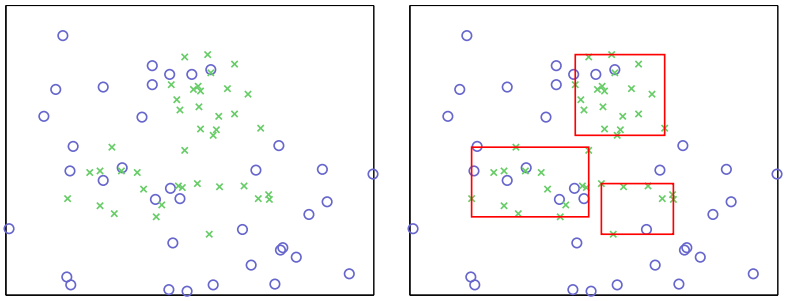
# Unsupervised Clustering with Axis-Aligned Rectangular Regions

Sung Hee Park*
Stanford University

Jae-Young Kim†
Stanford University



(a) Input data (b) Clustering with three rectangular regions

**Figure 1:** *An example of input and output of our clustering method*

## Abstract

This project presents a new method for binary clustering that classifies input data into two regions separated by axis-aligned rectangular boundaries. Given the number of rectangular regions to use, this algorithm automatically finds the best boundaries that are determined concurrently. The formulation of the optimization problem involves minimizing the sum of minimum functions. To solve this problem, we introduce underestimate of the minimum function with piecewise linear and convex envelope, which results in MILP (Mixed Integer and Linear Programming). We show several results of our algorithm and compare the effects of each term in our objective function. Finally, we demonstrate that our method can be used in image capturing application to determine the optimal scheme that minimize the total readout time of pixel data.

**Keywords:** machine learning, unsupervised clustering, axis-aligned rectangular regions, mixed integer and linear programming

## 1 Introduction

The main goal of clustering problem is to figure out an efficient algorithm to classify random data into several groups so that it can be easily handled in more complex processing stage afterwards. Each clustering algorithm is designed to handle data in a systematic way, based on the assumptions it makes on its input data. In this project, we would like to propose an unsupervised clustering method that classifies binary labeled input data into two separate regions. The main difference from other existing clustering methods is that we choose multiple rectangular regions to separate data. Each rectangular region is axis-aligned which means that its sides are aligned to the axes of orthogonal coordinate system and it is not allowed to rotate to arbitrary angles.

To make the clustering algorithm applicable to real world applications, we propose four desirable characteristics that our clustering algorithm should have.

**Unsupervised clustering.** It is desired that clustering algorithm should work by itself. Users can not always guide clustering if the size of input data is large or clustering should be done many times iteratively.

**Optimality.** Rather than using heuristic methods, solving the optimization problem guarantees to have the best results. Also, it can be served as a standard reference case to evaluate other variant clustering methods.

**Robustness to outliers.** The real data tend to be noisy and include many outliers. The algorithm should be robust enough to handle outliers. Also, it is very common to get non-separable input data and error terms should be handled properly.

**Adjustable objective function.** The method should be easily modified so that it can fit into various applications with different constraints. In addition, having explicit trade-off parameters will help users to tweak the algorithm to work well on the problem.

In the following sections, we will explain details about our algorithm and demonstrate it satisfies the properties stated above. Finally, we will talk about the application for image capturing that is utilized by our clustering method.

## 2 Clustering as an Optimization Problem

Input data is given with binary labels on the 2-dimensional plane. We want to formulate an optimization problem to classify binary labeled data with axis-aligned rectangles efficiently. Figure 1 shows one example when three rectangles are used to classify the given input data. In general, input data may not be separable with rectangular classification as in the Figure 1. Therefore, our goal is to maximize the number of interesting points classified in the rectangles, while keep the number of unwanted points as small as possible in them. In the following sections, we will define several penalty functions for misclassification and formulate an optimization problem for the case with one rectangular boundary. Then we will extend our framework to handle multiple rectangles at the same time.

### 2.1 Classification Error for One Rectangular Region

We would like to define two types of classification error. Type A error quantifies the amount of violation when unwanted points are located inside the rectangle. Similarly, type B error is incurred when

*e-mail: shpark7@stanford.edu
†e-mail: jykim1@stanford.edu

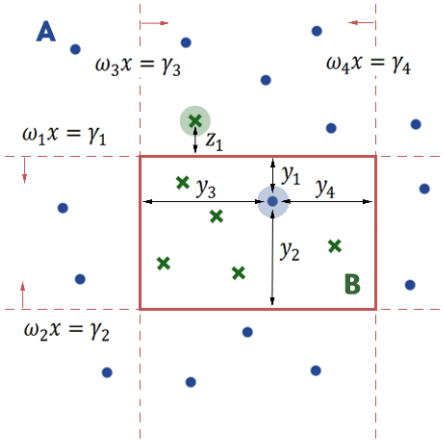



**Figure 2:** *Classification error for one rectangular region.*

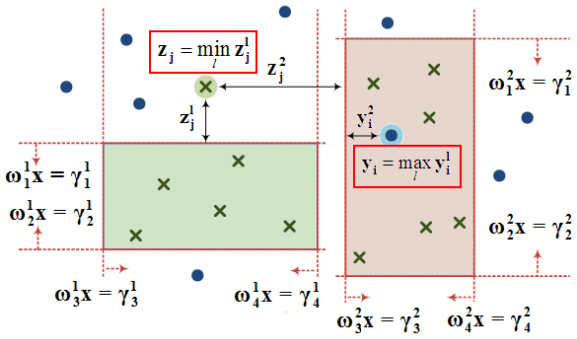



**Figure 3:** *Classification error for multiple rectangular regions.*

our target points are not included in the rectangle. In Figure 2, $y_i$ is type A error of $i^{th}$ data point with label A and $z_j$ represents type B error of $j^{th}$ point with label B. If we define a rectangle as a hyperplane set $\{\omega_k x = \gamma_k, k = 1, \ldots, 4\}$, type A and B error can be formulated as follows:

$$
\begin{aligned}
&y_{ik} = \max\{\omega_k A_i - \gamma_k, 0\} \\
&z_{jk} = \max\{-\omega_k B_j + \gamma_k, 0\} \\
&i = 1, \ldots, m_a,\ j = 1, \ldots, m_b,\ k = 1, \ldots, K \\
&\text{For } K = 4,\ \omega = \begin{pmatrix} 0 & 0 & -1 & 1 \\ -1 & 1 & 0 & 0 \end{pmatrix}
\end{aligned}
$$

where $A_i$ is $i^{th}$ input data with label A, $B_j$ is $j^{th}$ input data with label B, $m_a$ is the number of input data with label A and $m_b$ is the number of input data with label B.

$y_{ik}$ is an error incurred by $i^{th}$ data from $k^{th}$ side of the rectangle and $z_{jk}$ is an error by $j^{th}$ data from $k^{th}$ side of the rectangle. When a type A point is in the rectangle, then all the $y_{ik}$ are positive. Thus, we define type A error of $i^{th}$ points as $y_i = \min_k y_{ik}$. On the contrary, type B error happens when type B points have at least one positive value of $z_{ik}$. Type B error can be set as $z_j = \sum_{k=1}^{4} z_{jk}$. Then our goal is to minimize the sum of two error terms over all input points. However, we have the sum of minimum functions involved in our objective, which are concave. This term prevents us to directly apply convex optimization techniques. Thus we use some trick to formulate the problem as a MILP which will be described in Section 2.3.

## 2.2 Classification Error for Multiple Rectangular Regions

Let's define a variable $L$ as the number of rectangles. Then every terms we defined at the previous section can be used with index $l$ to represent they are the terms regarding $l^{th}$ rectangular region. Thus we can write $y_{ik}^l$ and $z_{jk}^l$ as

$$
\begin{aligned}
&y_{ik}^l = \max\{\omega_k^l A_i - \gamma_k^l, 0\} \\
&z_{jk}^l = \max\{-\omega_k^l B_i + \gamma_k^l, 0\} \\
&y_i^l = \min_k y_{ik} \\
&z_j^l = \sum_{k=1}^{4} z_{jk}
\end{aligned}
$$

If we consider all $L$ rectangles, then we have non-zero type A error when $A_i$ point is enclosed by any rectangular boundary. So, we can represent type A error as $y_i = \max_l y_i^l$ as shown in Figure 3. Likewise, type B points are located outside of all rectangles to have non-zero misclassification. Therefore, we can formulate it as $z_j = \min_l z_j^l$. Figure 3 shows type A and B errors when we use two rectangles. As a result, for the case with $L$ rectangular boundaries, our optimization problem becomes:

$$
\begin{aligned}
\min_{\gamma,y,z} \quad & \sum_{i=1}^{m_a} \max_l \min_k \{y_{ik}^l\} + \sum_{j=1}^{m_b} \min_l \Big\{ \sum_{k=1}^{4} z_{jk}^l \Big\} \\
\text{s.t.} \quad & y_{ik}^l \geq \omega_k^l A_i - \gamma_k^l \quad i = 1, \ldots, m_a, \forall l, \forall k \\
& z_{jk}^l \geq -\omega_k^l B_j + \gamma_k^l \quad j = 1, \ldots, m_b, \forall l, \forall k \\
& y_{ik}^l \geq 0 \quad i = 1, \ldots, m_a, \forall l, \forall k \\
& z_{jk}^l \geq 0 \quad j = 1, \ldots, m_b, \forall l, \forall k \\
\text{where} \quad & K = 4,\ \omega^l = \begin{pmatrix} 0 & 0 & -1 & 1 \\ -1 & 1 & 0 & 0 \end{pmatrix}
\end{aligned}
$$

## 2.3 MILP Formulation using Integer Variables

As noted at the beginning of this section, our objective function is not a convex function. To reformulate the problem as a convex form, we introduce new variables. Let's consider type B error term first.

$$
\begin{aligned}
&v_j^l = z_j^l t_j^l \quad \text{where } t_j^l \text{ is 0-1 integer} \\
&\sum_{l=1}^{L} t_j^l = 1
\end{aligned}
$$

$t_j^l$ are integer variables that can have values 0 and 1 only. Also, since the sum of $L$ variables is one, only one $t_j^l$ will have the value of 1 while other $L-1$ terms are zero. Here, we want to make $t_j^{l'} = 1$ where $l' = \text{argmin}_l z_j^l$ so that we have

$$
\sum_{l=1}^{L} v_j^l = \min_l z_j^l = \min_l \Big\{ \sum_{k=1}^{4} z_{jk}^l \Big\}.
$$

To enforce the relation, let's introduce underestimate of minimum function by piecewise linear and convex envelope as done in [Ryoo 2006; Ryoo and Sahinidis 2001].

$$v_j^l \geq \max\{z_j^l + Mt_j^l - M, 0\} \quad j = 1, \ldots, m_b, \forall l$$

where $M$ is an arbitrary positive number that is greater than the maximum value of $z_j^l$. We can see that $v_j^l$ will have non-zero value only when $t_j^l = 1$ and the optimization process will minimize $v_j^l$ to have the minimum of $z_j^l$ for each $j$. As a result, the terms relevant to $z_{jk}^l$ in the objective function can be formulated into:

$$\sum_{j=1}^{m_b} \min_l \Big\{ \sum_{k=1}^{4} z_{jk}^l \Big\} = \sum_{j}^{m_b} \sum_{l}^{L} v_j^l$$

Now objective function becomes linear. We can apply the same linearization trick to type A error $y_{ik}^l$. Let's define the following variable $u_i$.

$$u_i^l = \min_k \{y_{ik}^l\}$$
$$u_i = \max_l u_i^l = \max_l \min_k \{y_{ik}^l\}$$

Then, we can redefine $u_i$ using integer variables and the underestimate as follows:

$$u_{ik}^l = y_{ik}^l s_{ik}^l \quad \text{where } s_{ik}^l \text{ is 0-1 integer}$$
$$u_{ik}^l \geq \max\{y_{ik}^l + Ms_{ik}^l - M, 0\}$$
$$u_i \geq u_i^l = \sum_{k=1}^{4} u_{ik}^l \quad i = 1, \ldots, m_a, \forall l$$
$$\sum_k s_{ik}^l = 1 \quad i = 1, \ldots, m_a, \forall l$$

Consequently, our objective function for $y_{ik}^l$ becomes:

$$\sum_{i=1}^{m_a} \max_l \min_k \{y_{ik}^l\} = \sum_{i=1}^{m_a} u_i$$

Combining all together, our final convex optimization problem using integer variables becomes:

$$\begin{aligned}
\min_{u,v,y,z,t,s} \quad & C_1 \sum_{i=1}^{m_a} u_i + C_2 \sum_{j=1}^{m_b} \sum_{l=1}^{L} v_j^l \\
s.t. \quad & v_j^l \geq z_j^l + Mt_j^l - M \quad j = 1, \ldots, m_b, \forall l \\
& u_{ik}^l \geq y_{ik}^l + Ms_{ik}^l - M \quad i = 1, \ldots, m_a, \forall l, \forall k \\
& u_i \geq \sum_{k=1}^{4} u_{ik}^l \quad i = 1, \ldots, m_a, \forall l \\
& \sum_{l=1}^{L} t_j^l = 1 \quad j = 1, \ldots, m_b, t_j^l \in \{0, 1\} \\
& \sum_{k=1}^{4} s_{ik}^l = 1 \quad i = 1, \ldots, m_a, s_{ik}^l \in \{0, 1\}, \forall l \\
& y_{ik}^l \geq \omega_k^l A_i - \gamma_k^l \quad i = 1, \ldots, m_a, \forall l, \forall k \\
& z_{jk}^l \geq -\omega_k^l B_j + \gamma_k^l \quad j = 1, \ldots, m_b, \forall l, \forall k \\
& y_{ik}^l \geq 0 \quad i = 1, \ldots, m_a, \forall l, \forall k \\
& z_{jk}^l \geq 0 \quad j = 1, \ldots, m_b, \forall l, \forall k \\
& v_j^l \geq 0 \quad j = 1, \ldots, m_b, \forall l \\
& u_{ik}^l \geq 0 \quad i = 1, \ldots, m_a, \forall l, \forall k
\end{aligned}$$

where $C_1$ and $C_2$ are weighting parameters. If we increase $C_2$ weight by fixing $C_1$, then we consider type B errors are more important and optimization will try more to reduce type B classification error. On the contrary, if $C_1$ is increasing while $C_2$ is fixed, then optimization will focus more on trying to exclude $B_j$ points.

### 2.4 Applying Additional Geometry Constraints

Since our problem formulation involves only linear constraints, it is straightforward to add additional geometry constraints to determine the shape of rectangular regions. First, we can add quadratic terms in the objective function to control the dimensions of the rectangles.

$$\begin{aligned}
\min_{u,v,y,z,t,s} \quad & C_1 \sum_{i=1}^{m_a} u_i + C_2 \sum_{j=1}^{m_b} \sum_{l=1}^{L} v_j^l + C_3 \sum_{l=1}^{L} \{(w_l)^2 + (h_l)^2\} \\
& h_l = \gamma_1^l + \gamma_2^l \\
& w_l = \gamma_3^l + \gamma_4^l
\end{aligned}$$

$h_l$ and $w_l$ correspond to the height and width of $l^{th}$ rectangle. Thus, the third term will make tighter rectangular bounds and user can control $C_3$ to change the strength of the effect. This term also has the effect of minimizing the area of the rectangles because it is minimizing the upper bound of the area. This relation come from the following inequality.

$$h_l w_l \leq 0.5(h_l^2 + w_l^2)$$

Moreover, we can specify the aspect ratio of the rectangle or exactly fix the dimension of the rectangles by adding the following constraints.

$$w_l = \alpha h_l$$
$$w_l = \alpha_1, \; h_l = \alpha_2$$

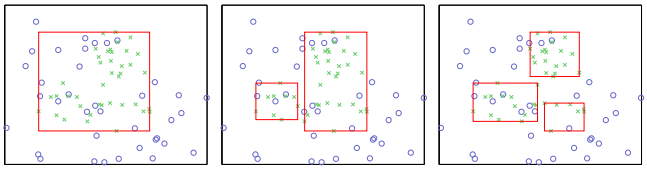
## 3 Clustering Results

### 3.1 Clustering Results

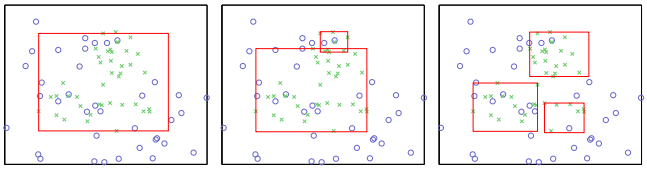
We applied our algorithm to a test data set with 64 type A points and 40 type B points. We used $C_1$=1, $C_2$ = 10, $C_3$ = 0.1 as weighting terms. AMPL/CPLEX is used to solve MILP. Figure 4 shows the visualization of the results we obtained. When no additional constraint is applied, the optimizer finds arbitrary axis-aligned rectangles to minimize objective function. This case gives the tightest rectangles that fit input data as shown in Figure 4(a). If we give fixed aspect ratio constraints, 4:3 in this example, all boundaries has the same shape but different scales. For L=2 case in Figure 4(b), we can see that rectangular regions are less tight than the case for arbitrary rectangles, but it still tries to find the best ones with same aspect ratio. Figure 4(c) is obtained by locating fixed size rectangles to reduce error terms. Since the area of each rectangle is the same, the third term in objective function are not taken into account. The solver places fixed-size rectangles in the best locations. The result looks quite compelling, which confirms the correctness of our problem formulation.

### 3.2 The Effects of the Weighting Terms

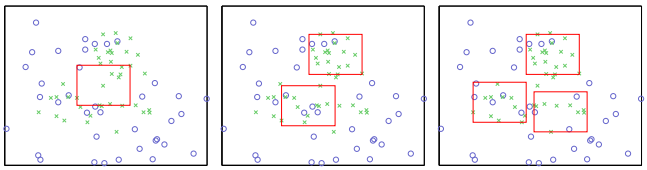
Here, we briefly show how the results will change depending on the weighting terms in the objective function. First, we compare the effect of first two terms of the objective. We set $C_1$=1 and use $C_2$=0.1, 1, 10. The red rectangle in Figure 5(a) corresponds to $C_2$=10 case which has the largest area among three. It tries to minimize misclassification of type B data and the rectangular region



(a) Clustering with arbitrary size rectangles

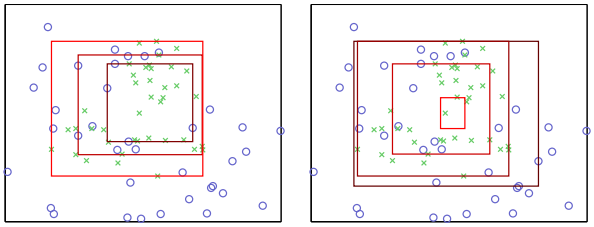


(b) Clustering with rectangles with 4:3 aspect ratio



(c) Clustering with fixed size rectangles

**Figure 4:** *Clustering results with different constraints for L=1, 2, 3. ($C_1$=1, $C_2$=10, $C_3$=0.1)*



(a) Weights on type A and B misclassification

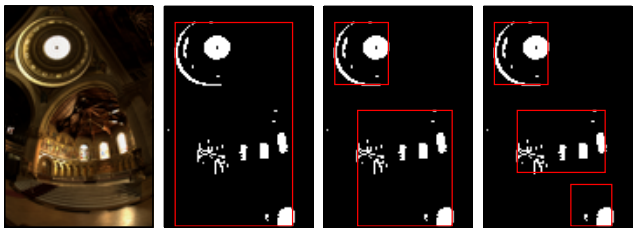
(b) Weights on the dimensions of a rectangle

**Figure 5:** *The effects of changing weight terms in the objective function.*

includes all type B points. On the other hands, the dark color rectangle, when $C_2$=0.1, makes correct decision for all type A points, forcing them to be located outside the rectangle. Figure 5(b) shows the effect of the third term in the objective. Let $C_1$=1, $C_2$=10 and $C_3$=0.1, 1, 10, 100. As $C_3$ gets bigger, the resulting rectangular region becomes smaller. Bigger $C_3$ is represented as darker color in Figure 5(b). This experiment shows that each term in the objective function plays an important role in deciding the best classification result and users can tweak the weighting parameters to obtain the boundaries they want.

## 4 Image Capturing Application

In this section, we will talk about a novel way of capturing image data using our algorithm. The readout speed of imaging system is often bandwidth-limited and it becomes a bottleneck for increasing capturing frame rates. For the limited amount of pixel budget you have at a certain amount of time, it will be better if we can only capture image data that we are more interested in, rather than capturing every pixel from whole scene. Thus, we can adaptively design an efficient image readout scheme by determining which information is more interesting and how to capture that data efficiently. Since



(a) Original (b) L=1 (c) L=2 (d) L=3

**Figure 6:** *Results of clustering applied on real image data*

most image sensors are restricted to readout only rectangular windows from sensor plane, determining how to capture data becomes a problem of deciding rectangular regions that efficiently incorporate important pixel data. Thus, once you determined which regions are more important, the next step can be solved by applying the algorithm discussed in this project.

For instance, let's say you cycle through multiple exposure levels by changing parameter settings every frame to capture high dynamic range video. If over-exposed or under-exposed regions are clumped in small regions, you will want to capture only badly exposed regions again since most parts of image are already well-captured by mid-exposure frames and you don't want to waste your pixel budget to capture them again. In this case, badly exposed pixels are points you want to put inside rectangular regions in next capture, while the rest of the area is supposed to be at outside of rectangles. This case is shown in Figure 6. Here, we want to set the regions so that we can capture most of over-exposed area while minimizing total readout time. We defined interesting pixels as the ones with value higher than the threshold, which means that they are saturated. A mask is generated to use it as an input data set for the algorithm. Figure 6(b) through 6(d) show the results applied on 86 x 128 image for the cases of one, two and three rectangular regions. The total readout time for each case will be determined by actual image sensor characteristics. The total readout time is the sum of pixel readout time and overhead time. Pixel readout time is proportional to the sum of area of all rectangular regions and overhead time grows as the number of rectangles increases. Thus, if the sensor has very small overhead time, then having more number of rectangular regions will result in less total readout time. However, if overhead time is big relative to pixel readout time, using one or two rectangles will be optimal.

## 5 Conclusion

In this project, we have proposed an unsupervised clustering algorithm that classifies binary labeled data into two regions separated by multiple axis-aligned rectangles. By formulating the problem as a mixed integer and linear programming, we are able to determine the boundaries of multiple rectangles at the same time. It gives satisfactory results and easily extended to problems with additional constraints such as enforcing rectangles to have specific aspect ratio or fixed sizes. We also demonstrated that this method can be applied to real image data to come up with an optimal image capturing scheme.

We believe that it will be very useful to extend this method to be applied on larger data sets. Rather than using all input data samples, finding a better representation of data by pre-clustering can help reducing the amount of computation required. Using hierarchical approach with some heuristics may give reasonable results in much less time. Also, our framework will serve as a good reference in evaluating various heuristic approaches.

## References

RYOO, H. S., AND SAHINIDIS, N. V. 2001. Analysis of bounds for multilinear functions. *J. of Global Optimization 19*, 4, 403–424.

RYOO, H. S. 2006. Pattern classification by concurrently determined piecewise linear and convex discriminant functions. *Comput. Ind. Eng. 51*, 1, 79–89.