16720J: Homework 3 - Object Detection

Luting Wang

October 26, 2015

1 Warming up with some theory (9pts)

Question 1.1 (2pts, 1 line)

The number of windows is:$(M-h)*(N-w)$.

Question 1.2 (5pts, 2-3 lines)

When the number of positive and negative samples differs greatly, using precision/recall metric is better than using accuracy metric for judging performance. Considering a extreme example: when we have 90 negative samples, and only 10 positive samples(actually it seldom occurs), if A classifier judges all samples negative, A would has an higher accuracy as 90%. However, B classifier judges 70 samples as negative in all negative samples, and judges 5 samples as positive in all positive samples, B would has an accuracy as 75%. But it is obvious that B classifier is more useful than A. So when we has negative and positive samples and their numbers are quite different, precision/recall metric has more valid reference value.

Question 1.3 (2 pts, 1 line)

Given 1000 bounding boxes in the training set for a single category, 1000 Exemplar detectors need to be trained, and only 1 Dalal-Trigges type template detectors need to be trained.

2 Object Detection via DPMs and Non-Maximum Suppression (40 pts)

2.1 Mean-Shift Clustering (20 pts)

Question 2.1.1 `MeanShift.m`

```
% Created by 15213796 on 2015-10-17.

function [CCenters,CMemberships] = MeanShift(data,bandwidth,stopThresh)

[numOfPoints, numOfDimPlusOne] = size(data);
numOfDim = numOfDimPlusOne -1;
numOfCluster = 0;          %the cluster number M
initPointsInd = 1:numOfPoints;   % the initial points
CCenters = [];        % center of each cluster
visited = zeros(1, numOfPoints);  %record the points which have been clusered
numOfInitPoints = numOfPoints;       % the initial point number to be ...
    clustered
CMemberships = zeros(1, numOfPoints);  %cluster membership
```

```
clusterVotes = zeros(1, numOfPoints);


while numOfInitPoints
    seedPoint = ceil((numOfInitPoints-1e-6)*rand);                    % ...
       pick up a rand point as seed point
    startPoint = initPointsInd(seedPoint);                          % use ...
       this point as start of mean
    myMean = data(startPoint, 1:numOfDim);                                ...
       %initial the mean to this point
    myMembers = [];     % the member of each cluster
    thisClusterVotes = zeros(1, numOfPoints);

    while 1              %utill convergence
        squareDistanceToAll = sum((repmat(myMean, numOfPoints, 1) - ...
           data(:, 1:numOfDim)).^2, 2);
        selectedInds = find(squareDistanceToAll < bandwidth^2);
        thisClusterVotes(selectedInds) = ...
           thisClusterVotes(selectedInds)+1;           % add vote to all ...
           the points belonging to this cluster
        myOldMean = myMean;  %save the old mean
        myMean = sum(data(selectedInds, ...
           1:numOfDim).*(repmat(data(selectedInds, numOfDimPlusOne),1, ...
           numOfDim)))./sum(data(selectedInds, numOfDimPlusOne));
        myMembers = [myMembers, selectedInds'];
        visited(myMembers) = 1;

        % stop this cluster
        if norm(myMean-myOldMean) < stopThresh
           %check for merge possibility
            mergeWith = 0;
            for i = 1:numOfCluster
                distanceToOther = norm(myMean - CCenters(i, :));
                if distanceToOther < bandwidth/2
                    mergeWith = i;
                    break;
                end
            end

            if mergeWith > 0
                CCenters(mergeWith, :) = 0.5*(myMean+ CCenters(mergeWith, ...
                   :));
                clusterVotes(mergeWith, :) = clusterVotes(mergeWith, :) + ...
                   thisClusterVotes;
            else
                numOfCluster = numOfCluster+1;
                CCenters(numOfCluster, :) = myMean;
                clusterVotes(numOfCluster, :) = thisClusterVotes;
            end
            break;
        end
    end
initPointsInd = find(visited == 0);
numOfInitPoints = length(initPointsInd);
end

[¬,CMemberships] = max(clusterVotes,[],1);  % a point belong to a cluster ...
   with the most votes
CMemberships = CMemberships';
end
```

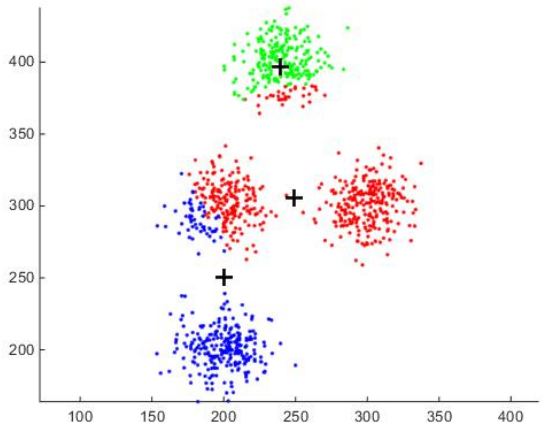

Figure 2: bandwidth=80, stopthresh=0.8

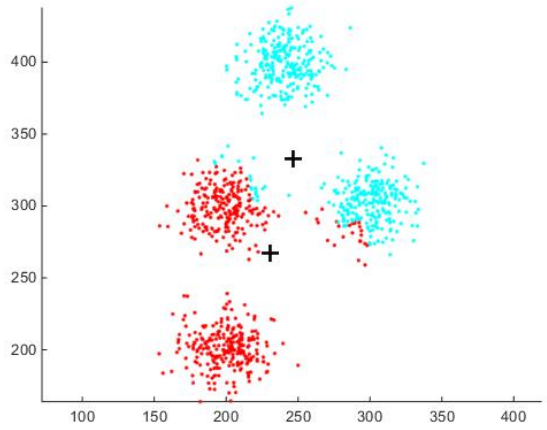

Figure 3: bandwidth=100, stopthresh=1

Question 2.1.2

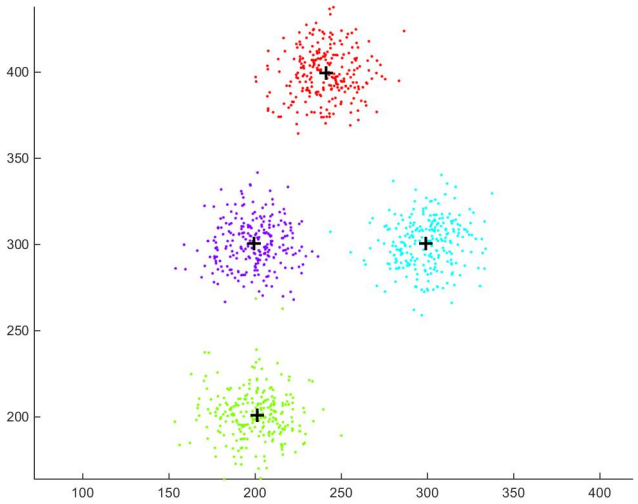

Figure 1: Mean Shift Clusters(bandwidth=30, stopthresh=0.3)

Question 2.1.3 (at most 3 lines in your write-up)

For low bandwidth(less than 20), based on the specific input data you give us, the points can't be clustered. And for high bandwidth(more than 80), we will get less clusters(3 or 2), which is against the original data distribution. When we use the weighted Mean-Shift algorithm to do clustering, we had better roughly know their distribution, eg. visualization of the input data. Then we can estimate the bandwidth(30) and stopthresh(0.3) to group like-modes. Please refer to Figure1, Figure2 and Figure3.

2.2 Detecting using Deformable Part Models (DPMs) (20 pts)

Question 2.2.1

Submit your `nms` function and include here

Figure 4: bandwidth=100, stopthresh=1, k=3

Figure 5: bandwidth=40, stopthresh=0.4, k=3

```
% Created by 15213796 on 2015-10-18.

function [refinedBBoxes] = nms(bboxes, bandwidth,K)

data = bboxes;
sizey = size(data, 2);
data(:, sizey) = data(:, sizey)+1;
stopThresh = 0.01*bandwidth;
[CCenters,CMemberships] = MeanShift(data,bandwidth,stopThresh);

 numOfClusters = size(CCenters, 1);

clusterWeight = hist(CMemberships,  numOfClusters)
[¬, index] = sort(clusterWeight, 'descend');

if K ≥ numOfClusters
    refinedBBoxes = CCenters;
else
    refinedBBoxes = CCenters(index(1:K), :);
end

end
```

Question 2.2.2 (at most 3 lines in your write-up)

After obtaining the clustered bounding boxes based on specific bandwidth, if K is larger than its number, I just choose all of them as top-K candidates(As you can see in Figure 4, when bandwidth is 100 and stopthresh is 1, I only get one clustered box.Although K is 3, the refinedBBoexes is only 1.). But if K is smaller than its number, I first sorted them based on the their "density", that to say, when I used Mean-Shift to cluster all the boxes and obtained their centers, each center has a cluster density, which represents how many points are clustered to the center. So I pick up the top-K densely boxes as final refinedBBoxes(As you can see in Figure 5, when bandwidth is 40 and stopthresh is 0.4, I can get 14 clustered boxes. And I choose the top 3 as refinedBBoexes.).

Question 2.2.3

(a) NMS Result 1

(b) NMS Result 2

(c) NMS Result 3

3 Reducing Exemplar Detectors (55 pts)

3.1 Detecting using Exemplar Detectors (10 pts)

Question 3.1.1 (10 pts)

```
% Created by 15213796 on 2015-10-16.

function [boundingBoxes] = batchDetectImageESVM(imageNames, models, params)

numCores=2;
imageDir = '../../data/voc2007';

try
    fprintf('Closing any pools...\n');
    parpool close;
catch ME
    disp(ME.message);
end

fprintf('Will process %d files in parallel to test the systems ...
    ...\n',length(imageNames));
fprintf('Starting a pool of workers with %d cores\n', numCores);
parpool('local',numCores);

L = length(imageNames);
boundingBoxes = cell(1,L);

parfor i=1:L
fprintf('Calculating the bounding boxes of the image %s\n', imageNames{i});
image = imread(fullfile(imageDir, imageNames{i}));
boundingBoxes{i} = esvm_detect(image, models, params);
end


fprintf('Closing the pool\n');
parpool close
```

3.2 Evaluating Detection Performance (15 pts)

Question 3.2.1 Theory (5 pts, 2 lines)

Average precision is the precision averaged across all values of recall between 0 and 1. And it is roughly the average area under the precision-recall curve for a set of queries. It provides a single figure measure of quality across recall levels.

Question 3.2.2 (10 pts)

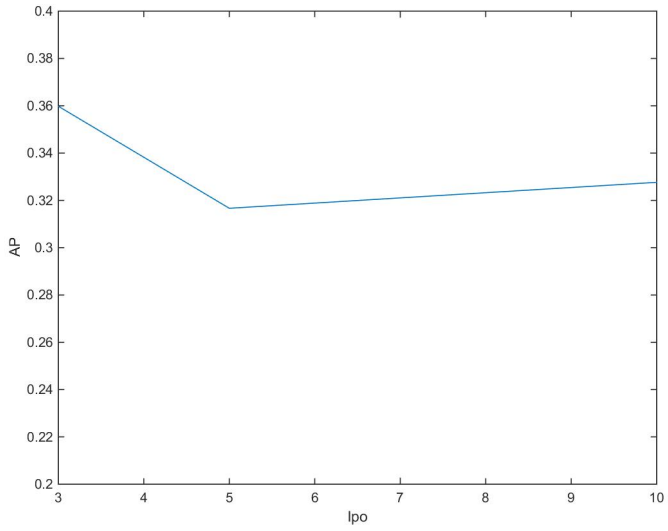

Figure 6: AP vs LPO (entire set of Exemplar)

```
% Created by 15213796 on 2015-10-18.

% Q3.2.2

addpath(genpath('../utils'));
addpath(genpath('../lib/esvm'));
load('../../data/bus_esvm.mat');
load('../../data/bus_data.mat');
params = esvm_get_default_params();

detect_levels_per_octave = [3, 5, 10];
ap = zeros(1, 3);
boundingBoxes = cell(1, 3);

for i = 1:3
params.detect_levels_per_octave = detect_levels_per_octave(i);
boundingBoxes{i} = batchDetectImageESVM(gtImages, models, params);
[¬,¬,ap(i)] = evalAP(gtBoxes,boundingBoxes{i},0.5);
end
% plot(detect_levels_per_octave, ap);
% xlim([3 10]);
% ylim([0.2 0.4]) ;
% xlabel('lpo');
% ylabel('AP');
% saveas(gcf,'q32_result.jpg')
```

As you can see, the number of scales influence the detector's performance. If we increase the number of scales, we would also introduce some fine but unnecessary information, and it would effect the detector's performance. Basically, 3 LPO is enough. And when LPO is 3, we could only get average precision as 0.36. I think it is because the training set contains some images which aren't "bus". Maybe we should reduce the number of exemplar detectors which can well represent "bus" features to obtain more accuracy.

3.3 Compacting the set of exemplar detectors

Question 3.3.1 (20 pts)

In this section, we need to compact the set of exemplar detectors while keeping the same average precision. First, I resize the bounding box to 100x100, and use all the 10000 pixels's filter response to do k-means. First, I choose K as 35. The average images for K=35 is shown below. Then I change K from 35 to 180 with a interval of 15(lpo=3), and see how average precision changes.

Figure 7: Average image for each clusters(K=35, without sampling)

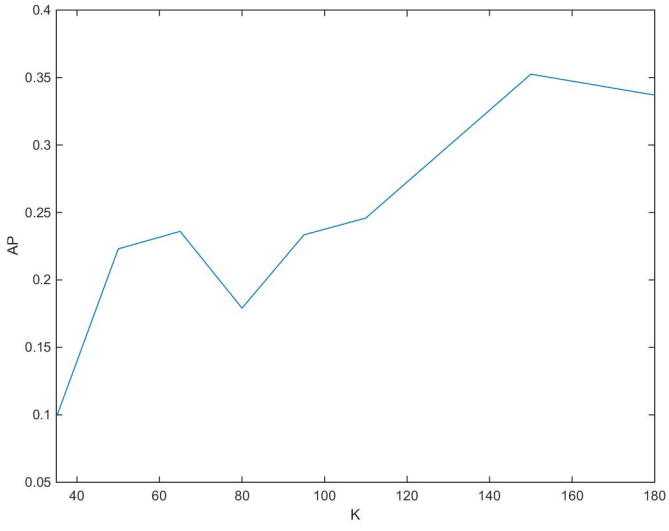

Figure 8: AP vs cluster number K (without sampling)

As you can see, when K is 35, the average precision is very low, only about 10%. It may because of the exemplars are not enough. As I increase K, the total trend of AP is increasing, and when K is 150, it is up to 36%, which is very close to the AP with all the models(229). So I can draw a conclusion that sometimes a smaller number of detectors is useful and perform as good as the entire set.

Then I use random 1000 pixels's filter response(sampling) to do k-means to compare with the first one, and draw a average image for K is 35 and a graph of AP vs.lpo. As you can see, when K is 35, the AP is about 16%, which is larger than the result if we don't do sampling. But as K is increasing, the increase rate is lower than the first test, and its maximum AP is lower than the first test. So I can conclude that when the number of exemplar is small, it is better to applying sampling to K-means. For other cases, there is no need to do sampling.

For more average images for different K with sampling and without sampling, please refer to the baselineESVM folder.

Figure 9: Average image for each clusters(K=35, with sampling)

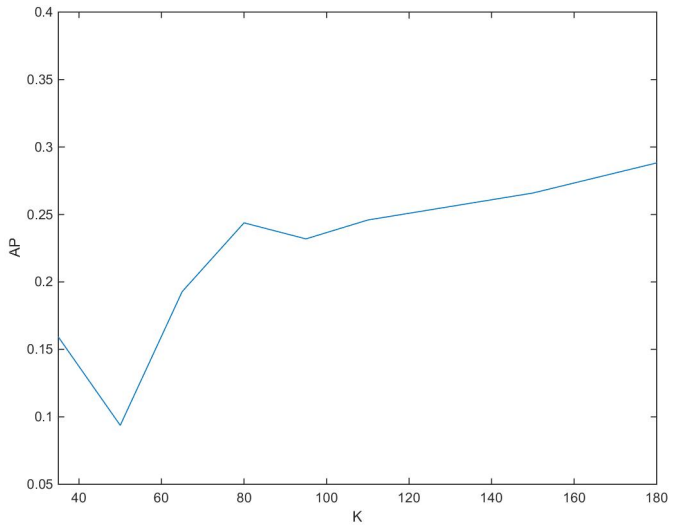

Figure 10: AP vs cluster number K(with sampling)

```
% Created by 15213796 on 2015-10-21.

% Q3.3.1

imageDir = '../../data/voc2007';
addpath(genpath('../utils'));
addpath(genpath('../lib/esvm'));
load('../../data/bus_esvm.mat');
load('../../data/bus_data.mat');

modelsize = length(models);
filterBank = createFilterBank();
alpha = 1000;
total_response = zeros(modelsize, length(filterBank)*3*alpha);
resizedimage = cell(1, modelsize);
response =  cell(1, modelsize);
sampledresponse =  cell(1, modelsize);


for i = 1:modelsize
    image = imread(fullfile(imageDir, models{i}.I));
    boundingbox =  models{i}.gt_box;
    boximage = ...
        image(boundingbox(2):boundingbox(4),boundingbox(1):boundingbox(3),:);
    resizedimage{i} = imresize(boximage, [100, 100]);
    response{i} = extractFilterResponses(resizedimage{i}, filterBank);
    pixels = randperm(alpha);
    sampledresponse{i}= response{i}(pixels, :);
    total_response(i, :) = reshape(sampledresponse{i}', 1, ...
        length(filterBank)*3*alpha);
end

K = 35;
[clusterindex_35, dictionary_35, ¬, distance_35] = kmeans(total_response, ...
    K, 'EmptyAction', 'drop');
[¬, detectorindex_35] = min(distance_35);
averageimage_35 = cell(1,K);

for i = 1: K
    array = find(clusterindex_35==i);
    totalimage = zeros(100,100,3);
    for j = 1 :size(array, 1)
        totalimage = totalimage + im2double(resizedimage{array(j)});
    end
    averageimage_35{i} = totalimage./size(array, 1);
end
imdisp(averageimage_35);
saveas(gcf,'average_image_35_sampled.jpg');

new_model_35 = models(detectorindex_35);
params = esvm_get_default_params();

L = length(gtImages);
new_boundingBoxes_35 = cell(1,L);
for i = 1:L
    fprintf('Calculating the bounding boxes of the image %s\n', gtImages{i});
    image = imread(fullfile(imageDir, gtImages{i}));
    new_boundingBoxes_35{i} = esvm_detect(image, new_model_35, params);
end

```

```
58 [¬,¬,ap_35] = evalAP(gtBoxes,new_boundingBoxes_35,0.5);
```

Question 3.3.2 (10 pts)

In this section, I try to use hog feature to do K-means, and pick the closest exemplars to compact the number of detector. First I choose K as 35 and draw the average image for 35 clusters, as shown below. Then I change the K from 35 to 180, and plot AP with different K. As you can see, when K is 35, the average image is more average than the one using filter response to do clustering. And when K is 180, we can get AP as high as 39%, which is higher than when we use filter response.

Figure 11: Average image for each clusters(K=35, using hog features)

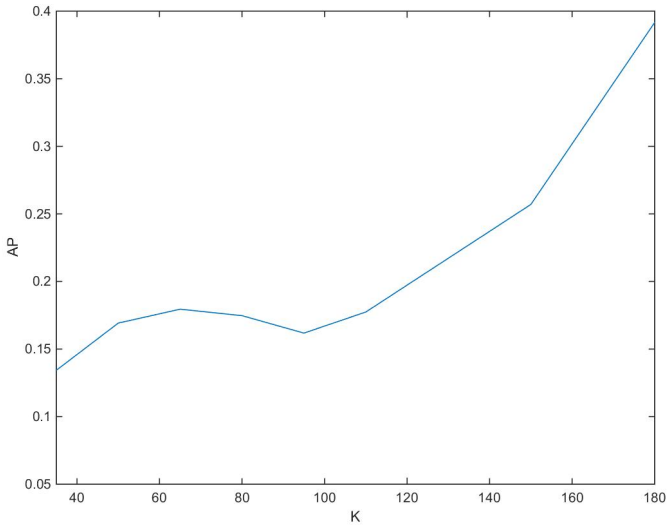

Figure 12: AP vs cluster number K

your code path here

```
% Created by 15213796 on 2015-10-21.

% Q3.3.2

imageDir = '../../data/voc2007';
addpath(genpath('../utils'));
addpath(genpath('../lib/esvm'));
load('../../data/bus_esvm.mat');
load('../../data/bus_data.mat');


modelsize = length(models);
uniform_hog_feature = cell(1, modelsize);
alpha = 1000;
total_response = zeros(modelsize, alpha);
for i = 1:modelsize
    hog_feature = models{i}.model.w;
    n = size(hog_feature, 1)*size(hog_feature, 2)*size(hog_feature, 3);
    uniform_hog_feature{i} = reshape(hog_feature, 1, n);
    pixels = randperm(n);
    total_response(i, :)  = uniform_hog_feature{i}(1, pixels(1:alpha));
end


K = 35;
[clusterindex_35, dictionary_35, ¬, distance_35] = kmeans(total_response, ...
    K, 'EmptyAction', 'drop');
[¬, detectorindex_35] = min(distance_35);
averageimage_35 = cell(1,K);

for i = 1: K
    array = find(clusterindex_35==i);
    totalimage = zeros(100,100,3);
    for j = 1 :size(array, 1)
        totalimage = totalimage + im2double(resizedimage{array(j)});
    end
    averageimage_35{i} = totalimage./size(array, 1);
end
imdisp(averageimage_35);
saveas(gcf,'average_image_35_332.jpg');

new_model_35 = models(detectorindex_35);
params = esvm_get_default_params();

L = length(gtImages);
new_boundingBoxes_35 = cell(1,L);
for i = 1:L
    fprintf('Calculating the bounding boxes of the image %s\n', gtImages{i});
    image = imread(fullfile(imageDir, gtImages{i}));
    new_boundingBoxes_35{i} = esvm_detect(image, new_model_35, params);
end

[¬,¬,ap_35] = evalAP(gtBoxes,new_boundingBoxes_35,0.5);
```

4 Extra credit: Segmentation transfer using ESVM (20 pts)

If you have attempted this extra-credit section please include a summary of your efforts here and include all relevant work in the folder `segTransfer`.