features!

你已经看到了简单的图片特征可以提升分类效果。到目前为止我们尝试了HOG和颜色直方图，但是其他类型的特征也许能得到更好的分类效果。设计一个新的特征。把它用在CIFAR-10上。解释你的特征是如何运作的，你为什么觉得它会对图像分类有用。在这个

*notebook*中实现它，使用交叉验证来调整超参数，并和HOG特征+颜色直方图的特征的*baseline*做比较。

Bonus: Do something extra!

用这个作业中的资料和代码做点有趣的事。是否有我们应该问的其他问题？你在做这个作业的时候有其他很棒的想法吗？做做吧！

这里可以验证上面提到过的把HOG特征和颜色直方图的特征单独拿出来效果是否变差。