

**Partial Solution to
Classified Information: The Data Clustering Problem**

Nargess Memarsadeghi and Dianne P. O'Leary

Suppose our original image is of size $m \times p$, with the color for each of the mp pixels recorded by b bits. Then the total storage requirement is mpb bits. In clustering, we will choose k pixel values and map each pixel to one of these.

Problem 1. Compare the number of bits required to store the original image and the clustered image.

Answer: The original image takes mpb bits, while the clustered image takes kb bits to store the cluster centers and $mp[\log_2 k]$ bits to store the cluster indices for all mp pixels. For jpeg images with RGB values ranging between 0 and 255, we need 8 bits for each of the $q = 3$ values (red, green, and blue). Therefore, an RGB image with 250,000 pixels takes $24 * 250,000 = 6,000,000$ bits, while the clustered image takes about $250,000 \log_2 4 = 500,000$ bits if we have 3 or 4 clusters and 250,000 bits if we have 2 clusters. These numbers can be further reduced by compression techniques such as run-length encoding¹.

We can state our clustering problem this way. Given n data points $\mathbf{x}_i \in \mathcal{R}^q$, $i = 1, \dots, n$, we want to find k cluster centers $\mathbf{c}_j \in \mathcal{R}^q$, $j = 1, \dots, k$ and assign each data point to a cluster. We assign \mathbf{x}_i to cluster \mathcal{C}_j if it is closer to that cluster's center than it is to any other center. (Break ties in an arbitrary way.) The distance from data point i to the cluster's center is thus

$$d_i = \min_{j=1, \dots, k} \|\mathbf{x}_i - \mathbf{c}_j\|,$$

and we define the radius of cluster j as

$$r_j = \max_{i: \mathbf{x}_i \in \mathcal{C}_j} d_i.$$

For good clustering, we want each point to be close to the cluster's center. Therefore, we might want to minimize either

$$R = \sum_{j=1}^k r_j^\ell$$

or

$$D = \sum_{i=1}^n d_i^\ell.$$

where $\ell = 1$ or 2 .

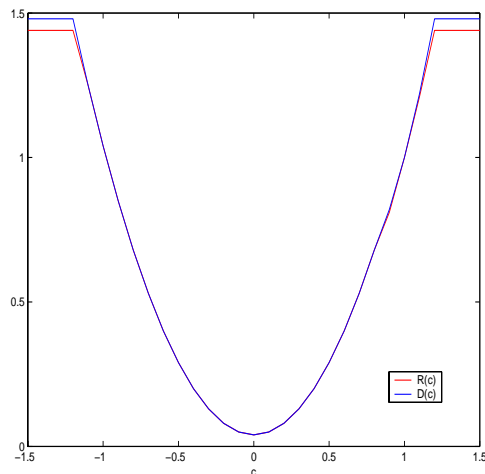


Figure 1: The functions R and D for a particular dataset.

Problem 2. For this problem, use the Euclidean norm with $q = 1$ and $\ell = 2$.

(a) If a function is *convex* and bounded below then any local minimizer is a global minimizer. If not, then an algorithm for minimization might report a local minimizer (a point as good or better than any in its neighborhood) rather than a global one (a point as good or better than any other). Consider the problem with $n = 2$ points and $k = 2$ clusters. Are D and R convex functions?

(b) Are D and R differentiable functions when $n = 2$ and $k = 2$?

(c) Derive a formula for the minimizer of D when $k = 1$ and n is arbitrary.

(d) Suppose we move one of our data points \mathbf{x}_i very far away from the other points, making it an *outlier*. As that point moves further away from the others, what will happen to the cluster centers?

Answer:

(a) Neither D nor R is convex everywhere. Figure 1 plots these functions for a particular choice of points as one of the cluster centers is moved. We fix the data points at 0 and 1 and one of the centers at 1.2, and plot D and R as a function of the second center c . For $c < -1.2$ and $c > 1.2$, the function D is constant, since the second cluster is empty, while for $-1.2 < c < 1.2$, the function is quadratic. Since each function is above some of its secants (the line connecting two points on the graph), each function fails to be convex.

(b) Neither D nor R is differentiable everywhere. Again see Figure 1. The function D fails to be differentiable at $c = -1.2$ and $c = 1.2$. Trouble occurs at the points where a data value moves from one cluster to another.

(c) We want to minimize the function

$$D(\mathbf{c}) = \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{c}\|^2$$

over all choices of \mathbf{c} . Since there is only one center \mathbf{c} , this function is convex and differentiable everywhere, and the solution must be a zero of the gradient.

Differentiating with respect to \mathbf{c} we obtain

$$\sum_{i=1}^n 2(\mathbf{x}_i - \mathbf{c}) = 0$$

so

$$\mathbf{c} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i.$$

It is easy to verify that this is a minimizer, not a maximizer or a stationary point, so the solution is to choose \mathbf{c} to be the centroid of the data points.

(d) As one data point moves from the others, eventually a center will “follow” it, giving it its own cluster. So the clustering algorithm will fit $k - 1$ clusters to the remaining $n - 1$ data points.

Problem 3. Use your favorite optimization function to minimize R with $\ell = 2$ and the Euclidean norm. Use the “Charlie” image provided on the website, and provide a function to evaluate R . Try $k = 3, 4, 5$.

Also minimize D with the same parameters.

- (a) How does the number of variables increase with k ?
 - (b) How does the running time increase with k ?
 - (c) Evaluate the results of the four clusterings and justify the criteria that you choose. As one criterion, discuss how the clustered images look in comparison to the original image.
 - (d) How might a good value of k be determined experimentally?
- Experiment with other choices of points to study the algorithm’s sensitivity to this choice.

Answer: Sample code is given on the website. Note that when using a general purpose optimization code, it is important to match the termination criteria to the scaling of the variables and the function to be minimized; otherwise the code might terminate prematurely (even with negative entries in the cluster centers if, for example, the objective function values are all quite close together) or never terminate (if small changes in the variables lead to large changes in the function). We chose to scale R and D by dividing by 256 and by the number of points in the summation.

The solution is *very* sensitive to the initial guess, since there are many local minimizers.

(a) The number of variables is kq .

(b) Although the number of variables is quite small (9 for $k = 3$ and 15 for $k = 5$), evaluating the function R or D is quite expensive, since it involves a mapping each of the 1,000 pixels to a cluster. Therefore, the overhead of the algorithm is insignificant and the time is proportional to the number of function evaluations. The functions are not differentiable, so modeling as a quadratic function is not so effective. This slows the convergence rate, although only 15-25 iterations are used. This was enough to converge when minimizing D , but not enough for R to converge.

Actually, a major part of the time in the sample implementation is post-processing: the construction of the resulting image!

(c) Figures 2 and 3 show the results with $k = 3, 4, 5$ clusters. The solution is very dependent on the initial guess, but the rather unlikely choice that we made (all zeros in the green coordinate) gave some of the best results.

Our first evaluation criterion should be how the image looks, sometimes called the “eyeball norm”. In the results for minimizing D , it is harder to differentiate the dog from the background. For minimizing R with $k = 3$, his white fur is rendered as green and the image quality is much worse than for minimizing D or using k -means. For $k = 4$ or $k = 5$, though, minimizing R yields a good reconstruction, with good shading in the fur and good rendering of the table legs in the background, and the results look better than those for minimizing D . (Note that the table legs were not part of the sample that determined the cluster centers.)

We can measure the quantitative change in the images, too. Each pixel value x_i in the original or the clustered image is a vector with q dimensions, and we can measure the relative change in the image as

$$\left(\frac{\sum_{i=1}^n \|x_i^{original} - x_i^{clustered}\|^2}{\sum_{i=1}^n \|x_i^{original}\|^2} \right)^{1/2}.$$

This measure is usually smaller when minimizing R rather than D : .363 vs .271 for $k = 3$, .190 vs .212 for $k = 4$, and .161 vs .271 for $k = 5$. The optimization program sometimes stops with negative coordinates for a cluster center or no points in the cluster; for example, for $k = 5$, minimizing D produced only 3 nonempty clusters, and for $k = 2$, minimizing R produced only 2 nonempty clusters.

(d) If $q < 4$, then k might be chosen by looking at the data. For larger values of q , we might try increasing values of k , stopping when the cluster radii fall below the noise level in the data or when the cluster radii stay relatively constant.

Only one choice of data values appears in the sample program, but we can easily modify the program to see how sensitive the solution is to the choice of data.

Problem 4. Implement the k -Means algorithm and run it with the same data and values of k as Problem 3. Compare its performance to that of the algorithm in Problem 3.

Answer: Sample code is given on the website and results are shown in Figures 4. This k -Means code is much faster than that for Problem 3, The best results for $k = 3$ and $k = 5$ are those from k -means, but the $k = 4$ result from minimizing R seems somewhat better than the other $k = 4$ results. The quantitative measures are mixed: the 2-norm of the relative change is .247, .212, and .153 for $k = 3, 4, 5$ respectively, although the algorithm was not run to convergence.

Problem 5. Consider the dataset of $n = 20$ datapoints with $q = 2$, shown in Figure 2:

$$(1, -1 + 2j/9), (-1, -1 + 2j/9),$$

for $j = 0, \dots, 9$. Run the k -means algorithm with $k = 2, 3, 4$. Initialize the centers to the first k points in the list

$$(-1, -1), (1, 1), (-1, 1), (1, -1).$$

Display the clustered data. Discuss the effects of choosing the “wrong” value for k .

Then repeat the experiment, initializing the centers to

$$(0, -1 + 2j/(k-1)),$$

$j = 0, \dots, k-1$. Note that although the answer is different, it is also a local minimizer of the function R . Compare with the first set of answers and discuss the difficulty it illustrates with this kind of clustering.

Answer: The website contains sample code, and Figures 5 and 6 display the results. Each datapoint is displayed with a color and symbol that represent its cluster. An intuitive clustering of this data is to make two vertical clusters, as determined by the algorithm with the first initialization and $k = 2$. Note, however, that the distance between the top and bottom data points in each cluster is the same as the distance between the clusters (measured by the minimum distance between points in different clusters)! The two clusters determined by the second initialization have somewhat greater radii, but are not much worse. What is worse about them, though, is that there is less distance between clusters.

If we choose to make too many clusters ($k > 2$), we add artificial distinctions between data points.

Problem 6. Consider the dataset from Problem 5, but multiply the second component of each vector by 100. Repeat the clustering experiments, applying the same transformation to the initial centers. Discuss why coordinate scaling is important in clustering algorithms.

Answer: The website contains sample code, and Figures 7 and 8 display the results.

Coordinate scaling definitely changes the merits of the resulting clusters. The clusters produced by the second initialization have much smaller radii. Non-linear scalings of the data also affect clustering; for example, the results of clustering the pixels in “Charlie” could be very different if we represented the image in coordinates other than RGB.

References

1. Rafael C. Gonzalez, Richard E. Woods, Digital Image Processing (2nd Edition) Addison-Wesley Pub Co, 2002; Chapter 8.



Figure 2: The images resulting from minimizing R .

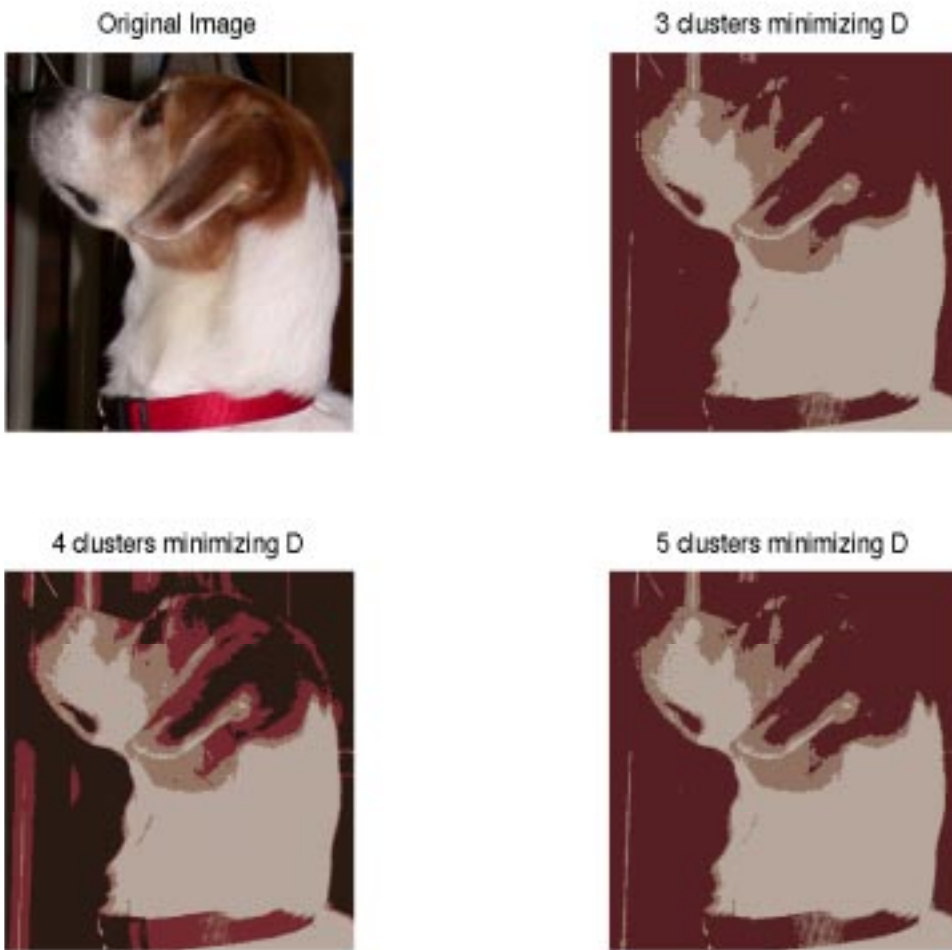


Figure 3: The images resulting from minimizing D .

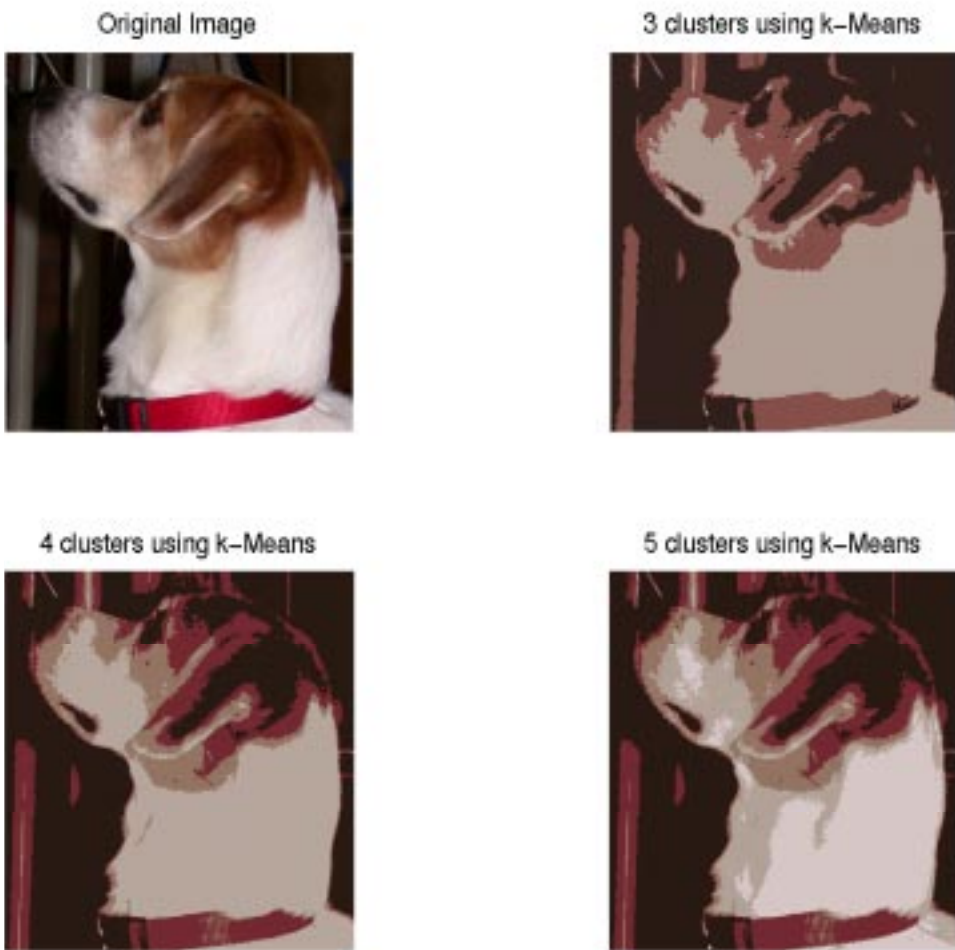


Figure 4: The images resulting from k -means.

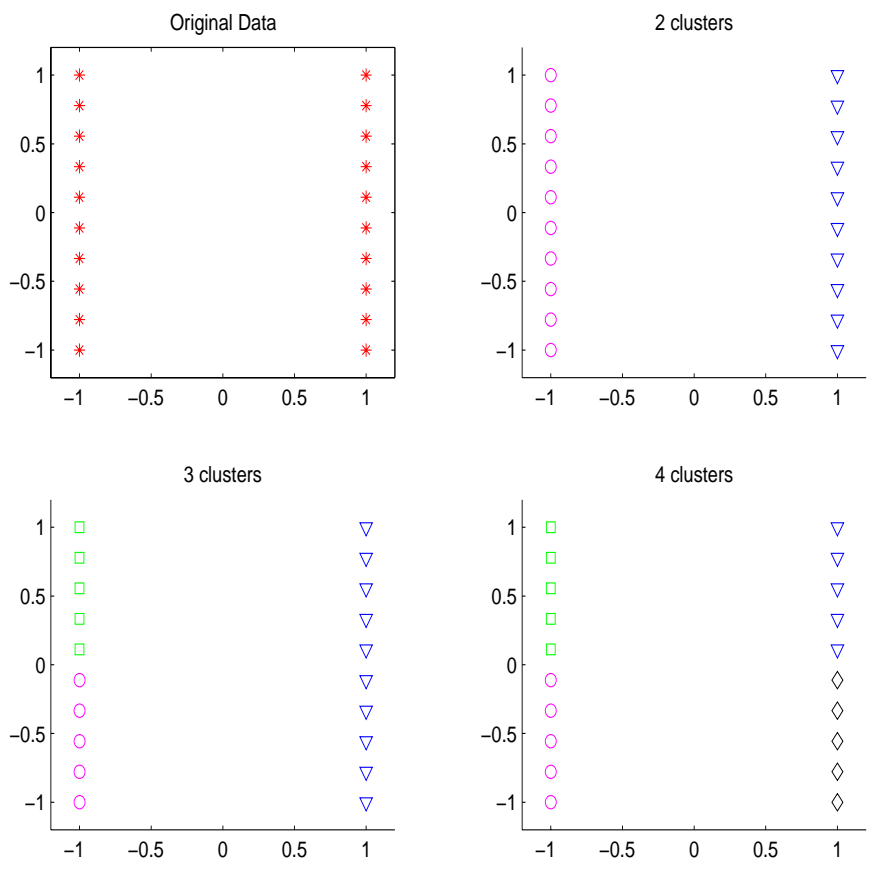


Figure 5: Clustering the data for Problem 5 using the first initialization of centers.

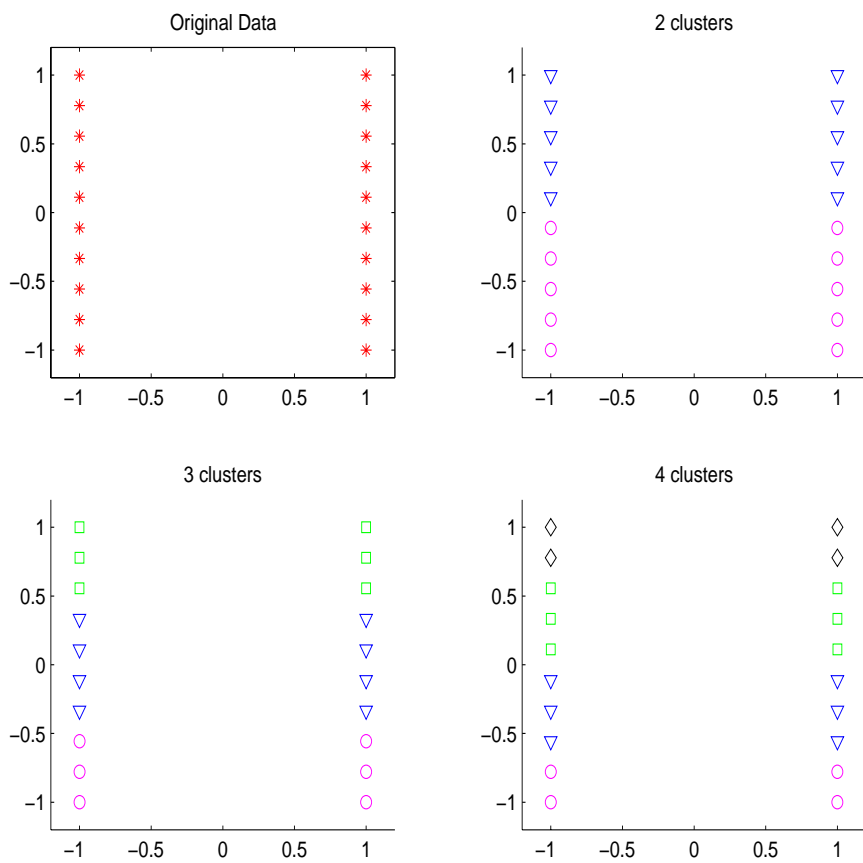


Figure 6: Clustering the data for Problem 5 using the second initialization of centers.

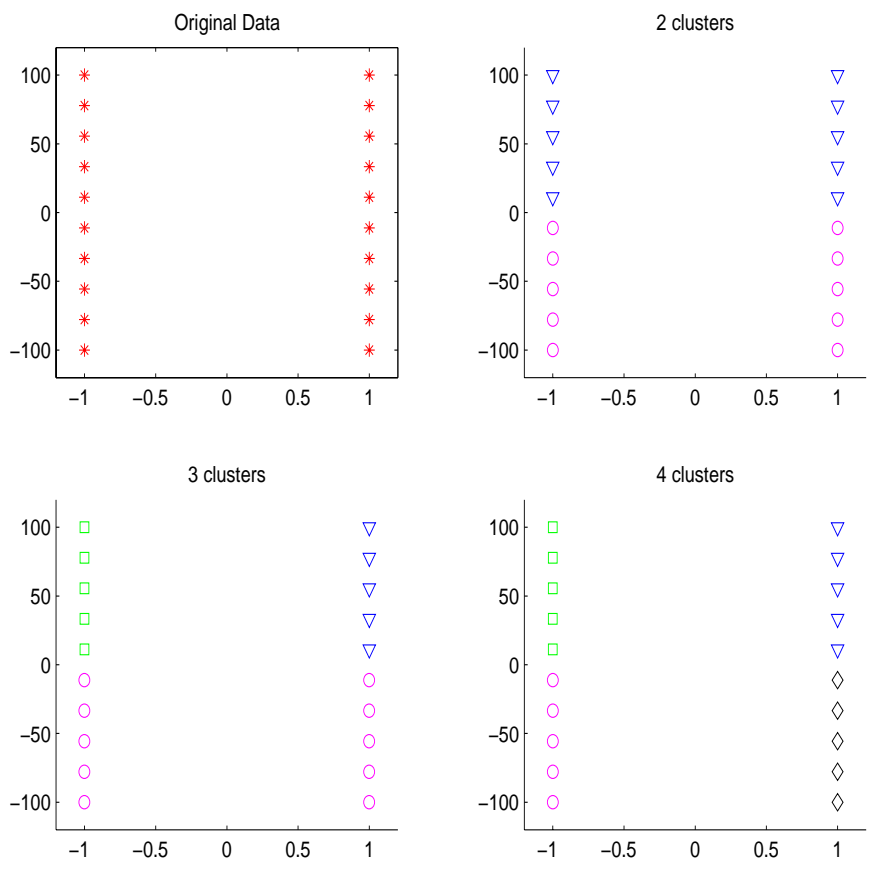


Figure 7: Clustering the data for Problem 6 using the first initialization of centers. Note that the vertical scale is far different from the horizontal.

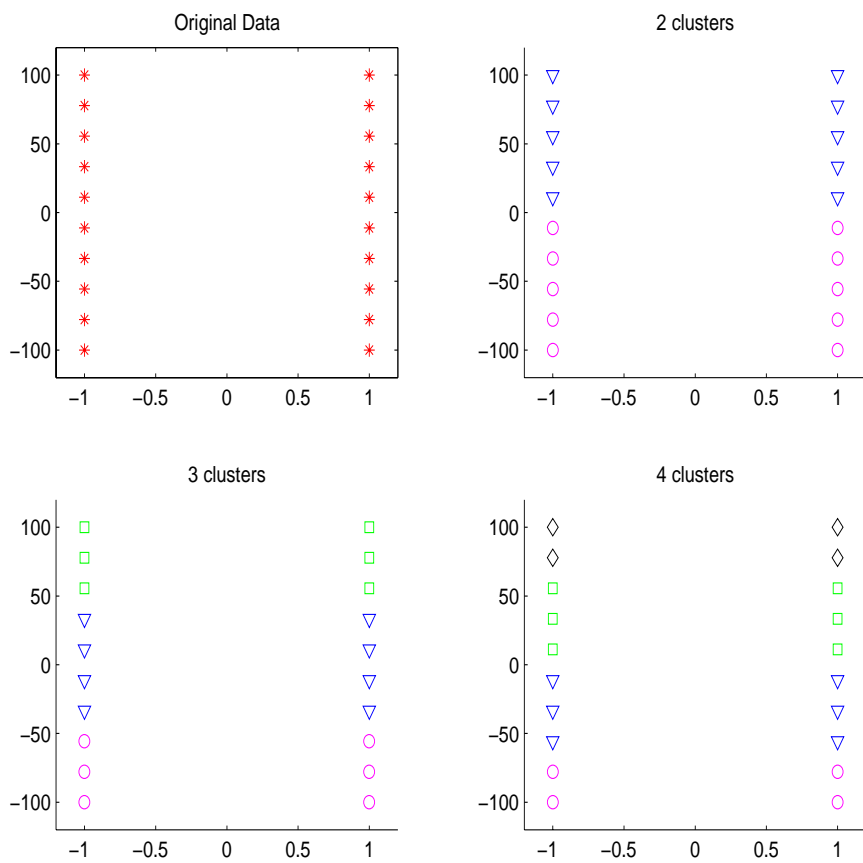


Figure 8: Clustering the data for Problem 6 using the second initialization of centers. Note that the vertical scale is far different from the horizontal.