IN4320 Machine Learning

Reinforcement Learning

07 June 2019

TU Delft
Coordinated by Prof David Tax

Matthijs Bekendam 4725751 J.M.Bekendam@student.tudelft.nl

Contents

1 Question 1

1.1 b

Different width parameters were tested (see Figure 3) and a width parameter of 0.1 was selected to work with.

Over segmenting takes you closer to standard supervised learning since eventually (in the most extreme case) each pixel would be a segment. Under segmenting has a chance of losing important information about the object we are interested in. Hence, it would be better to over segment (ignoring computational load). Also evident from Figure 3, smaller segments tend to yield a lower error rate.

1.2 c

The amount of bags obtained is the same as the amount of pictures, 120 The amount of features per instance is 3, characterised by the RGB color values. The amount of instances per bag are shown in Figure 1.

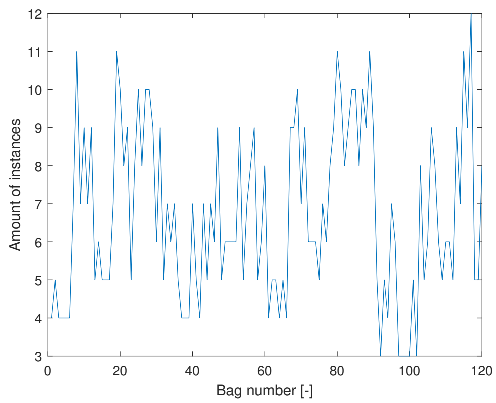
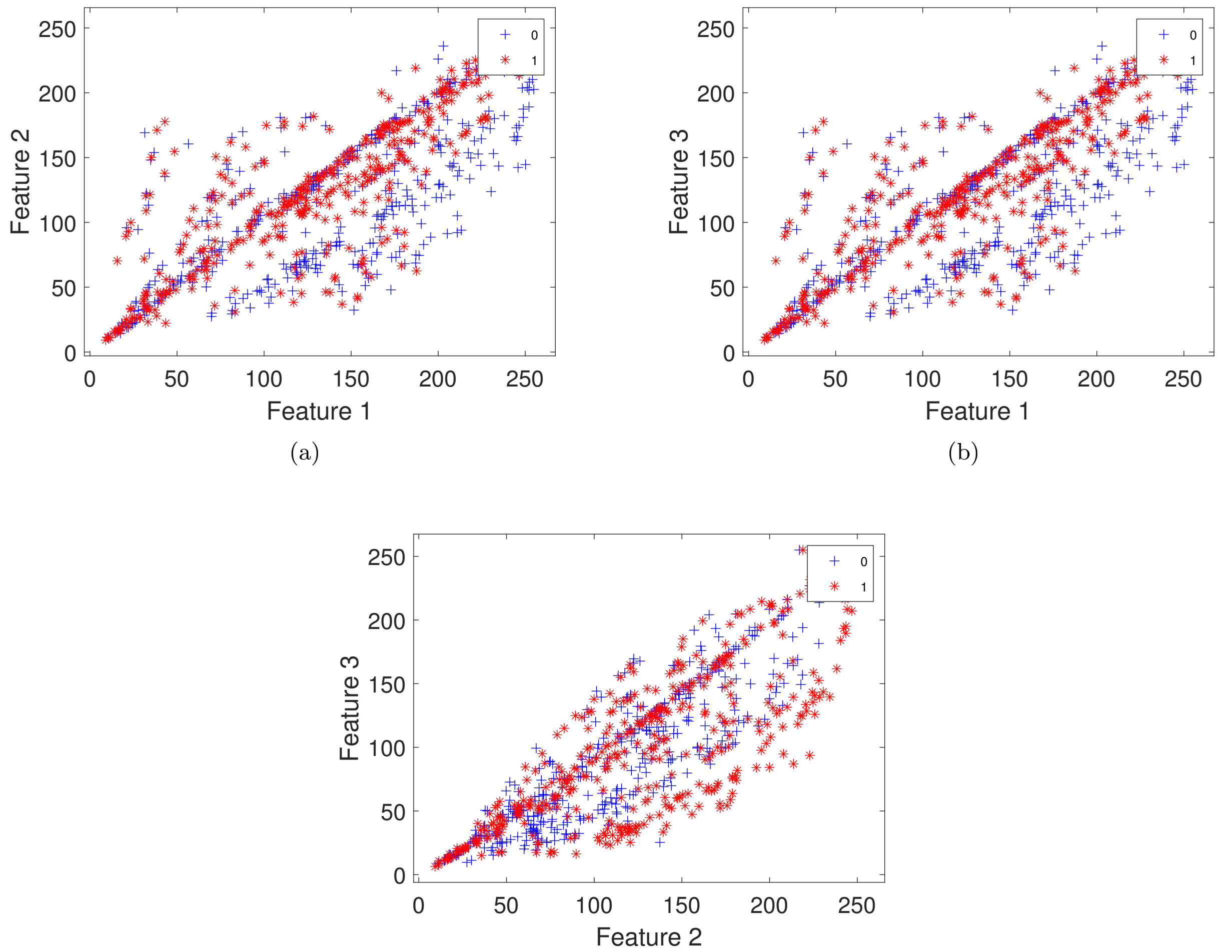


Figure 1: Bag number vs amount of instances.

Instances vary from 3 till 11, these values are depend on the width parameters selected. Larger width parameters yield viewer instances per bag.



(a) (b) (c)

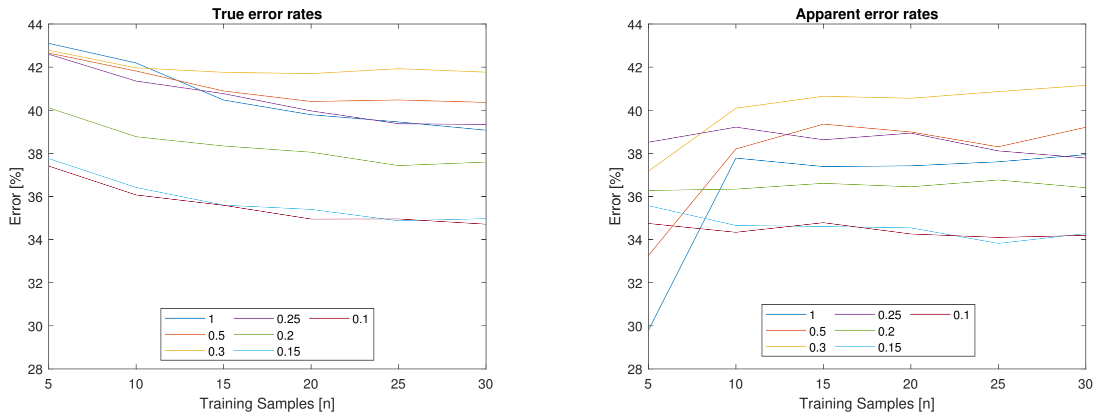
Figure 2: All features plotted against one another to see correlations in- and separability off the data.

From Figure 2 it became evident that the data is not separable.

1.3 e

21 apples and 12 bananas were miss-classified, resulting in an error of 27.5 %

The entire data set is used to train the data and is then used to test the error rates, this does not yield a realist error rates (=apparent error). Furthermore, multiple iterations should be ran using different training samples to ensure convergence of the error rates. To get a more trustworthy error, one should calculate the true error and run multiple iterations. This is done by separating the data in a training and a test set. The classifier will be trained on the training set and tested on the test set.



(a) True errors.

(b) Apparent errors.

Figure 3: Apparent- and True Error rates vs. Training set sizes. The lines denote different values for the width parameter. 200 iterations were performed to ensure stable convergence of the error rates.

1.4 f

To improve the performance of the classifier the following two methods are proposed:

- Add weights to instances
- Add soft labels.

Weights could be added to the instances, making certain colors more important. For example, for the classification of apples, one could give red colors more importance than blue colors. Soft labels could be added such that there is a distinction between certain environments, for example when classifying tigers, a jungle background is more of an indicator for tigers than an office building.

2 Question 2

Equation 2.1 (from [2]) shows that:

$$\mathbf{m}\left(\mathbf{B}_i\right) = \left[s\left(\mathbf{x}^1, \mathbf{B}_i\right), s\left(\mathbf{x}^2, \mathbf{B}_i\right), \cdots, s\left(\mathbf{x}^n, \mathbf{B}_i\right)\right]^T \tag{2.1}$$

our feature vector will be as large as the amount of instances ($\mathbf{x}^n$) in our problem. This will be 313 using all data as training set.

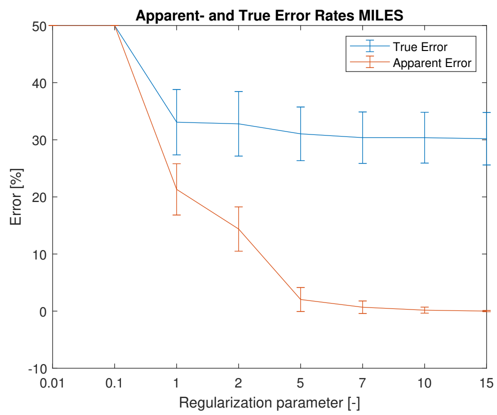


Figure 4: Miles Apparent error rates vs Regularization term. A sigma value of 12 was used and a training set size of 20 was used.

As denoted in Figure 4, this Miles classifier can perform better when using the correct regularization term. The performances is also depended on the sigma value used. This is a scaling term for the features and since the MILES works with a distance norm this influences the performance.

Looking at the true error of both MILES and the naive MIL classifier, the MILES out performs the naive classifier.

3 Question 3

The third MIL classifier implemented was the MISVM classifier. As a reference, [1] was used.

Build Classifier:
For each bag B
    If B is a positive bag
        Initialize class label for each instance x_i within B as $y_i = 1$;
    Else
        Initialize the label of each instance x_i within B as $y_i = 0$;
Do
    Build standard single-instance SVM model based on the labeled data;
    For each positive bag B^+
        For each single-instance x_i within B^+
            Compute SVM output $f(x_i) = \sum_j \alpha_j K(x_j, x_i) + b$;
            If $(f(x_i) \leq 0)$ $\quad y_i = 0$;
            Else $y_i = 1$;
        If $(\sum_i y_i == 0)$ //no positive classification
            Find instance x_{i^*} within bag B^+ where $i^* = \arg\max_i f(x_i)$;
            Set $y_i^* = 1$;
While (single-instance labels have been changed)

Classify:
Initialize *distribution*;
For each single-instance x_i within the unknown bag
    Compute $f(x_i) = \sum_j \alpha_j K(x_j, x_i) + b$;
    If $(f(x_i) \leq 0)$ $\quad y_i = 0$;
    Else $y_i = 1$;
If $(\sum_i y_i == 0)$
    $distribution[0] = 1.0$; // predicted as a negative bag
Else
    $distribution[0] = 0$; // predicted as a positive bag
$distribution[1] = 1 - distribution[0]$;
return *distribution*;

Figure 5: MISVM pseudo-code from [1].

The MATLAB adaptation of this code can be reviewed in Section 4.6.

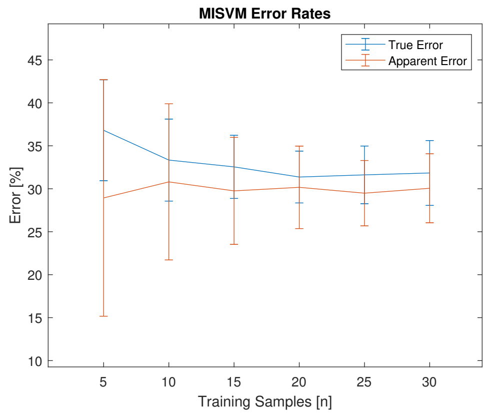


Figure 6: MISVM True- and Apparent error rates vs training set size. 200 iterations were performed for stable convergence of the error.

Figure 6 reveals that MISVM outperforms the MILES by a small margin. Both these classifiers do better than the naive classifier.

4 CODE

4.1 Extractinstances

Listing 1: Code Extractinstances

```
function [output,bag_label] = extractinstances(img_scaled,img_normal,width)

shifted_img = im_meanshift(img_scaled,width);
max_segments = max(max(shifted_img));
RGB_segments = zeros(max_segments,3);
colorcount = zeros(max_segments,3) ;
bag_label = linspace(1,max_segments,max_segments);

for i = 1:768
     for j = 1:1024
        for z = 1:max_segments
            if shifted_img(i,j) == z
                for color = 1:3
                    RGB_segments(z,color) = RGB_segments(z,color) + double(img_normal(i,j,color));
                    colorcount(z,color) = colorcount(z,color) + 1;
                end

            end


        end

     end
end

output = RGB_segments ./ colorcount;

end
```

4.2 Gendatmilsival

Listing 2: Code Gendatmilsival

```
labels_apple = zeros(60,1)
labels_banana = ones(60,1)

bags_apple = RBG_segmented_apples
bags_banana = RBG_segmented_banana

data = {bags_apple bags_banana}
bags = cat(1,data{:})
label_data = [labels_apple;labels_banana]

datapr = bags2dataset(bags,label_data)

apple_prdata = bags2dataset(bags_apple,labels_apple)
banana_prdata = bags2dataset(bags_banana,labels_banana)
```

4.3 combineinstlabels

Listing 3: Code combineinstlabels

```
function [ baglab_predicted ] = combineinstlabels( instlabel_inbag )

numel0 = numel(find(instlabel_inbag == 0));
numel1 = numel(find(instlabel_inbag == 1));

if numel0 > numel1
    baglab_predicted = 0;
else
    baglab_predicted = 1;
end

end
```

4.4 bagembed

Listing 4: Code bagembed

```
function [ dist ] = bagembed( bag, all_instances, sigma )

for i = 1:size(all_instances,1)
    for j = 1:size(bag,1)
        all_dist(i,j) = exp(-norm(bag(j,:)-all_instances(i,:))/sigma^2);
        if all_dist(i,j) == 1
            all_dist(i,j) = 0;
        end
    end

    dist(i,1) = max(all_dist(i,:));
end

end
```

4.5 Question 3: MISVM

Listing 5: Code MISVM

```
%% MISVM

samplez=[5,10,15,20,26,29];
counter=0;
counter3 = 0;
max_con = 150;
error_MISVM_true = zeros(length(samplez),1);
error_MISVM_ap = zeros(length(samplez),1);

error_MISVM_true_std = zeros(max_con,length(samplez));
error_MISVM_ap_std = zeros(max_con,length(samplez));


for train = 1:length(samplez)
for conv = 1:max_con
clear instances_all
[bags_apple_test,bags_apple_train,bags_banana_test,bags_banana_train] =
    traintestsplit(samplez(train),RBG_segmented_apples,RBG_segmented_banana);
```

```
train
counter = 0;
counter3 = 0;
for i = 1:length(bags_apple_train)

    [n,~]=size(bags_apple_train{i}(:,:));

    for j = 1:n
        counter = counter + 1;
        counter3 = counter3 + 1;

        instances_all(counter3,:)=bags_apple_train{i}(j,:);

    end

end

counter2 = 0
for i = 1:length(bags_banana_train)

    [n,~]=size(bags_banana_train{i}(:,:));

    for j = 1:n
        counter3 = counter3 + 1;
        counter2 = counter2 + 1;

        instances_all(counter3,:)=bags_banana_train{i}(j,:);

    end

end

iteration = 20000;
label_svm_array = ones(counter2,iteration);

%loop here with k
k = 0;
check = true

while check
k = k+1
if k > 50
    check = false
end
SVMModel = fitcsvm(instances_all,[zeros(counter,1);label_svm_array(:,k)]);
Beta = SVMModel.Beta;
f = @(x) x(1)*Beta(1)+x(2)*Beta(2)+x(3)*Beta(3) + SVMModel.Bias;
Labels_svm = cell(length(bags_banana_train),1);

for i = 1:length(bags_banana_train)
      [n,~]=size(bags_banana_train{i}(:,:));
      y_temp = zeros(n,1);
      y_val = zeros(n,1);
    for j = 1 : n
        y_temp(j,1) = f(bags_banana_train{i}(j,:));
        if y_temp(j,1) <= 0
            y_val(j,1) = 0;
        else
            y_val(j,1) = 1;
```

```


        end
    end

    if sum(y_val) == 0
        [val, idx] = max(y_temp);
        y_val(idx) = 1;
    end

    Labels_svm{i} = y_val;
end
%create normal array for labels
counter4 = 0;
for i = 1:length(bags_banana_train)

    [n,~]=size(Labels_svm{i}(:,:));

    for j = 1:n
        counter4 = counter4 + 1;
        label_svm_array(counter4,k+1)=Labels_svm{i}(j,1);
    end

end
if k > 1
    if sum(abs(label_svm_array(:,k-1) - label_svm_array(:,k))) == 0 ;
       final_label_misvm = label_svm_array(:,k+1);
       check = false
    end
end

end

f = @(x) x(1)*Beta(1)+x(2)*Beta(2)+x(3)*Beta(3) + SVMModel.Bias;

%moving on to classification
%True
misvm_labels_t = zeros(length(bags_apple_test)+length(bags_banana_test),1);
data1 = {bags_apple_test bags_banana_test};
bags1 = cat(1,data1{:});
label_data_t = [zeros(length(bags_apple_test),1);ones(length(bags_banana_test),1)];

for p = 1:length(misvm_labels_t)
        [n,~]=size(bags1{p}(:,:));
        labels_temp = zeros(n,1);
    for z = 1:n
        if f(bags1{p}(z,:)) <= 0
             labels_temp(z,1) = 0;
        else
             labels_temp(z,1) = 1;
        end
    end

    if sum(labels_temp) == 0
        misvm_labels_t(p,1) = 0;
    else
        misvm_labels_t(p,1) = 1;
    end
end
```

```

error_MISVM_true(train,1) = error_MISVM_true(train,1) +
    sum(abs(label_data_t-misvm_labels_t))/length(misvm_labels_t);
error_MISVM_true_std(conv,train) = sum(abs(label_data_t-misvm_labels_t))/length(misvm_labels_t);

%moving on to classification
%aparrent
misvm_labels = zeros(length(bags_apple_train)+length(bags_banana_train),1);
data2 = {bags_apple_train bags_banana_train};
bags2 = cat(1,data2{:});
label_data = [zeros(length(bags_apple_train),1);ones(length(bags_banana_train),1)];

for p = 1:length(misvm_labels)
       [n,~]=size(bags2{p}(:,:));
       labels_temp = zeros(n,1);
   for z = 1:n
       if f(bags2{p}(z,:)) <= 0
           labels_temp(z,1) = 0;
       else
           labels_temp(z,1) = 1;
       end
   end

   if sum(labels_temp) == 0
       misvm_labels(p,1) = 0;
   else
       misvm_labels(p,1) = 1;
   end
end

error_MISVM_ap(train,1) = error_MISVM_ap(train,1) +
    sum(abs(label_data-misvm_labels))/length(misvm_labels);
error_MISVM_ap_std(conv,train) = sum(abs(label_data-misvm_labels))/length(misvm_labels);


end
end

error_MISVM_ap = error_MISVM_ap/max_con
error_MISVM_true = error_MISVM_true/max_con
```

4.6 Traintestsplit

Listing 6: Code Traintestsplit

```

function [bags_apple_test,bags_apple_train,bags_banana_test,bags_banana_train] =
    traintestsplit(k,RBG_segmented_apples,RBG_segmented_banana)

labels_apple = zeros(60,1);
labels_banana = ones(60,1);

%k = [5]


for z = 1:length(k)

```

```
    p = randperm(60,k(z));
    p=sort(p);
    notp = zeros(length(p)-k(z),1);
    counter = 1;

    for o = 1:60
        if isempty(find(p(:,:)-o == 0))
            notp(counter,1) = o;
            counter = counter +1;

        end
    end
    notp=sort(notp);
    bags_apple_train = cell(k(z),1);
    bags_apple_test = cell(60-k(z),1);

    bags_banana_train = cell(k(z),1);
    bags_banana_test = cell(60-k(z),1);


    counter2=0;
    counter3=0;

    for i = 1:60
        try
            if  i == p(i-counter3)
                bags_apple_train{i-counter3}(:,:) = RBG_segmented_apples{p(i-counter3)}(:,:);
                bags_banana_train{i-counter3}(:,:) = RBG_segmented_banana{p(i-counter3)}(:,:);
                counter2 = counter2+1;

            elseif i == notp(i-counter2)
                bags_apple_test{i-counter2}(:,:) = RBG_segmented_apples{notp(i-counter2)}(:,:);
                bags_banana_test{i-counter2}(:,:) = RBG_segmented_banana{notp(i-counter2)}(:,:);
                counter3 = counter3 +1;
            end
        end
    end


end
end
```

5 References

[1] : MIS-Boost: Multiple Instance Selection Boosting,Emre Akbas and Bernard Ghanem and Narendra Ahuja, CoRR,2011,
[2] : "MILES: Multiple-instance learning via embedded instance selection." by Chen, Yixin, Jinbo Bi, and James Ze Wang, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2006): 1931-1947