Machine Learning Assignment 5

Que Shuhao(4739124)

May 31, 2018

a

First, we create a function $y = e^{-x} + x - 1$, which is basically subtracting the left part by the right part.
First-order derivative of the function will be:
$y' = -e^{-x} + 1$
Second-order derivative of the function will be:
$y'' = e^{-x}$

Now we set y' to 0 and get x = 0
Since $y''(x = 0) = 1 > 0$
So when x = 0, the value of the function y will be the minimum: $y(x = 0) = 0$
So we have the conclusion $y \geq 0$
So, $e^{-x} + x - 1 \geq 0$
So, $e^{-x} \geq 1 - x$

b

```
function [f,theta, y] = weak_learner(trainSet, label)

[x,fea_num] = size(trainSet);
label_name = unique(label);


for i = 1:fea_num
    f = i;
    count = 1;
    max_theta = max(trainSet(:,i));
    min_theta = min(trainSet(:,i));
    distance = []; %initialize the distance as zero
    param = [];
   %iterate through all available thetas in feature i
    for theta = min_theta:1:max_theta
        result1 = zeros(x,1);
        result2 = zeros(x,1);
        for j = 1:x   %x is the number of samples
            if trainSet(j,i) <= theta
```

```
                %result1 and result2 are used to distinguish different results using different signs '<' and '>'
                %different signs are represented by the value of y. When y
                %is 0, the sign is '<', when y is 1, the sign is '>'
                result1(j,1) = label_name(1); %label w1
                result2(j,1) = label_name(2); %label w2
            else
                result1(j,1) = label_name(2);
                result2(j,1) = label_name(1);
            end
        end
        if pdist2(result1', label') >= pdist2(result2', label')
            y = 1; %'>'
            result = pdist2(result2',label');
        else
            y = 0; %'<'
            result = pdist2(result1',label');
        end
        distance(1,count) = result;
        param(:,count) = [f; theta; y];
        count = count + 1;
    end
    [min_val, ind] = min(distance);
    min_distance(i) = min_val; %storing the minimum error of feature i
    param_op(:,i) = param(:,ind); %storing the corresponding parameters f, theta and y
end

[min_error, index] = min(min_distance);
f = param_op(1,index);
theta = param_op(2,index);
y = param_op(3,index);
```

C

The generated data is shown in Figure 1. The total number of samples are 100 and it has two labels 1 and 2. There are only 2 feature columns.

The optimal parameters obtained by my decision dump for this dataset is:
feature = 1
threshold = 0.7094
y = ′ <′

The decision stump does not change output results if the second feature is scaled because the second feature is never chosen as the best feature to distinguish between two labels. While if the first feature is scaled, since the first of generated data using gendats.png

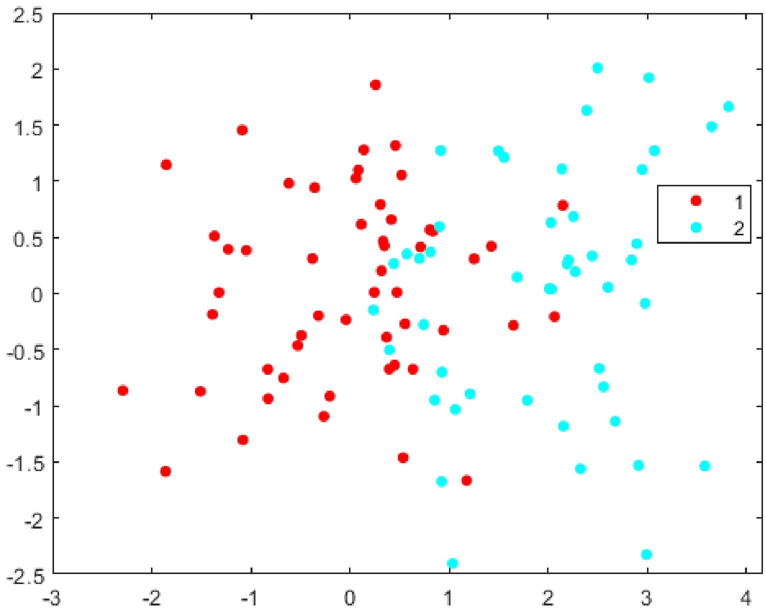

Figure 1: Scatterplot of generated data using gendats

feature is always chosen as the best feature to distinguish between two labels the output optimal threshold will also be scaled consequently.

d

After implementing the weak_learner function on dataset optdigitsubset, the following result is obtained:

condition1: Use the first 50 objects for each class for training, and the rest for testing.

classification error on the test objects: 1.76%

condition2: Use random subsets of 50 for training, and the rest for testing. Iteration time is 10 in the experiment, as illustrate in Figure 2

the mean of the error: 2.22%

standard deviation of the error: 0.0092

As illustrated by the numbers, the performance with training on the first 50 object is unexpectedly good considering the error rate based on condition1 is smaller than average error rate based on condition2.

e

The dataset used for testing the implementation of weak learner function that can receive weights as input is shown in Figure 3. The number of objects in dataset is 100 in total. The number of feature columns are 2.

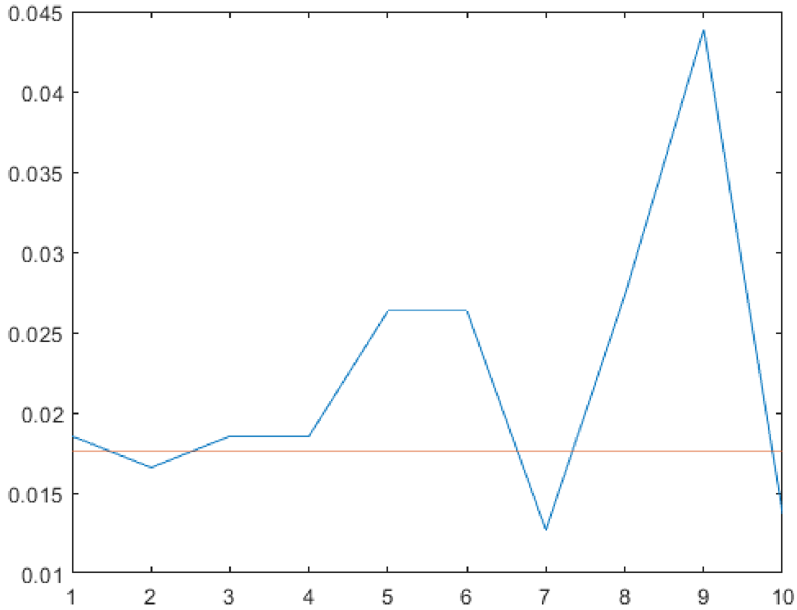

Figure 2: Classification errors on the test objects. Red line denotes the error based on condition1 and blue line denotes the errors based on condition2

Weight assigning scheme 1: the first object is assigned with the largest weight 0.9902 while the rest are assigned with the equal weight as 9.9e-05.

Weight assigning scheme 2: all the objects are assigned with the equal weight as 0.01.

The output result of scheme 1 is:
feature = 1
threshold = 2.4474
y $= ' <'$
classification error on training set: 31%

The output result of scheme 2 is:
feature = 1
threshold = 1.3906
y $= ' <'$
classification error on training set: 10%

As illustrated by both the output results and the scatterplot of the generated dataset, it can be seen that since the largest weight is assigned to the first object (the value of its feature 1 is 2.4409), it has larger influence on deciding the optimal threshold value (2.4474 in the experiment), which consequently affects the prediction accuracy on training dataset. Since the first object is assigned with the largest weight, the learner will adjust the parameters to make sure that this very object will be labeled correctly in order to minimize the total error.

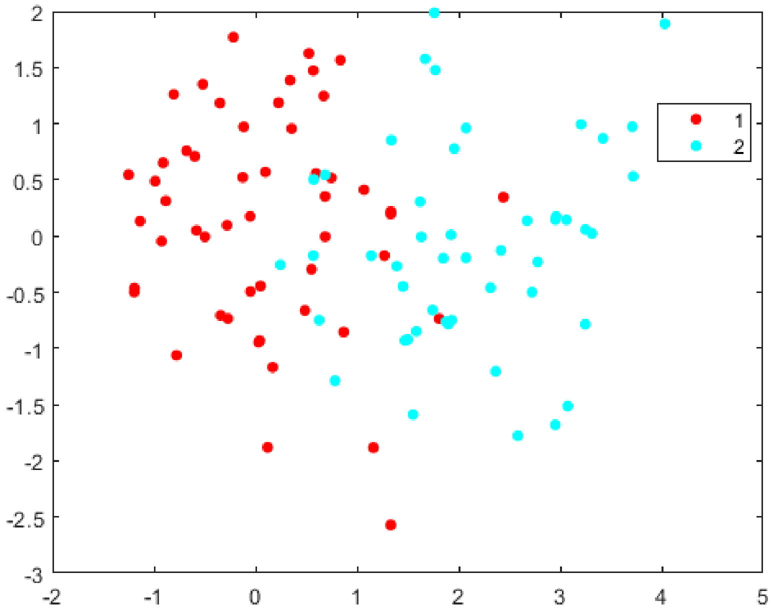

Figure 3: Generated Dataset

f

a class: adaboost_new

```
classdef adaboost_new
    properties
    f(1,:)
    theta(1,:)
    y(1,:)
    beta_vec(1,:)
    iter_num
    weights(1,:)
    label_name(1,:)
    end
        methods
            function obj = adaboost_new(TrainSet, train_label, T)
                if nargin == 3
                    [num_labeled_example, num_feature] = size(TrainSet);
                    count = 1;
                    %Intialize the weight vector
                    weight_vector = (1/num_labeled_example)*ones(num_labeled_example,1);
```

```
            while count <= T
                %normalize the weight vector
                p = weight_vector/sum(weight_vector);
                %provide the weak_learner with distribution p
                [f,theta, y] = weak_learner_weighted(TrainSet, train_label, p);
                %store the outputs of weak_learner
                f_vec(count) = f;
                theta_vec(count) = theta;
                y_vec(count) = y;
                % Calculating error
                test_result = hypo(f, theta, y, TrainSet, train_label);
                error = sum(p.*abs(test_result - train_label));
                beta(count) = error / (1 - error);
                %update the weight vector
                for i = 1:length(weight_vector)
                    weight_vector(i) = weight_vector(i)*(beta(count)^(1-abs(test_result(i)-train_label(i))));
                end
                count = count + 1;
            end
          obj.f = f_vec;
          obj.theta = theta_vec;
          obj.y = y_vec;
          obj.beta_vec = beta;
          obj.iter_num = T;
          obj.weights = weight_vector;
          obj.label_name = unique(train_label);
          end
    end
    function test_result_boost = classify(obj, test_data)
    %output the hypothesis
            for j = 1:max(size(test_data))
                sum_left = 0;
                sum_right = 0;
                for i = 1:obj.iter_num
                    test_result = hypo(obj.f(i), obj.theta(i), obj.y(i), test_data(j,:), obj.label_name);
```

```
                            sum_left = sum_left + log(1/obj.beta_vec(i))*test_result;
                            sum_right = sum_right + 0.5*log(1/obj.beta_vec(i));
                        end
                        if sum_left >= sum_right
                            test_result_boost(1,j) = 1;
                        else
                            test_result_boost(1,j) = 0;
                        end
                    end
                end
            end
end
```

call_adaboost.m

```
close all
clear all

VecX = dlmread('optdigitsubset.txt');

N1 = 554;
N2 = 571;
sample_num = 50;
VecX1 = zeros(sample_num, 64);
VecX2 = zeros(sample_num, 64);
Vec1 = zeros(1,N1); %label 0
Vec2 = ones(1,N2); %label 1

for i = 1:1:sample_num
    for j = 1:1:64
        VecX1(i,j)=VecX(i,j);
        VecX2(i,j)=VecX(i+N1,j);
    end
end

%training data labeled
label = [Vec1(1:50), Vec2(1:50)];
label = label';
TrainSet = [VecX1;VecX2];
%test dataset
test_data_1 = VecX(sample_num+1:N1, :);
test_data_2 = VecX(N1+sample_num+1:N1+N2,:);
test_label = [Vec1(sample_num+1:N1),Vec2(sample_num+1:N2)];
test_data = [test_data_1;test_data_2];


```

```
iter_num = 17;

for T = 1:iter_num
    ada = adaboost_new(TrainSet, label, T); %fit adaboost model with training dataset
    test_result_boost1 = classify(ada, test_data); % predict test dataset and output labels
    error_boost1 = sum(abs(test_result_boost1 - test_label));
    error_rate_boost1(T) = error_boost1/length(test_label);

    test_result_boost2 = classify(ada, TrainSet);
    error_boost2 = sum(abs(test_result_boost2 - label'));
    error_rate_boost2(T) = error_boost2/length(label);
end

figure;
plot([1:T], error_rate_boost1);
hold on
plot([1:T], error_rate_boost2);
xlabel('iteration number') % x-axis label
ylabel('error rate') % y-axis label
legend('Test Error','Train Error')
```

g

The generated banana dataset is shown in Figure 4. The generated simple dataset (gendats) is shown in Figure 5.

The iteration number is set to be a fixed number: 100

The weights assigned to test objects are shown in Figure 6 and 7. For Banana Dataset, which corresponds to Figure 6, the test object 13 is assigned with the largest weight and for test objects 8, 47 and 56, they are also assigned with large weights. As for the simple dataset generated by using gendats, which corresponds to Figure 7, the test object 49 is assigned with the largest weight, and for test objects 13 and 60, they are also assigned with large weights.

The more difficult the objects are to be classified correctly, the higher weights will be assigned to them. Therefore, these objects are those who are more difficult to be classified correctly.

h

As illustrated in Figure 8, the error on training dataset will drop to 0 when iteration number reaches 3. As for the errors on test objects, the error rate shows unstable changes as the iteration number T increases. When iteration number T = 17, the error rate reaches its lowest value, which is 0.0078.

If I take the classifier with the optimal T = 17, the training object 66 out of 100 training objects gets the highest weight. And as illustrated in Figure 9, the banana data.png

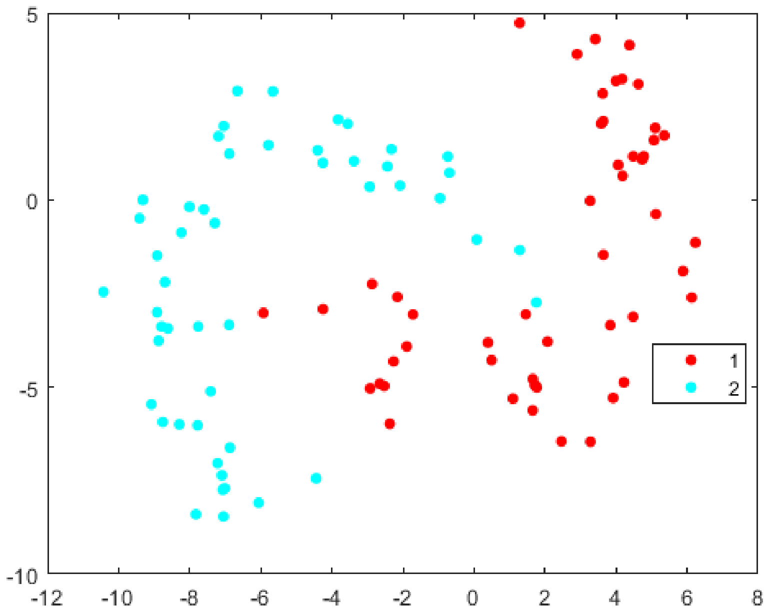

Figure 4: Banana Dataset

objects 17 and 29 (class 0) and the objects 86 (class 1) are also assigned with relatively higher weights. These four objects are shown in Figure 10.

The more difficult the objects are to be classified correctly, the higher weights will be assigned to them. Therefore, these objects are those who are more difficult to be classified correctly.

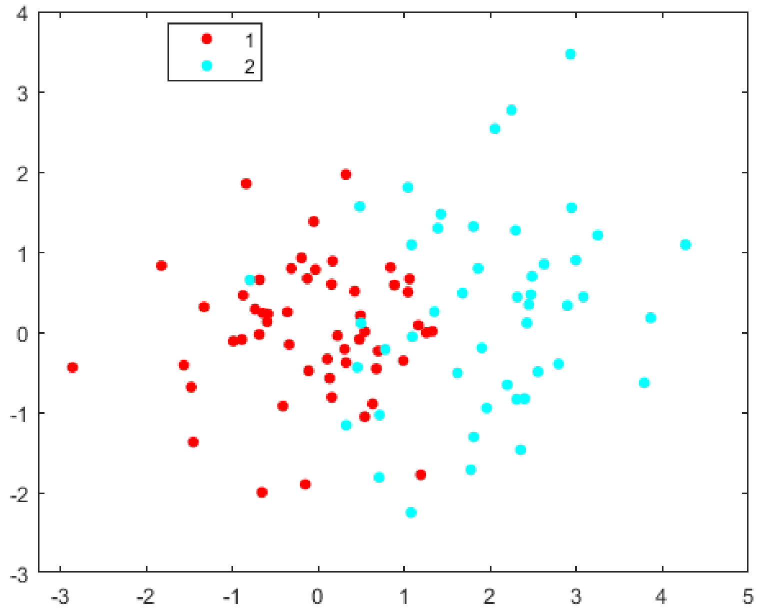

Figure 5: Gendats Dataset

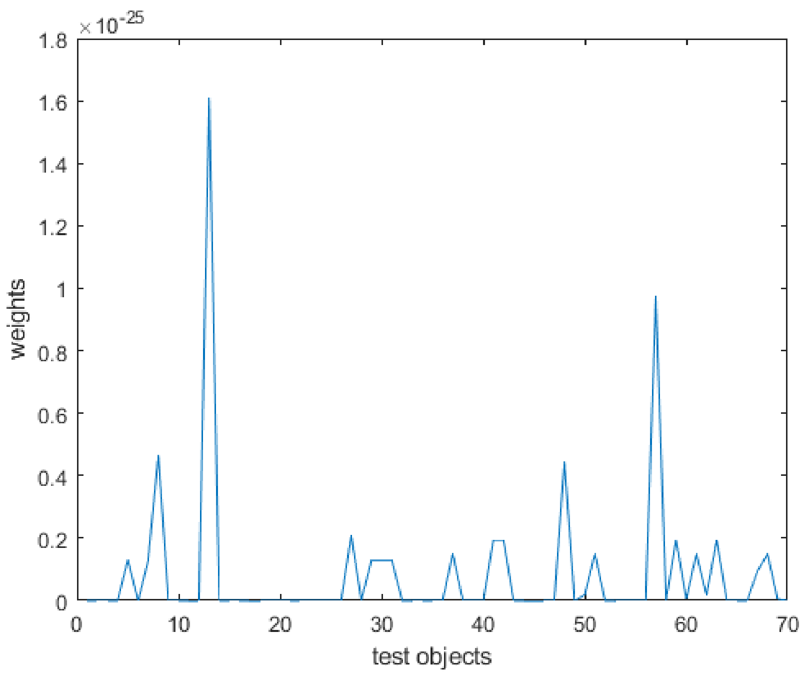

Figure 6: Weights assigned to test objects

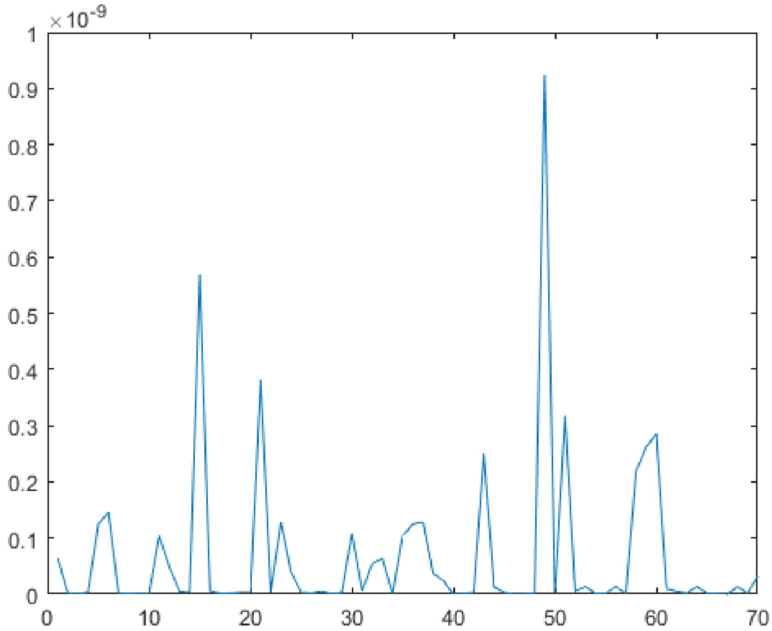

Figure 7: Weights assigned to test objects

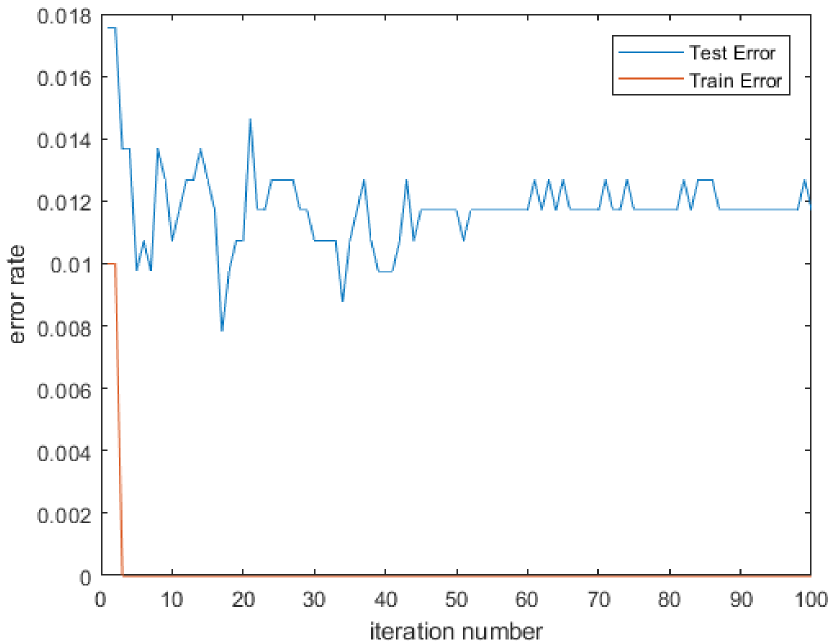

Figure 8: Classification errors on test and training objects

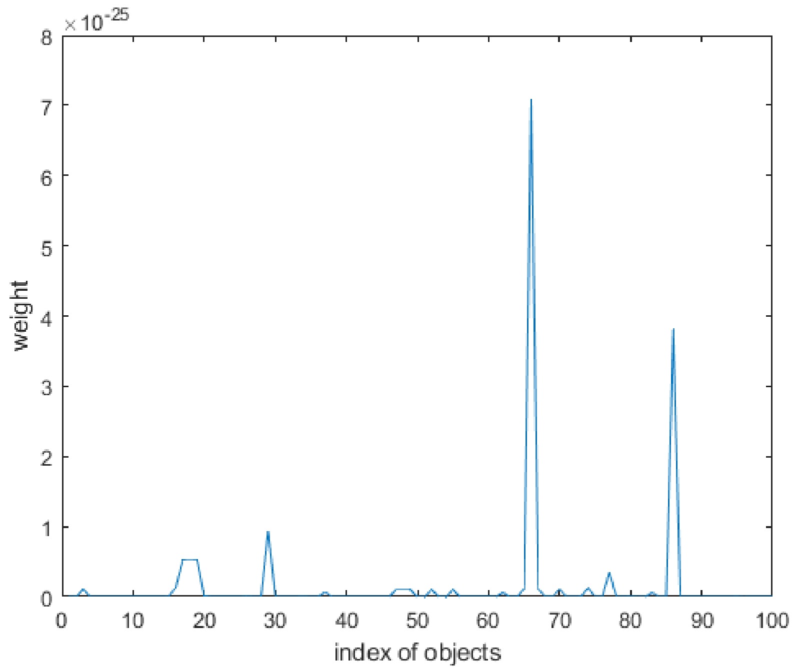

Figure 9: Weights on training objects

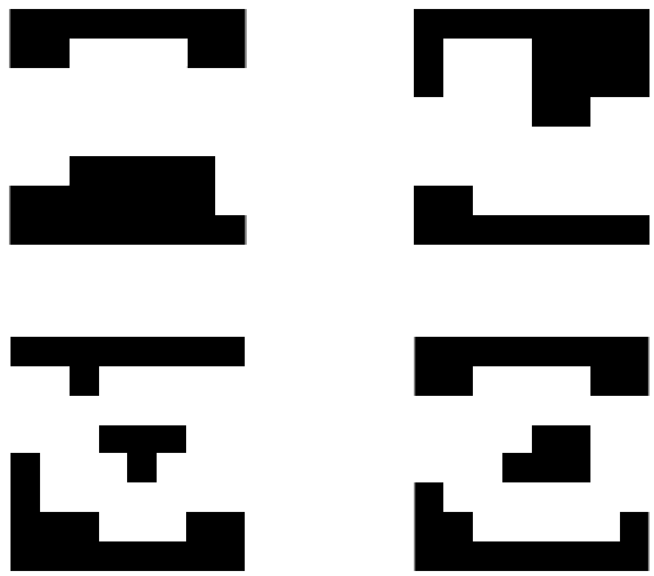

Figure 10: Objects assigned with high weights