# Computational Photography

Assignment 5 - Classification
Summer 2023

## Introduction

In this assignment you will demonstrate your ability to perform feature extraction for the purpose of image classification.

You **CANNOT** use any built-in Machine Learning functions to do KNN for you. Nor can you use any function to extract GIST or HOG representations for you.

## Grading

| | |
|---|---|
| Theory Questions | 30pts |
| K-NN on Grayscale histograms | 35pts |
| K-NN on Gists | 35pts |
| **TOTAL** | 100pts |

Table 1: Grading Rubric

## 1 (Theory Questions

1. Given the following image pixel intensity values

$$I = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 3 \end{bmatrix}$$

   (a) Perform k-means clustering with k=2. Your initial references vectors will be $a_1 = [1]$ and $a_2 = [2]$. You will be "running" the k-means algorithm *manually*. Until the cluster assignments don't change:

      i. Assign an observation (pixel) to the reference vector it is closest to, using the Euclidean distance on the pixel value.

      ii. Update the reference vectors to be the means of their members.

   Show the value of the reference vectors, and the cluster assignments for each iteration you must perform (10pts).

   **1st Iteration:**
   $Cluster1(A_1) : [1, 0, 0, 1] \rightarrow A_1 = \frac{(1+1)}{4} = 0.5$
   $Cluster2(A_2) : [2, 3] \rightarrow A_2 = \frac{(2+3)}{2} = 2.5$
   **2st Iteration:**
   $Cluster1(A_1) : [0.5, 0, 0, 0.5] \rightarrow A_1 = \frac{(.5+.5)}{4} = 0.25$
   $Cluster2(A_2) : [2.5, 3] \rightarrow A_2 = \frac{(2.5+3)}{2} = 2.75$

   (b) Using the same grayscale image above, compute the weights between the pixels for a fully-connected, undirected graph representation. You can either show this as a drawing of the graph (in which case, a hand-drawn visualizations, inserted into your PDF, with the weights on the edges is fine) or provide the weight matrix (the upper-diagonal of it is fine, since it will be symmetric). For the weights of the edges connecting pixels, we'll use a combination of their value and location. For pixel $a$, let $a_i$ be its value/intensity, and $(a_x, a_y)$ be its location. We can then compute the similarity/weight between pixels a and b as:

$$w(a, b) = e^{-((a_i-b_i)^2+(a_x-b_x)^2+(a_y-b_y)^2)}$$

   In addition, consider a pixel not to be connected to itself (weight=0). You should leave your weights in terms of $e$ (10pts).

$$\textbf{Weight Matrix:} \begin{bmatrix} 0.0498 & 0.3679 & 0.0498 & 0.0001 \\ 0.0001 & 0.0498 & 0.3679 & 0.0000 \\ 0.0000 & 0.0025 & 0.0025 & 0.1353 \end{bmatrix}$$

   (c) Find the minimum non-trivial graph cut using the matrix formulation way shown in class. You may (and should) use function like *svd* do eigen-decomposition for you. You'll likely need to read its documentation to understand how to the inputs and outputs work. Show the intermediate matrices needed for your eigen-decomposition, namely $D$ and $W$, and what the chosen eigenvalue/vector pair is. Finally draw your new (cut) graph (and include that image) and/or just tell us which pixels belong to which groups. (10pts).

## The Dataset

The success of traditional machine learning algorithms is highly dependent on feature extraction. In this assignment you will get experience:

1. Forming training and testing data sets
2. Implementing and evaluating a K-Nearest-Neighbors classifier.
3. Extracting global and local features

On BBlearn we have provided a dataset for use in training a classifier to detect images as containing a car vs not containing a car. Since we need labels (for our training set and to evaluate our testing set) we will only use the TrainImages subdirectory for both training and and testing data.

The dataset is structured as follows: http://cogcomp.org/Data/Car/

The following code can be used to parse this subdirectory:

```
d = './CarData/TrainImages'
files = dir(d);
X = [];
Y = [];
for f = files'
   if ~f.isdir
       im = imread([d, '/', f.name]);
 %do whatever you need to in order to generate feature vector for im
 %which I'll call fv
       X(end+1,:) = fv;
       Y(end+1,1) = ~strcmp(f.name(1:3),'neg');
   end
 end
```

And to shuffle and divide your data in to training and validation subsets...

```
rng(0);
inds = randperm(size(X,1));
num = size(X,1)/3;
X = X(inds,:);
Y = Y(inds,:);

Xtrain = X(1:2*num,:);
Ytrain = Y(1:2*num,:);

Xvalid = X(2*num+1:end,:);
Yvalid = Y(2*num+1:end,:);
```

## 2 Classifying an Image using Grayscale Histograms

For each image in your dataset, compute a grayscale histogram with 256 bins and extract a class label from the first three characters of the file name, neg,pos. For simplicity you may want to enumerate these class labels.

*Note: It may take a while to traverse this directory and make your data matrices. Therefore, you may consider saving your representations and labels in some file and read them in as needed.*

Now we want to make our training and validation sets. First set the random number generator's seed to zero so that you can have reproducible results. Next shuffle the data and select 2/3 of it for training and the remaining for validation.

Now that we have our datasets created, let's classify!

Go through each validatioon sample's histogram and classify them as car or not-car using a k-nearest neighbors approach, where $k = 5$. For our similarity metric, we'll use the histogram intersection (where $D$ is the number of bins):

$$sim(a, b) = \sum_{j=1}^{D} min(a_j, b_j)$$

Note that histogram intersection is a *similarity* measurement, so when implementing KNN you'll want to find the *K most* ***similar*** observations (as opposed the the K nearest).

For your report, compute the accuracy as the percentage of the validation images classified correctly. In addition show:

1. One image correctly labeled as a car

2. One image correctly labeled as not a car

3. One image incorrectly labeled as a car

4. One image incorrectly labeled as not a car

**Accuracy: 65.4%**

## 3 Classifying an Image using Gists

Next let's attempt to classification using local gradient information.

Divide your grayscale images into 10 non-overlapping $20 \times 20$ sub-images, computing an 8-bin HOG at each sub-image. Concatenate these ten 8-bin HOGs to form an 80-feature representation of the image.

Now repeat your classification (kNN with k=5) and evaluation as in Part 1, but using this representation. Note that now your histogram intersection similar metric will just sum over 80 bins. **Accuracy: 70%**

## Submission

For your submission, upload to Blackboard a single zip file containing:

1. PDF writeup that includes:
    (a) Your answer to the theory question(s).
    (b) Accuracy for Part 2
    (c) Sample images for Part 2
    (d) Accuracy for Part 3
    (e) Sample images for Part 3
2. A README text file (not Word or PDF) that explains
    - Features of your program
    - Name of your entry-point script
    - Any useful instructions to run your script.
3. Your source files