最近在刷cs231n的课程和作业，在这里分享下自己的学习过程，同时也希望能够得到大家的指点。

写在前面：

1. 这仅仅是自己的学习笔记，如果侵权，还请告知;
2. 代码是参照lightaime的github，在其基础之上做了一些修改;
3. 在我之前，已有前辈在他的知乎上分享过类似内容;
4. 讲义是参照杜客等人对cs231n的中文翻译。

温馨提醒：

1. 文档的所有程序是使用python3.5实现的，如果你是python2的用户，可能要对代码稍作修改；
2. 文章频繁提到将代码保存到`.py`中，是为了方便接下来的模块导入，希望读者可以理解。

这节的作业我们通过以下几点来完成：

1. 特征提取；
2. 数据处理；
3. svm训练；
4. neural net训练。

前面我们所介绍的，不管是softmax线性分类器还是神经网络，输入都是图像的像素信息，通过分类器完成特征的学习和提取，进而实现对图像的分类工作。那么如果我们网络输入的是已经学习好的特征呢，这样分类的效果是不是就更好了呢？我们这节的主要任务就是将提取到的图像特征输入网络，进而完成图像分类。

1 特征提取 featuresExtract.py

对于每一张图片，我们使用HOG来提取边缘特征、色彩直方图（使用HSV颜色空间的色彩通道）来提取颜色特征，然后将二者加和来获得最后的特征向量。

代码如

```
import numpy as np
from scipy.ndimage import uniform_filter
import matplotlib
```

```
def extract_features(imgs,feature_fns,verbose=False):
      '''Given pixel data for images and several feature functions that
can operate on  single images, apply all feature functions to all images,
concatenating the feature vectors for each image and storing the features
for all images in a single matrix.'''
```

```
      '''Inputs:'''
      '''- imgs: N x H X W X C array of pixel data for N images.'''
      '''- feature_fns: List of k feature functions. The ith feature function should take as input an H x W x D array and return a (one-dimensional) array of length F_i.'''
      '''- verbose: Boolean; if true, print progress.'''
      '''Returns:'''
      '''An array of shape (N, F_1 + … + F_k) where each column is the concatenation of all features for a single image.'''
    num_images=imgs.shape[0]行
    if num_images==0:
        return np.array([])
    feature_dims=[]
    first_image_features=[]
    for feature_fn in feature_fns:
        feats=feature_fn(imgs[0].squeeze())
        assert len(feats.shape)==1,'Feature functions must be one-dimensional'
        feature_dims.append(feats.size)
        first_image_features.append(feats)

        total_feature_dim=sum(feature_dims)
        imgs_features=np.zeros((num_images,total_feature_dim))
        imgs_features[0]=np.hstack(first_image_features).T

    for i in range(1,num_images):
        idx=0
        for feature_fn,feature_dim in zip(feature_fns,feature_dims):
            next_idx=idx+feature_dim
            imgs_features[i,idx:next_idx]=feature_fn(imgs[i].squeeze())
            idx=next_idx
        if verbose and i % 1000==0:
            print('Done extracting features for %d / %d images' % (i,num_images))

    return imgs_features

                    ...
def rgb2gray(rgb):
    return np.dot(rgb[…,:3],[0.299,0.587,0.144])

def hog_feature(im):
    #对一幅图像提取hog特征
    if im.ndim==3:
        image=rgb2gray(im)
    else:
        image=np.atleast_2d(im)把输入看成至少有两个dimension的array
```

```
44
45      sx,sy=image.shape #图像尺寸
46      orientations=9 #每个细胞直方图通道个数
47      cx,cy=(8,8) #每个细胞像素个数
48
49      #计算图像梯度
50      gx=np.zeros(image.shape)
51      gy=np.zeros(image.shape)
52      gx[:,:-1]=np.diff(image,n=1,axis=1) #计算x方向的梯度
53      gy[:-1,:]=np.diff(image,n=1,axis=0) #计算y方向的梯度
54      grad_mag=np.sqrt(gx**2+gy**2) #梯度幅值
55      grad_ori=np.arctan2(gy,(gx+1e-15))*(180/np.pi)+90 #梯度方向
56
57      n_cellsx=int(np.floor(sx/cx)) #一个区间的细胞个数
58      n_cellsy=int(np.floor(sy/cy))
59
60
61      orientation_histogram=np.zeros((n_cellsx,n_cellsy,orientations))
62      for i in range(orientations):
63          #为每个细胞单元构建梯度方向直方图
64          temp_ori=np.where(grad_ori<180 / orientations*(i+1),grad_ori,0)
65          temp_ori=np.where(grad_ori >=180 / orientations*i,temp_ori,0)
66          cond2=temp_ori>0
67          #把细胞单元组合成大的块，块内归一化梯度直方图
68          temp_mag=np.where(cond2,grad_mag,0)
69          orientation_histogram[:,:,i]=uniform_filter(temp_mag,size=
(cx,cy))[int(cx/2)::cx,int(cy/2)::cy].T
70      return orientation_histogram.ravel()
71
72  def color_histogram_hsv(im,nbin=10,xmin=0,xmax=255,normalized=True):
73      #对一幅图像提取颜色直方图
74      ndim=im.ndim
75      bins=np.linspace(xmin,xmax,ndim+1)
76      hsv=matplotlib.colors.rgb_to_hsv(im/xmax)*xmax
77
 imhist,bin_edges=np.histogram(hsv[:,:,0],bins=bins,density=normalized)
78      imhist=imhist*np.diff(bin_edges)
79
80      return imhist
```

注释是我根据自己的理解加上的，如有不当，还请指正哈～

其中，HOG特征提取算法的实现大致过程如下：
HOG特征提取方法就是将一个image（你要检测的目标或者扫描窗口）：

1. 灰度化（将图像看做一个x,y,z（灰度）的三维图像）；

2. 采用Gamma校正法对输入图像进行颜色空间的标准化（归一化）；目的是调节图像的对比度，降低图像局部的阴影和光照变化所造成的影响，同时可以抑制噪音的干扰；
3. 计算图像每个像素的梯度（包括大小和方向）；主要是为了捕获轮廓信息，同时进一步弱化光照的干扰。
4. 将图像划分成小cells（例如6*6像素/cell）；
5. 统计每个cell的梯度直方图（不同梯度的个数），即可形成每个cell的descriptor；
6. 将每几个cell组成一个block（例如3*3个cell/block），一个block内所有cell的特征descriptor串联起来便得到该block的HOG特征descriptor；
7. 图像image内的所有block的HOG特征descriptor串联起来就可以得到该image（你要检测的目标）的HOG特征descriptor了。这个就是最终的可供分类使用的特征向量了。
   具体实现过程详见zouxy09博客

这样，我们就完成了对图像边缘和颜色特征的提取工作。

2 数据处理 DataPreprocess.py

数据处理主要有以下几步：

1. cifar10数据下载；
2. 特征提取；
3. 数据预处理（减**训练集**均值，除以**训练集**方差）；
4. 增加一个维度，由(N,D)变为(N,D+1)。
   主要代码如下：

```
import numpy as np
from data_utils import load_cifar10
from featuresExtact import
hog_feature,color_histogram_hsv,extract_features
```

```
def get_cifar_data(num_training=49000,num_validation=1000,num_test=1000):
    cifar10_dir='..\knn\cifar-10-batches-py'
    X_train,y_train,X_test,y_test=load_cifar10(cifar10_dir)
    # 验证集
    mask=range(num_training,num_training+num_validation)
    X_val=X_train[mask]
    y_val=y_train[mask]
    #训练集
    mask=range(num_training)
    X_train=X_train[mask]
    y_train=y_train[mask]
    #测试集
    mask=range(num_test)
    X_test=X_test[mask]
```

```
    y_test=y_test[mask]

    return X_train,y_train,X_val,y_val,X_test,y_test
X_train,y_train,X_val,y_val,X_test,y_test=get_cifar_data()
num_color_bins = 10
feature_fns = [hog_feature, lambda img: color_histogram_hsv(img,
nbin=num_color_bins)]
X_train_feats = extract_features(X_train, feature_fns, verbose=True)
X_val_feats = extract_features(X_val, feature_fns)
X_test_feats = extract_features(X_test, feature_fns)

mean_feat=np.mean(X_train_feats,axis=0,keepdims=True)
X_train_feats-=mean_feat
X_val_feats-=mean_feat
X_test_feats-=mean_feat
std_feat=np.std(X_train_feats,axis=0,keepdims=True)
X_train_feats/=std_feat
X_val_feats/=std_feat
X_test_feats/=std_feat

X_train_feats=np.hstack([X_train_feats,np.ones((X_train_feats.shape[0],1)
)])
X_val_feats=np.hstack([X_val_feats,np.ones((X_val_feats.shape[0],1))])
X_test_feats=np.hstack([X_test_feats,np.ones((X_test_feats.shape[0],1))])
```

从代码中我们可以看到，对特征的预处理和对图像像素的预处理是差不多的，我们可以把特征当作是对图像像素更形象化的表达。

这样，我们就完成了对提取来的特征的处理工作，下面我们就使用提取到的特征来训练和预测，看看会不会比直接对图像像素进行训练和预测强些。（哈哈，答案当然是肯定的，如果你有深度学习的一些基础的话。）

3 使用特征训练svm trainSVM_with_Features.py

使用提取到的特征训练svm，同前期文章，我们只需要将训练数据载入到训练模型即可，这里我们直接利用交叉验证来选择最好的svm线性分类模型。
代码如下：

```
from linear_classifer import LinearSVM
from DataPreprocess import *
learning_rates=[1e-9,1e-8,1e-7]
regularization_strengths=[(5+i)*1e6 for i in range(-3,4)]

results={}
best_val=-1
```

```
best_svm=None

for rs in regularization_strengths:
    for lr in learning_rates:
        svm=LinearSVM()
        loss_hist=svm.train(X_train_feats,y_train,lr,rs,num_iters=6000)
        y_train_pred=svm.predict(X_train_feats)
        train_accuracy=np.mean(y_train==y_train_pred)
        y_val_pred=svm.predict(X_val_feats)
        val_accuracy=np.mean(y_val==y_val_pred)
        if val_accuracy > best_val:
            best_val=val_accuracy
            best_svm=svm
        results[(lr,rs)]=train_accuracy,val_accuracy

for lr,reg in sorted(results):
    train_accuracy,val_accuracy=results[(lr,reg)]
    print('lr %e reg %e train accuracy: %f val accuracy: %f' %
(lr,reg,train_accuracy,val_accuracy))

print('best validation accuracy achieved during cross-validation: %f' %
best_val)
```

输出结果如下：

```
lr 1.000000e-09 reg 2.000000e+06 train accuracy: 0.413000 val accuracy:
0.414000
lr 1.000000e-09 reg 3.000000e+06 train accuracy: 0.416857 val accuracy:
0.422000
lr 1.000000e-09 reg 4.000000e+06 train accuracy: 0.415694 val accuracy:
0.418000
lr 1.000000e-09 reg 5.000000e+06 train accuracy: 0.414857 val accuracy:
0.413000
lr 1.000000e-09 reg 6.000000e+06 train accuracy: 0.415796 val accuracy:
0.424000
lr 1.000000e-09 reg 7.000000e+06 train accuracy: 0.416816 val accuracy:
0.424000
lr 1.000000e-09 reg 8.000000e+06 train accuracy: 0.415286 val accuracy:
0.412000
lr 1.000000e-08 reg 2.000000e+06 train accuracy: 0.410408 val accuracy:
0.396000
lr 1.000000e-08 reg 3.000000e+06 train accuracy: 0.412816 val accuracy:
0.407000
lr 1.000000e-08 reg 4.000000e+06 train accuracy: 0.413102 val accuracy:
0.431000
```

```
lr 1.000000e-08 reg 5.000000e+06 train accuracy: 0.413878 val accuracy: 0.413000
lr 1.000000e-08 reg 6.000000e+06 train accuracy: 0.409347 val accuracy: 0.413000
lr 1.000000e-08 reg 7.000000e+06 train accuracy: 0.408449 val accuracy: 0.405000
lr 1.000000e-08 reg 8.000000e+06 train accuracy: 0.416286 val accuracy: 0.411000
lr 1.000000e-07 reg 2.000000e+06 train accuracy: 0.405796 val accuracy: 0.394000
lr 1.000000e-07 reg 3.000000e+06 train accuracy: 0.384469 val accuracy: 0.390000
lr 1.000000e-07 reg 4.000000e+06 train accuracy: 0.370122 val accuracy: 0.385000
lr 1.000000e-07 reg 5.000000e+06 train accuracy: 0.376531 val accuracy: 0.386000
lr 1.000000e-07 reg 6.000000e+06 train accuracy: 0.367367 val accuracy: 0.358000
lr 1.000000e-07 reg 7.000000e+06 train accuracy: 0.343245 val accuracy: 0.326000
lr 1.000000e-07 reg 8.000000e+06 train accuracy: 0.354245 val accuracy: 0.363000
best validation accuracy achieved during cross-validation: 0.431000
```

在验证集上的准确率最高可达0.431，明显优于svm分类器0.396的准确率。

我们看下预测的效果，代码如下：

SVM_Predict_and_Visualize.py

```
import numpy as np
from trainSVM_with_Features import best_svm
from DataPreprocess import X_test_feats,y_test

y_test_pred=best_svm.predict(X_test_feats)
test_accuracy=np.mean(y_test==y_test_pred)
print(test_accuracy)
```

结果如下：

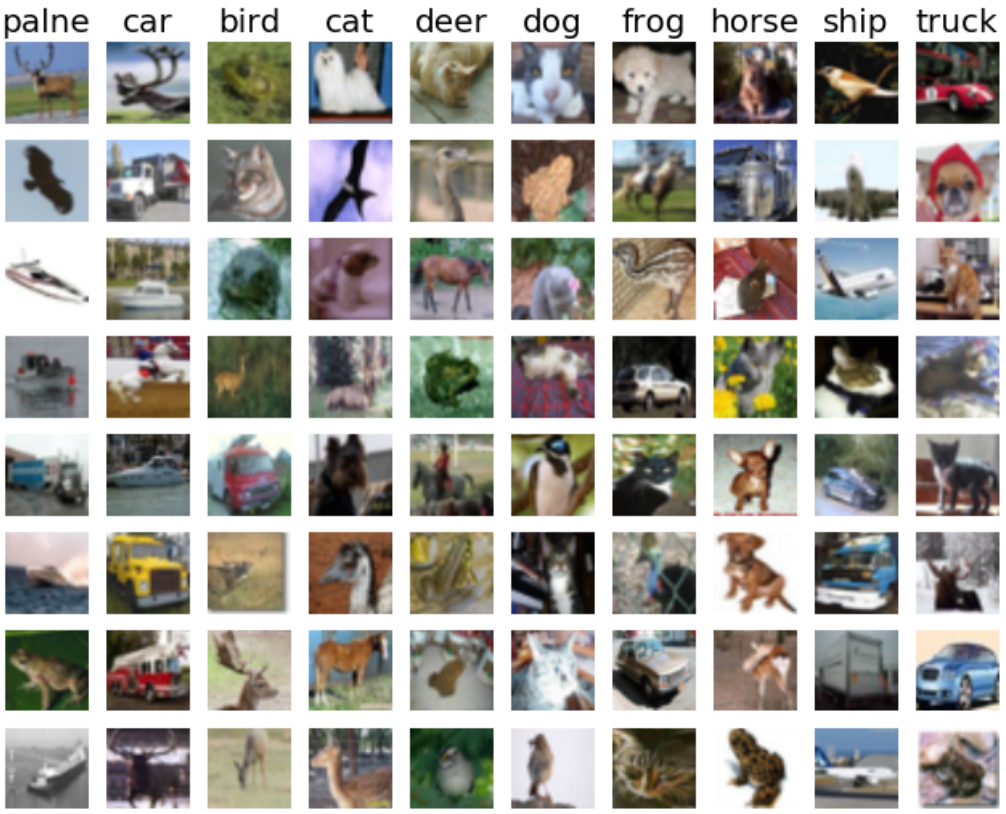
```
0.423
```

0.423的准确率，比以图像像素为输入的svm线性分类器好很多了，但是还是有些瑕疵的，我们可以看下哪些被错误分类了，代码如下：

```
examples_per_calss=8
classes=
['palne','car','bird','cat','deer','dog','frog','horse','ship','truck']
```

```
for cls,cls_name in enumerate(classes):
    idxs=np.where((y_test !=cls) & (y_test_pred==cls))[0]
    idxs=np.random.choice(idxs,examples_per_calss,replace=False)
    for i ,idx in enumerate(idxs):
        plt.subplot(examples_per_calss,len(classes),i*len(classes)+cls+1)
        plt.imshow(X_test[idx].astype('uint8'))
        plt.axis('off')
        if i ==0:
            plt.title(cls_name)
plt.show()
```

展示结果如图：



我们可以看下，比如第一列，模型把一头鹿都分到了飞机的这一类中。

虽然有些错误，但总之我们通过训练提取之后的特征，能够获得比之前更高的预测准确率，还算是有了进步的。

4 使用特征训练神经网络 trainNeuralNet_with_Features.py

我们同样来看下在神经网络上面的效果，代码如下：

```
input_dim=X_train_feats.shape[1]
```

```
hidden_dim=500
num_classes=10

net=TwoLayerNet(input_dim,hidden_dim,num_classes)

results={}
best_val=-1
best_net=None

learning_rates = [1e-2, 1e-1, 5e-1, 1, 5]
regularization_strengths = [1e-3, 5e-3, 1e-2, 1e-1, 0.5, 1]

for lr in learning_rates:
    for reg in regularization_strengths:
        net = TwoLayerNet(input_dim, hidden_dim, num_classes)
        # Train the network
        stats = net.train(X_train_feats, y_train, X_val_feats, y_val,
                          num_iters=1500, batch_size=200,
                          learning_rate=lr, learning_rate_decay=0.95,
                          reg=reg, verbose=False)
        val_acc = (net.predict(X_val_feats) == y_val).mean()
        if val_acc > best_val:
            best_val = val_acc
            best_net = net
        results[(lr, reg)] = val_acc

for lr, reg in sorted(results):
    val_acc = results[(lr, reg)]
    print('lr %e reg %e val accuracy: %f' % (lr, reg, val_acc))

print('best validation accuracy achieved during cross-validation: %f' %
best_val)

test_acc = (best_net.predict(X_test_feats) == y_test).mean()
print (test_acc)
```

输出结果如下：

```
lr 1.000000e-02 reg 1.000000e-03 val accuracy: 0.196000
lr 1.000000e-02 reg 5.000000e-03 val accuracy: 0.140000
lr 1.000000e-02 reg 1.000000e-02 val accuracy: 0.172000
lr 1.000000e-02 reg 1.000000e-01 val accuracy: 0.079000
lr 1.000000e-02 reg 5.000000e-01 val accuracy: 0.087000
lr 1.000000e-02 reg 1.000000e+00 val accuracy: 0.105000
lr 1.000000e-01 reg 1.000000e-03 val accuracy: 0.526000
```

NeuralNet_Predict_and_visualize.py

```
import numpy as np
from trainNeuralNet_with_Features import best_net
from DataPreprocess import X_test_feats,y_test
y_test_pred=best_net.predict(X_test_feats)
test_accuracy=np.mean(y_test==y_test_pred)
print('test accuracy: %f' %test_accuracy)
import matplotlib.pyplot as plt
from DataPreprocess import X_test
#分类结果可视化
examples_per_calss=8
classes=['palne','car','bird','cat','deer','dog','frog','horse','ship','truck']
for cls,cls_name in enumerate(classes):
    idxs=np.where((y_test !=cls) & (y_test_pred==cls))[0]
    idxs=np.random.choice(idxs,examples_per_calss,replace=False)
    for i ,idx in enumerate(idxs):
        plt.subplot(examples_per_calss,len(classes),i*len(classes)+cls+1)
        plt.imshow(X_test[idx].astype('uint8'))
        plt.axis('off')
        if i ==0:
            plt.title(cls_name)
plt.show()
```

```
lr 1.000000e-01 reg 5.
lr 1.000000e-01 reg 1.
lr 1.000000e-01 reg 1.
lr 1.000000e-01 reg 5.
lr 1.000000e-01 reg 1.
lr 5.000000e-01 reg 1.
lr 5.000000e-01 reg 5.
lr 5.000000e-01 reg 1.
lr 5.000000e-01 reg 1.
lr 5.000000e-01 reg 5.
lr 5.000000e-01 reg 1.
lr 1.000000e+00 reg 1.
lr 1.000000e+00 reg 5.
lr 1.000000e+00 reg 1.
lr 1.000000e+00 reg 1.
lr 1.000000e+00 reg 5.
lr 1.000000e+00 reg 1.
lr 5.000000e+00 reg 1.
lr 5.000000e+00 reg 5.000000e-03 val accuracy: 0.119000
lr 5.000000e+00 reg 1.000000e-02 val accuracy: 0.102000
lr 5.000000e+00 reg 1.000000e-01 val accuracy: 0.079000
lr 5.000000e+00 reg 5.000000e-01 val accuracy: 0.087000
lr 5.000000e+00 reg 1.000000e+00 val accuracy: 0.087000
best validation accuracy achieved during cross-validation: 0.588000
0.568
```

最后在测试集上的准确率达到了0.568，我们回去看下使用原始图像数据预测的准确率0.502，还是有了不错的提升的。

5 总结

本次作业中，我们首先提取了图像的轮廓和颜色特征，将这些特征用于网络训练，得到了比使用原始数据更好的训练结果。
这对我们之后的学习，有两点帮助：

1. 我们可以增加深度网络的深度，因为深度越深，学到的特征也就越多；
2. 如果网络不够复杂，我们可以首先采用机器学习或者传统方法提取我们需要的特征，将这些特征用于训练。