IN 4320 Machine Learning Assignment 6

Xiangwei Shi
4614909

1 The Naive MIL classifier

(b)

In this part, I created a function named **read_images**, which all images from a given directory. The code is shown below.

```matlab
function images_matrix = read_images(directory)
images_addres = [directory,'\*.jpg'];
images = dir(images_addres);
images_num = size(images);
images_matrix = cell(images_num);
for i = 1:images_num
    images_matrix{i,1} = ...
        imread(strcat(images(i).folder,'\',images(i).name));
end
end
```

(c)

In this part, a function named **extractinstances** is implemented. And the matlab code is shown below.

```matlab
function resulting_features = extractinstances(image,width)
%%
% This function is for 1(c) of Multiple Instance Learning: image
% classification of In4320 Machine Learning Computer Exercise.
% Segment an image, compute the average red, green and blue color per
% segment
% use im_meanshift to segment an image
im = im_meanshift(image,width);
% plot the segments to find the width parameter
% imshow(im,[]);
seg_num = size(unique(im),1);
resulting_features = zeros(seg_num,3); % 3 channels: RBG
% compute mean for each segment
segment_cell = cell(0);
segment_idx = [];
segment_num = [];
```

```
img = im2double(image);
for i = 1:size(im,1)
    for j = 1:size(im,2)
        if ¬ismember(im(i,j),segment_idx)
            segment_idx = [segment_idx im(i,j)];
            segment_num = [segment_num 1];
            segment_cell{size(segment_cell,2)+1} = reshape(img(i,j,:),1,3);
        else
            idx = find(segment_idx == im(i,j));
            segment_cell{idx} = segment_cell{idx} + ...
                reshape(img(i,j,:),1,3);
            segment_num(idx) = segment_num(idx) + 1;
        end
    end
end
% return the resulting_features matrix
for i = 1:seg_num
    segment_cell{i} = segment_cell{i}./segment_num(i);
    for j = 1:3 % rgb channels
        resulting_features(i,j) = segment_cell{i}(1,j);
    end
end
end
```

As shown above, there is an input, **width**, along with image input. By comparing the results with different values of **width**, the value of this parameter is set as 30, which enables to separate the background from the foreground, as shown in 1 below. And I also find that it takes more time with small value of width than that with big value of width.

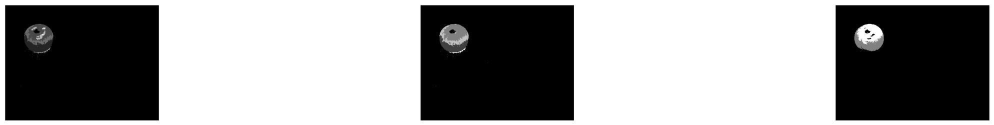
(a) width = 15 (b) width = 30 (c) width = 45

Figure 1: Results with different values of width

(d)

In this part, a function named **gendatmilsival** is created to generate a MIL datasets. The dataset is created by using the instances extracted above. And label 1 and 2 represent the instances of apple and banana, respectively. And the codes of Matlab are shown below.

```
function bag_dataset = gendatmilsival(directory)
%%
bags_data = cell(0);
bags_label = [];
width = 30;
% apple images
apple_directory = [directory,'\apple'];
apple_images_matrix = read_images(apple_directory);
```

```
for i = 1:size(apple_images_matrix,1)
    instances = extractinstances(apple_images_matrix{i},width);
    bags_data{i,1} = instances;
end
% banana images
banana_directory = [directory,'\banana'];
banana_images_matrix = read_images(banana_directory);
for i = 1:size(banana_images_matrix,1)
    instances = extractinstances(banana_images_matrix{i},width);
    bags_data{i+size(banana_images_matrix,1),1} = instances;
end
% label
bags_label = ...
    [ones(size(apple_images_matrix,1),1);2*ones(size(banana_images_matrix,1),1)];
% storing them in a Prtools dataset
bag_dataset = bags2dataset(bags_data,bags_label);
end
```

There are 60 apple images and 60 banana images in the 'sival_apple_banana' folder. Therefore, I obtained 120 bags. And all of the instances have 3 features, since each image has three channels (RGB). For per bag, there are 2 to 9 different instances in general, which is determined by the algorithm of segmentation. And in total, I get 663 instances.
To see if the instances from the two classes are separable or not, a scatterplot is shown below as Figure 2. Although some blue points are separated from the red ones, most of the points are not separable. It is difficult to separate the classes. We can also find that many points are centering on the diagonal, which may because these instances are related to the background of the images and the background in both classes are similar.

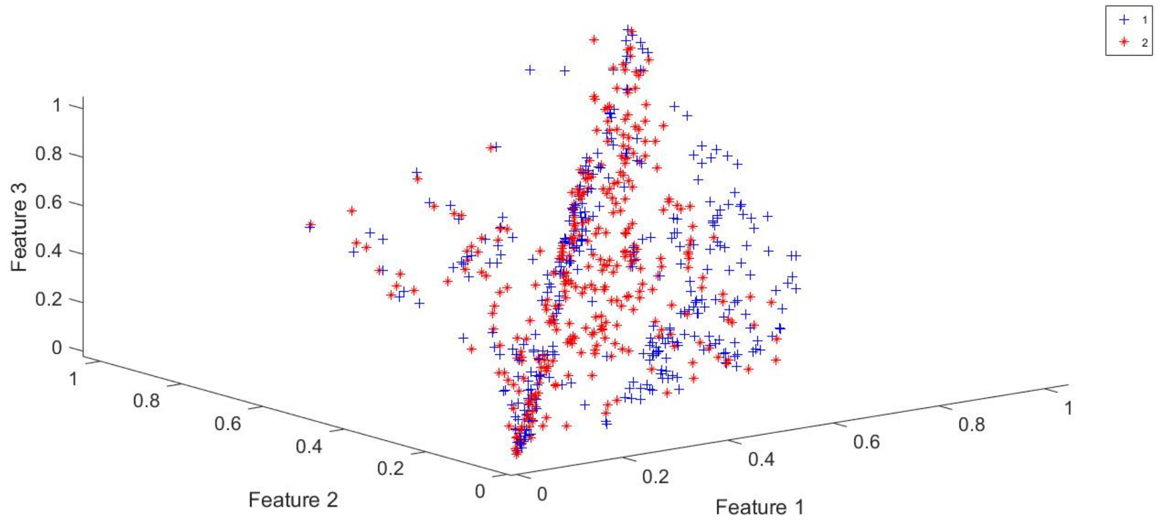

Figure 2: Scatterplot of instances of the apple & banana dataset

(e)

In this part, a function named **combineinstlabels** is created to accept a list of labels and output a single label obtained by majority voting. And if all labels appear with the same frequency, the output still is single. In our case, the single output label will represent apple when the frequencies of apple and banana labels are the same.

```
function out_label = combineinstlabels(list_label)
labels = unique(list_label);
num = length(labels);
best_label_num = 0;
for i = 1:num
    if best_label_num < sum(list_label==labels(i))
        out_label = labels(i);
        best_label_num = sum(list_label==labels(i));
    end
end
end
```

(f)

In this part, a Fisher classifier is trained in two ways, not trustworthy one and trustworthy. Firstly, I trained a Fisher classifier in not trustworthy way. In this training procedure, I used all instances extracted from (d) as training dataset. And then applied the trained classifier to each instance in a bag, classified the instances and combined the label outputs using **combineinstlabels** from (e). And the code is shown below. In this scenario, 11 apple images are misclassified as banana images, and 16 banana images are misclassified as apple images. Since the same dataset was used to train and test classifier. To estimate the classification error in a trustworthy way, the dataset should be split to training and test dataset, which was accomplished below.

```
%% not trustworthy
instance = bag_dataset.data;
fisher = fisherc(bag_dataset);
start = 1;
predict = [];
for i = 1:size(bags_data,1)
    instance_length = size(bags_data{i},1);
    instance_bag = instance(start:start+instance_length-1,:);
    start = start+instance_length;
    result = labeld(instance_bag,fisher);
    out_label = combineinstlabels(result);
    predict = [predict; out_label];
end
residule = bags_label - predict;
% apple misclassified as banana
apple_error = sum(residule == -1);
% banana misclassified as apple
banana_error = sum(residule == 1);
```

Secondly, I estimated the classification error in a trustworthy way. In this scenario, I split the bag dataset into training and test dataset with a ratio as 0.8. Then I used the IDs of bags to get the instances of each bag. And the code is shown below. And I repeated this training and test procedures for ten times. There were 2.8 apple images misclassified into banana images, and 3.4 banana images were misclassified as apple images on average.Since there are 12 test images in each class, the average test error is 0.2583.

```matlab
%% trustworthy
% split the dataset with bags ID, and then use instances of related ...
    bags ID
% to train and test the classifier
bag_id = [1:size(bags_data,1)]';
dataset_bag = prdataset(bag_id,bags_label);
error = cell(2,10);
for j = 1:size(error,2)
    [train_bag,test_bag,train_bag_id,test_bag_id] = ...
        gendat(dataset_bag,0.8);
    train_instance_id = [];
    test_instance_id = [];
    for i = 1:size(train_bag,1)
        temp = find(getident(bag_dataset,'milbag')==train_bag_id(i));
        train_instance_id = [train_instance_id;temp];
    end
    train_instance = bag_dataset(train_instance_id,:);
    fisher = fisherc(train_instance);
    predict = [];
    for i = 1:size(test_bag,1)
        temp = find(getident(bag_dataset,'milbag')==test_bag_id(i));
        result = labeld(bag_dataset.data(temp,:),fisher);
        out_label = combineinstlabels(result);
        predict = [predict; out_label];
    end
    test_label = test_bag.labels;
    residule = test_label - predict;
    %apple error
    error{1,j} = sum(residule == -1);
    %banana error
    error{2,j} = sum(residule == 1);
end
```

(g)

1. There are only 120 images in total, which are not enough for this multi-instance classification. Therefore, the first approach to improve the performance of the classifier could be increase the size of data. One possible way to do this is image augmentation.

2. In this classification problem, we only extract three dimensions of features of images

(RGB channels). Therefore, the number of dimensions of features could not enough for this classification. Likewise, the number of features of images could not be optimal (the number of instance for each bag of image). In order to increase the dimensions of feature, we could not only use RGB channels but also HSV channels. And when we extract instances using **mean_shift** function, we should regard the width as a parameter and tune this parameter by plotting the learning curve, training error and test error as functions of this parameter.

3. Another method to improve the performance of classifier could be change the classifier. Fisher classifier could not be the optimal classifier for this classification problem. Another classifier, such as SVM with kernels could be more suitable.

4. In this case, we use majority voting to decide the label for the instance from the same bag, which may not be the optimal way. We can try to apply the probability.

2 MILES

(a)

The function **bagembed** is implemented as below. For all apple-banana images, the size of feature_vector is 120×663.

```matlab
function feature_vector = bagembed(bag_dataset,instance_list)
num_bag = size(bag_dataset,1);
num_instance = size(instance_list,1);
sigma = 25;
feature_vector = zeros(num_bag,1);
for i = 1:num_bag
    best = 0;
    for j = 1:num_instance
        temp = ...
            exp((-1/sigma^2)*sum((bag_dataset(i,:)-instance_list(j,:)).^2));
        if temp>best
            best = temp;
        end
    end
    feature_vector(i) = best;
end
end
```

(b)

We choose $\sigma = 25$, which satisfies that not all numbers become 0 or 1. And we train and test this dataset in two scenarios.

(c)

Firstly, we used all the dataset as training and test data, and there are only 6 images were misclassified on average (10 times), where the error rate is 0.05. Secondly, I split the whole dataset as training and test data with a ratio of 0.8. In this scenario, there are 2.3 images were misclassified, where there were 24 test images and the test error is 0.0958. Clearly, this classifier is much better than the Naive MIL classifier, which is maybe because this method uses all the bag and instance representation and the Naive one only uses bags of several instances to train the classifier. To improve the performance of this MILES classifier, we can firstly increase the size of training dataset. And since the number of dimensions is large, performing features selection or features extraction before training could be an option. At last, we can also change different classifier instead of LIKNON classifier.

3 Another MIL classifer

In this section, I implement another MIL classifier, which is called Bayesiann-kNN or Citation-kNN. And it is proposed in [1]. In [1], Hausdorff Distance was used to measure the distance between two subsets of a metric space, which is defined as: $H(A,B) = \max h(A,B), h(B,A)$, where $h(A,B) = \max_{a\in A} \min_{b\in B} ||a-b||$. And the authors of [1] modified the measurement of the distance and took the k-th ranked distance rather than the maximum, $h_k(A,B) = kth_{a\in A} min_{b\in B}||a-b||$. Therefore, k nearest neighbors (reference) should be found first, and a citation approach is also applied (the bags that cite the new bag before their j-th nearest neighbor). And then a majority vote is used between reference and citers. If they are the same, the negative label will be chosen.
The Matlab code is shown below. And function **HausdorffDist** is used from link.

```
% load('matlab.mat');
datset_bag = prdataset(bags_data,bags_label);
iteration = 10;
error = zeros(iteration,1);
k =3;
for j = 1:iteration
    % split training and test dataset
    [train_bag,test_bag,train_bag_id,test_bag_id] = ...
        gendat(dataset_bag,0.8);
    train_instance_id = [];
    test_instance_id = [];
    for i = 1:size(train_bag,1)
        temp = find(getident(bag_dataset,'milbag')==train_bag_id(i));
        train_instance_id = [train_instance_id;temp];
    end
    train_instance = bag_dataset(train_instance_id,:);
    for i = 1:size(test_bag,1)
        temp = find(getident(bag_dataset,'milbag')==test_bag_id(i));
        test_instance_id = [test_instance_id;temp];
    end
    test_instance = bag_dataset(test_instance_id,:);
    % distance matrix of the training data
    distance_matrix = zeros(size(dataset_bag,1),size(dataset_bag,1)-1);
    for m = 1:size(train_instance,1)
        k = 1;
        for n = 1:size(train_instance,1)
            if i≠j
                distance_matrix(i,k) = ...
                    HausdorffDist(train_instance{i},train_instance{j});
                k = k+1;
            end
        end
    end
    % test
    test_error = 0;
    for m = 1:size(test_instance,1)
        output = ...
            predict(train_instance,distance_matrix,unique(getident(train_instance,'milba
        true_label = test_instance.labels;
        if output≠true_label(i)
            test_error = test_error +1;
        end
    end
    test_error = test_error/size(test_instance,1);
end
```

```
function output = predict(dataset, ...
    distance_matrix,bag_id,test_instance,k,R,C)
% This function labels a new bag
bags_num = size(dataset,1);
idx_bags = unique(getident(dataset,'milbag')),1);
list_dist = [];
% test data distances
for i = 1:bags_num
    res = ...
        HausdorffDist(test_instance,dataset(find(getident(dataset,'milbag') ...
        == idx_bags(i)),:),k);
    list_dist = [list_dist res];
end
% compute references
[¬,max_id] = sort(list_dist,'ascend');
id_r = [];
for i = 1:R
    list_r_idx = find(getident(dataset,'milbag')==bag_id(max_id(i)));
    id_r = [id_r list_r_idx];
end
r_lab = dataset(id_r,:).labels;
% compute citers
c_lab = [];
for i=1:idx_bags
    res = ...
        HausdorffDist(dataset(find(getident(dataset,'milbag')==idx_bags(i)),:),test_insta
    idx = find(test_instance==bag_id(i));
    list_dist = cat(2,distance_matrix(idx,:),res);
    [max_value,max_id] = sort(list_dist,'ascend');
    rank = find(max_value==res);
    if rank≤c
```

```
        id_r = find(getident(dataset,'milbag')==(idx_bags(i)));
        ref = getlab(dataset(id_r(1),:));
        c_lab = [c_lab;ref];
    end
end
% label
nb_1=0;
nb_2=0;
for i = 1:size(c_lab,1)
    if c_lab(i) == 1
        nb_1 = nb_1+1;
    else
        nb_2 = nb_2+1;
    end
end
for i = 1:size(r_lab,1)
    if r_lab(i) == 1
        nb_1 = nb_1 +1;
    else
        nb_2 = nb_2+1;
    end
end
if nb_2>nb_1
    output = 2;
else
    output = 1;
end
end
```

I ran the training and test procedure for ten times, and also test different parameters to get better result. I find that the third maximum of the distance will result the smallest error, which is 0.15208 (7.3 images were misclassified). From the results, we can find that this classifier is better than Naive classifier but worse than MILES classifier. One possible reason could be the classifier depends on the distance of bags which is up to the instances for the whole image but the instances with similar background will result the difficulty for classification, such as the points on the diagonal in Figure 2.

References

[1] J. Wang and J.-D. Zucker, “Solving multiple-instance problem: A lazy learning approach,” 2000.