Programming Assignment 2

In this assignment you will implement a system for visual object recognition which is one of the fundamental tasks in computer vision. In this task you will be given multiple sets of images for training. Each of those sets contain several pictures taken from a specific category of objects (the category names are the labels). You also will be given a set of images as test set in which you don't know their category names. Your job is to predict a category name for each image in the test set.

As before: Questions are given in red

Part 1: Dataset

We are going to use an state-of-the-art dataset in visual object recognition. The name of the dataset is "Caltech101". It has 101 categories of objects and around 100 images per each category. You can see the images from this dataset here: http://www.vision.caltech.edu/Image_Datasets/Caltech101/

Due to the enviormental conditions (light, shadows, direction of camera, ...) and intra-category variation of visual objects, the raw RGB/HSV pixel values of images can not give us consistent information about images. Therefor, we need to extract a robust descriptor for each images. DON'T WORRY these are already extracted for you. We used the covariance of image pixels' value in each region of an image and made a histogram out of those covariance values. You can find the details about this descriptor here: http://www.merl.com/reports/docs/TR2005-111.pdf
You can find the dataset in "./caltech101/". There you will see three folders "/training/","/validation/","/test/". In the "/training/" and "/validation/" you have 101 subfolders each of them named by a category of objects (e.g. "car","dog","tree",...) and in each of those subfolders you have a number of text files which are the image descriptors. In folder "/test/" you have a bunch of textfiles same as before each of them is an image descriptor but you don't have their category names.
Load the training and validation data using the following code:

In [1]:
```
import numpy as np
import os
DIR_categories=os.listdir('./caltech101/training/');
imFeatures=[]
imLabels=[]
i=0;
for cat in DIR_categories:
    if os.path.isdir('./caltech101/training/'+ cat):
       i=i+1;
       DIR_image=os.listdir('./caltech101/training/'+ cat +'/');
       for im in DIR_image:
           F = np.genfromtxt('./caltech101/training/'+cat+'/'+im, delimiter='
           F = np.reshape(F,21*28)
           F = np.mat(F);
           F = F.tolist();
           imFeatures.append(F);
           imLabels.append(i);
```

Q1: How many dimensions (features) does the image descriptor has?

Q2: Make a barplot for the training set that shows the number of images in each category. (x-axis ---> category label, y-axix ---> number of image in the category)

Part 2: Linear Classifiers

So far in our class you learned how to design a binary classifier meaning that you only had two different labels (+1,-1). But here we have more than two categories (101 categories). How can we learn a classifier to classify examples from more than two classes? This problem is called "Multiple-Class Classification" (see CIML Chapter 5). To deal with this problem, we can learn separate binary classifiers for each category independently (e.i. 101 different classifiers) by taking the positive samples from one category and the negative samples from all the remaining categories. For a given test sample we apply all the 101 classifiers on it and choose the category label based on the classifier that gave us the most confident classification score. This method is called One-vs-All.

Q3:Can you suggest any other way to deal with multiple class classification? Compare it with One-vs-All.

Q4: In the One-vs-All senario you will have much more negative samples than positives for each classifier. Does this fact cause a reduction in the performance of the learning algorithm? if yes, why? and what is the your solution?

A) Averaged Perceptron:

Q5: Using the One-vs-All method train 101 binary classifiers on the training set using the Averaged Perceptron algorithm (Algorithm 7 in CIML, Chapter 3). Use your classifiers to predict a category label for each image in the verification set.

Please paste source code here.

Evaluate you results by this accuracy measure: $Acc = 100 \times \frac{num-of-correct-predictions}{N}$ where N is the number of all images that you are testing on.

Q6: Randomly select 5 image per category in the train set and learn the (multi-class) perceptron and report your accuracy on the validation set. Do it for 10,20,30,40,50, and 60 images per category in train set and make a plot showing the number-of-training-samples-per-category(x-axis) vs Accuracy(y-axis). (This is commonly called a *learning curve*). Discuss the curve on the plot. Can we always get better accuracy by having more training data? Why? (NOTE: If there is one category with less number of samples that you need for training use all the samples in that category for training)

Q7: Using 50 training samples per category learn the (multi-clss) averaged perceptron with different number of iterations [1,10,50,100,500,1000]. Make and discuss a plot presenting number-of-iterations(x-axis) vs Accuracy(y-axis).

B) Subgradient descent for $l2$-regularized hinge loss

Similar to section A use the one-vs-all method to train 101 linear SMVs using subgradient descent (CIML: Alg 23, p94).

Q8: Using 50 training samples per category learn the multi-class SVM with differnt regularization parameter $\lambda = 2^{[-3,-2,-1,0,1,2,3,4,5,6]}$. Make and discuss a plot showing (λ vs Accuracy on the validation set). How does λ can affect on generalization?

Q9: Using 50 training samples per category learn the multi-class SVM with different step-size $\eta = [0.0001, 0.001, 0.01, 0.1, 0.3, 0.5, 0.8, 1]$. Make and discuss a plot of η vs Accuracy on the validation set. Discuss about the curve on the plot.

Part 3: Linear SVM

We are going to try Linear SVMs again as our binary classifier, but this time not using subgradient descent. Instead, we are going to use a very useful library for SVM. This library called "libsvm" you can find it here: "./libsvm-3.12/". Unix and MacOS users can do:

1) run "make" under"./libsvm-3.12/"
2) run "make" under "./libsvm-3.12/python/"

Windows users must follow the structure given in the "./libsvm-3.12/python/README".

To make sure that it is correctly installed you should be able run this code:

```
In [ ]: from svmutil import *
        y, x = svm_read_problem('../heart_scale')
        m = svm_train(y[:200], x[:200], '-c 4 -t 0')
        p_label, p_acc, p_val = svm_predict(y[200:], x[200:], m)
```

Q10: Same as Part 2-A using the One-vs-All method and varying the number of training examples per category, learn linear SVMs and report the same plot as Part 2-A.

Q11: Now fix the number of training samples at 50 and vary the hyperparameter C in the range [0.0001,0.001,0.01,0.1,0.5,1,10,100,1000,1000,10000]. Draw a plot showing the change of C vs Accuracy in the validation test. There should be an arc-shape curve with maximum on the plot. Discuss reasons for having lower accuracy on either side of the maximum.

Part 4: Compettition on test set

Q12: Do your best in validation set and choose your best model and apply it on the test set and report your predicted labels in a text file named "test.out" with this format: every line of the file began with the name of an image file then a comma and then the category label. Here is an example:
image_test0001.jpg.txt,accordion

image_test0007.jpg.txt,car

image_test0018.jpg.txt,tree
... .. .

We will compute the accuracy of your result and grade you based on the position of your accuracy in comparison with other students. Prizes will be awarded!