Machine Intelligence 2 **Exercise Sheet 8**
SoSe 2022, Obermayer/Kashef/Strömsdörfer due: 2022-06-30 23:55

K-means Clustering

Exercise T8.1: From "hard" K-means to Pairwise Clustering to "soft" K-means (tutorial)

(a) How do you train K-means in batch vs. online-fashion?

(b) How can you use a pairwise representation for clustering points?

(c) What is "soft" pairwise clustering?

(d) How do you go from "hard" K-means to "soft" K-means clustering?

(e) How does annealing affect pairwise/K-means clustering?

In this problem set we will implement and apply the batch K-means algorithm, the online version, and the "soft" clustering procedures. The file `cluster.dat` contains a data set of $p = 500$ (2-dimensional) observations generated from four different Gaussians with four different means.

Exercise H8.1: K-means Clustering – batch version (homework, 3 points)

Write a program that implements the *batch* version of K-means clustering and partitions the given data set into M clusters. Repeat the clustering procedure for different initializations of the prototypes and $M = 2, 3, 4, 5, 6, 7, 8$. Include the following steps:

A: Initialization –

1. Set the initial position of each prototype $\underline{\mathbf{w}}_q$ randomly around the mean of the entire dateset.
2. Set the maximum number of iterations: $t_{max} = 5$

B: Optimization –

Implement the K-means update. Each iteration should contain the following two steps

1. Assign all datapoints to their closest prototype.
2. Re-compute the location of the prototypes due to the new assignments.

C: Visualization –

(a) Visualize data points and prototypes for each iteration in a sequence of scatter plots.

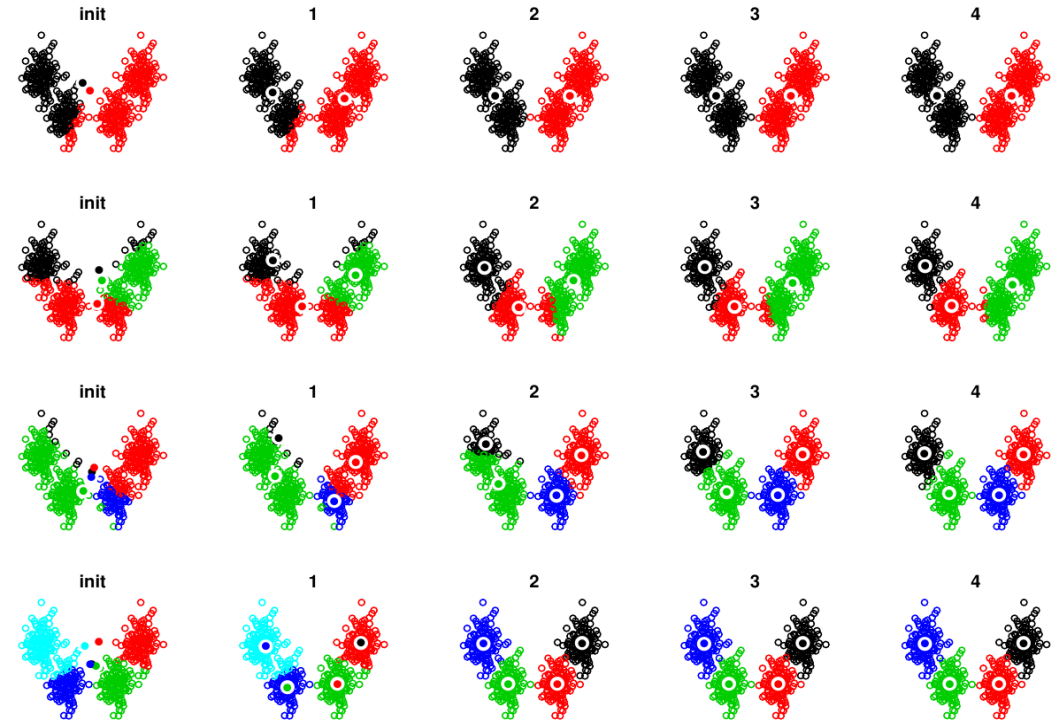
(b) For two different initializations, plot the error function E vs. the iteration t

$$E_{\left[\{m_q^{(\alpha)}\},\{\underline{\mathbf{w}}_q\}\right]} = \frac{1}{p}\sum_{\alpha=1}^{p}\sum_{q=1}^{M} m_q^{(\alpha)} \left\|\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_q\right\|_2^2$$

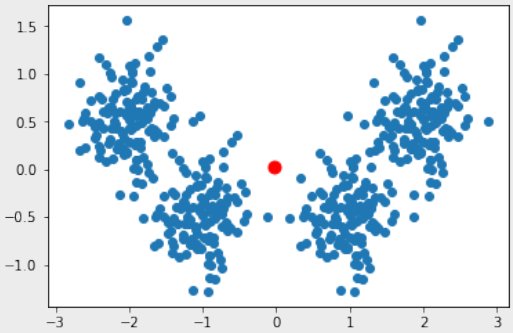
(c) Create a plot (Voronoi-Tesselation) to show how the resulting solution would potentially assign new data points (i.e. show the decision boundaries that separate the clusters).

Solution

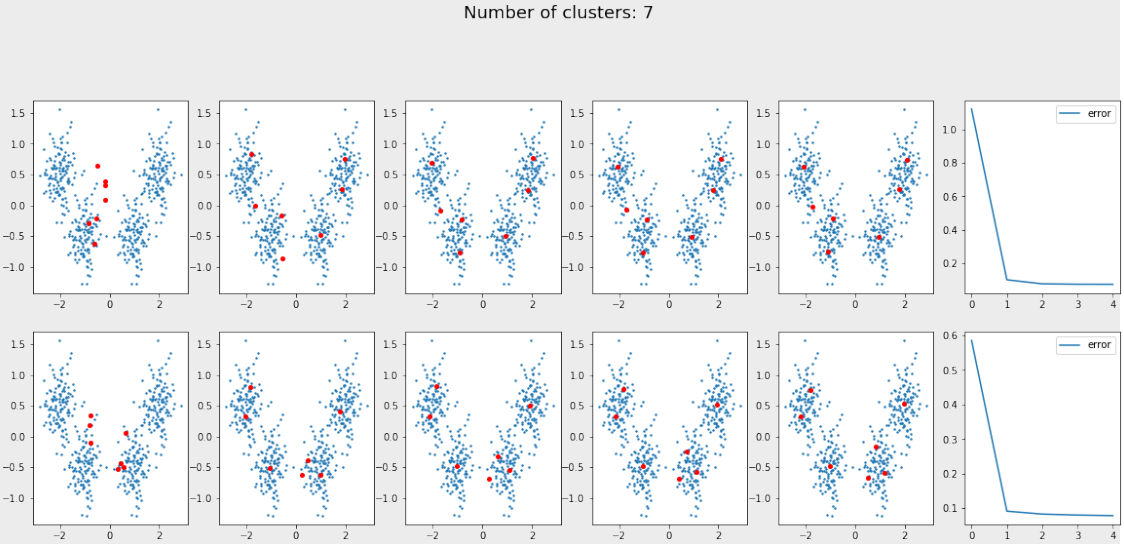
(a)



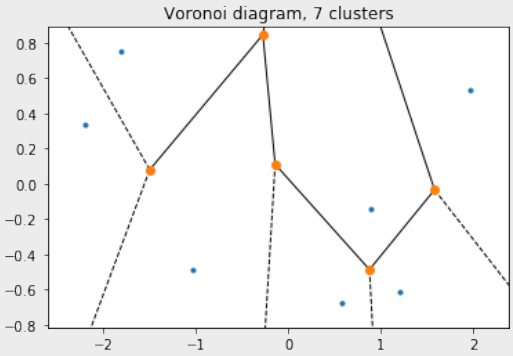
(b)



Sample solution for $M = 7$ (two different initializations)



(c)Voronoi-Tesselation for $M = 7$



Exercise H8.2: Online K-means Clustering (homework, 3 points)

Write a program that implements the *online* version of K-means clustering and partitions the given data set into $M = 4$ clusters. Include the following steps:

A: Initialization –

1. Set the initial position of each prototype $\underline{\mathbf{w}}_q$ randomly around the mean of the entire dateset.
2. Select an initial learning step ε_0
3. Set the maximum number of iterations t_{max} equal to the data set size p.

B: Optimization –

1. Choose a suitable $\tau < 1$ and implement online K-means clustering using the following "annealing" schedule for ε:

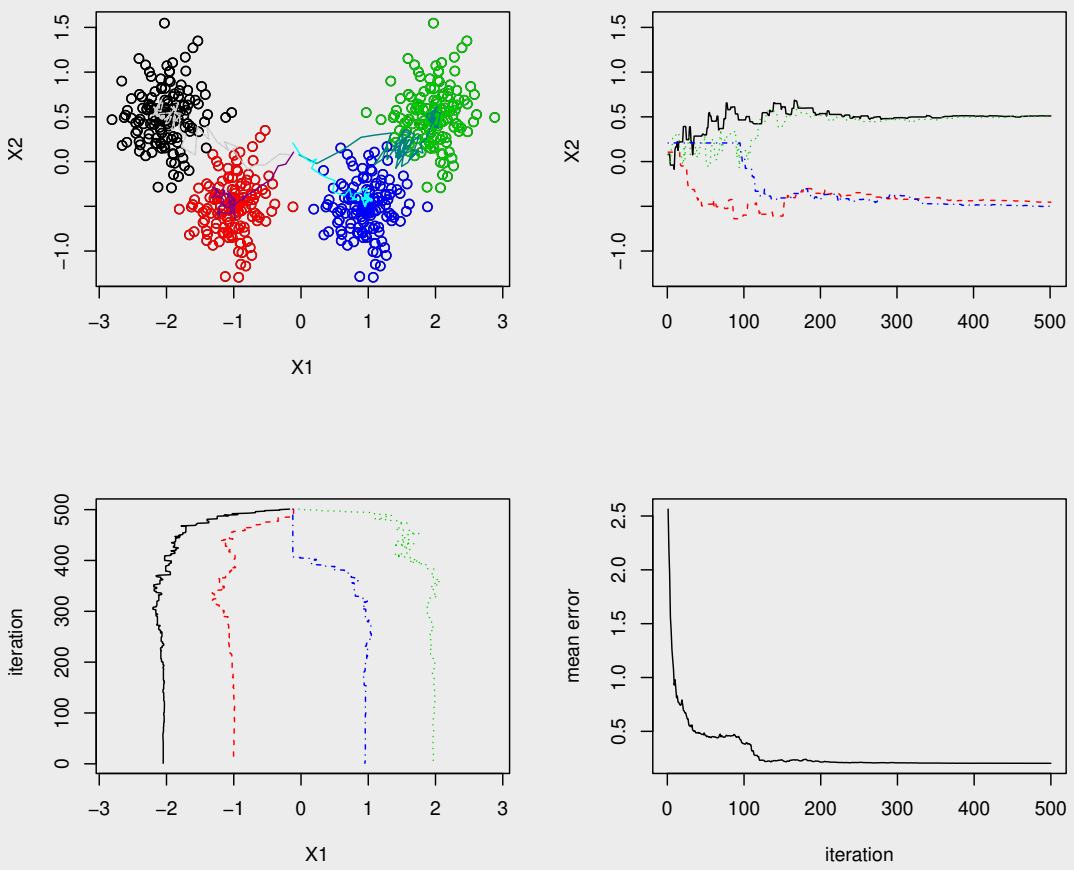
$$\varepsilon_t = \varepsilon_0 \quad \text{for } t = 0, ..., \frac{t_{max}}{4} \qquad \text{and} \qquad \varepsilon_t = \tau\varepsilon_{t-1} \quad \text{for } t = \frac{t_{max}}{4} + 1, ..., t_{max}$$

C: Visualization –

(a) Visualize the data points and prototypes, where data points are colored according to the cluster they will *finally* be assigned to *after* $t_{\max}$. Additionally, show for each cluster the sequence of prototype positions $\underline{\mathbf{w}}_q$ by connecting them with straight lines (each iteration pair $t \to t + 1$ one line per prototype - to trace the position of a prototype over time). Use different colors for the prototype lines than the colours used for the data points.

(b) Plot the error function E (as above) against the iteration t. Is E nonincreasing?

Solution



- The bottom-left plot shows the x_1 coordinate of each prototype and how it moves over time.
- The top-right plot shows this for the x_2 coordinate of each prototype

Exercise H8.3: Soft K-means Clustering (homework, 4 points)

Implement the *soft* K-means algorithm with squared Euclidean distances (the batch version) and apply it to the same data as before. Proceed as follows:

(a) Set $M = 12$ initial prototypes $\mathbf{w}_q$ randomly around the data set mean and choose a convergence tolerance θ.

(b) For fixed β (no annealing), let the optimization procedure run until convergence, that is $\|\mathbf{w}_q^{new} - \mathbf{w}_q^{old}\|_2 < \theta \; \forall q$. Repeat this for different $\beta \in [0.2, 20]$ e.g. in steps of $\Delta\beta = 0.2$. Use the same initial prototypes for all runs.

(c) Visualize the data set, initial and final prototypes for each (fixed) β in one scatter plot.

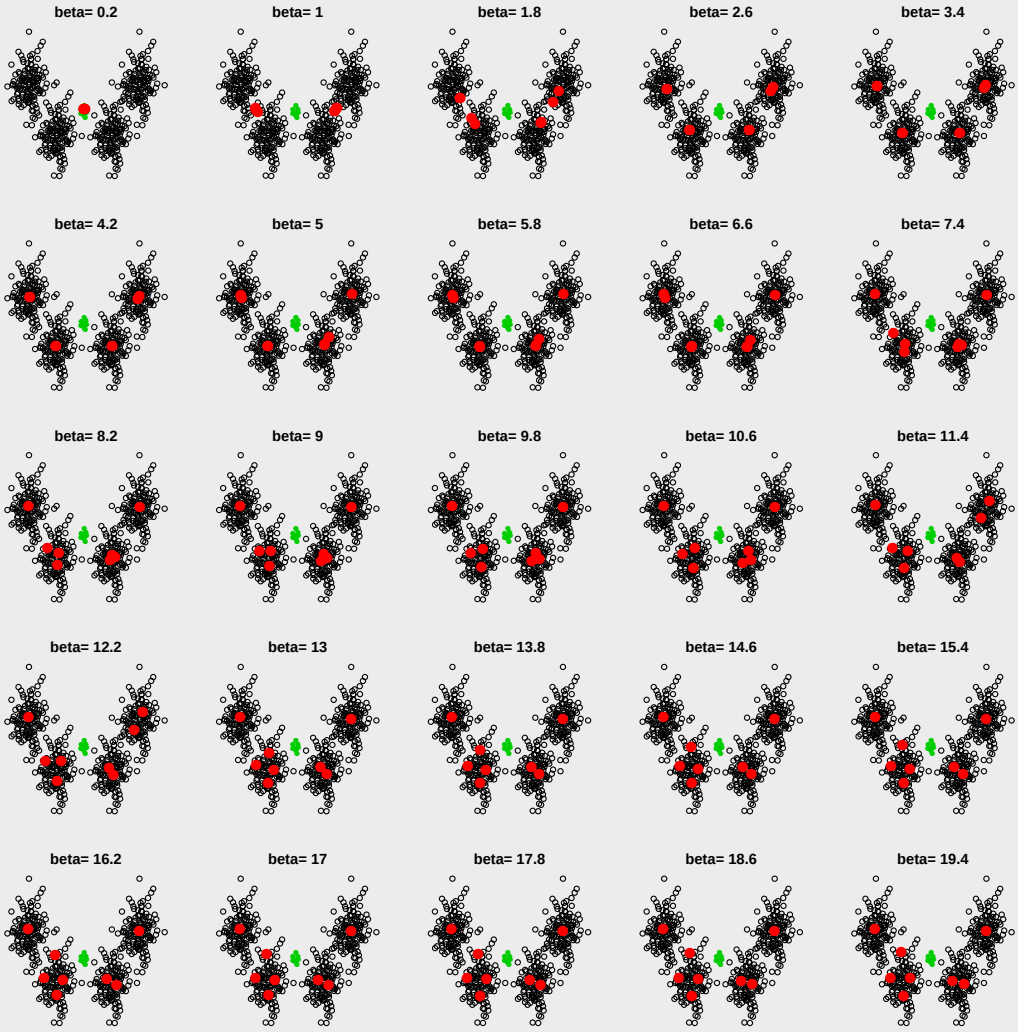
(d) In additional simulations, run the optimization using an annealing schedule: increase β after each iteration. E.g. $\beta_0 = 0.2,\ \tau = 1.1,\ \beta_{t+1} = \tau\beta_t$.

(e) Show the data set, initial and final prototypes of the "annealed" clustering solutions in a scatter plot. Additionally, show the position of each prototype (x_2 coordinate only) as a

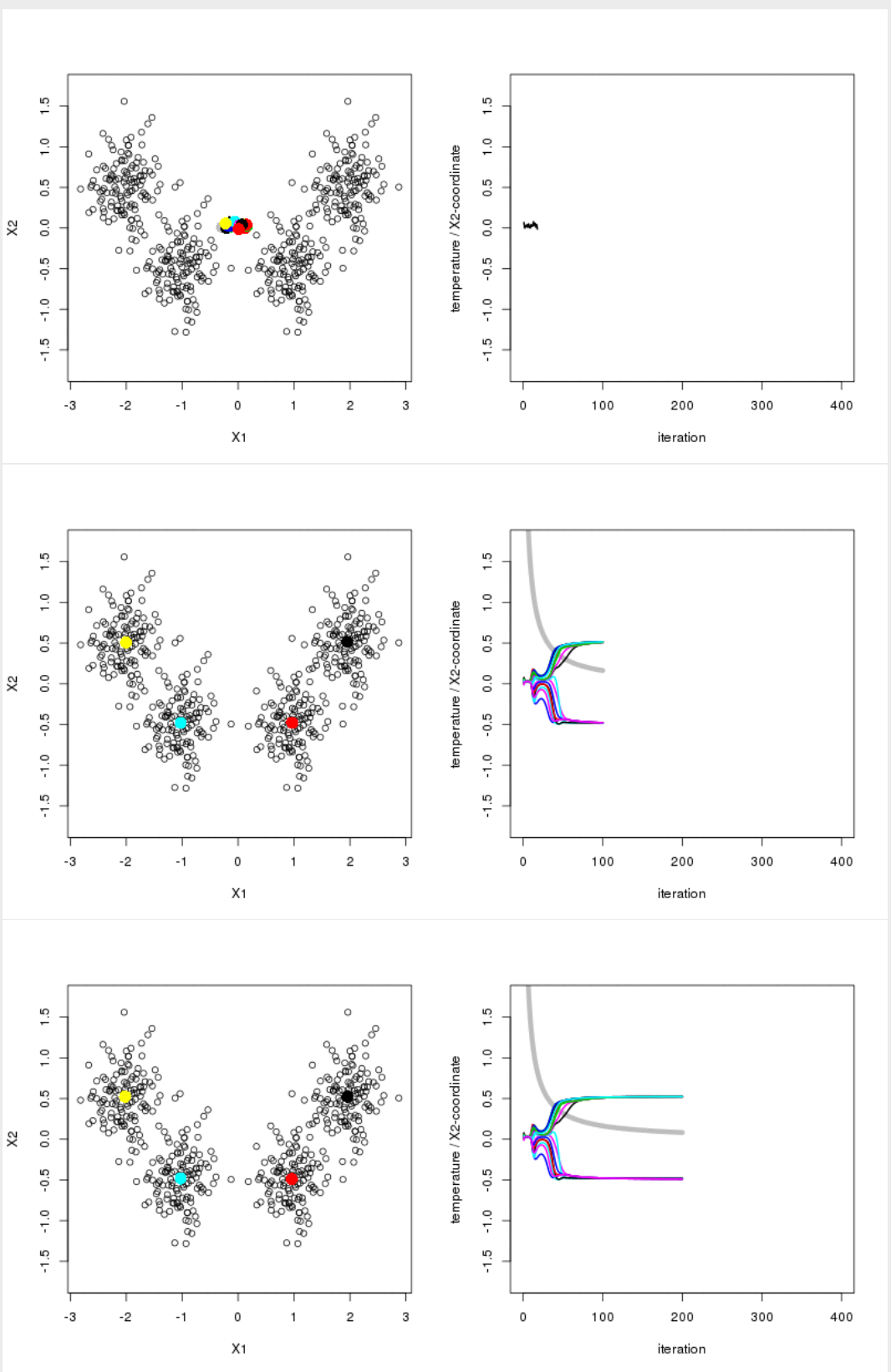
function of t – all in one figure (i.e., $M = 12$ lines). How "soft" are data points assigned now?

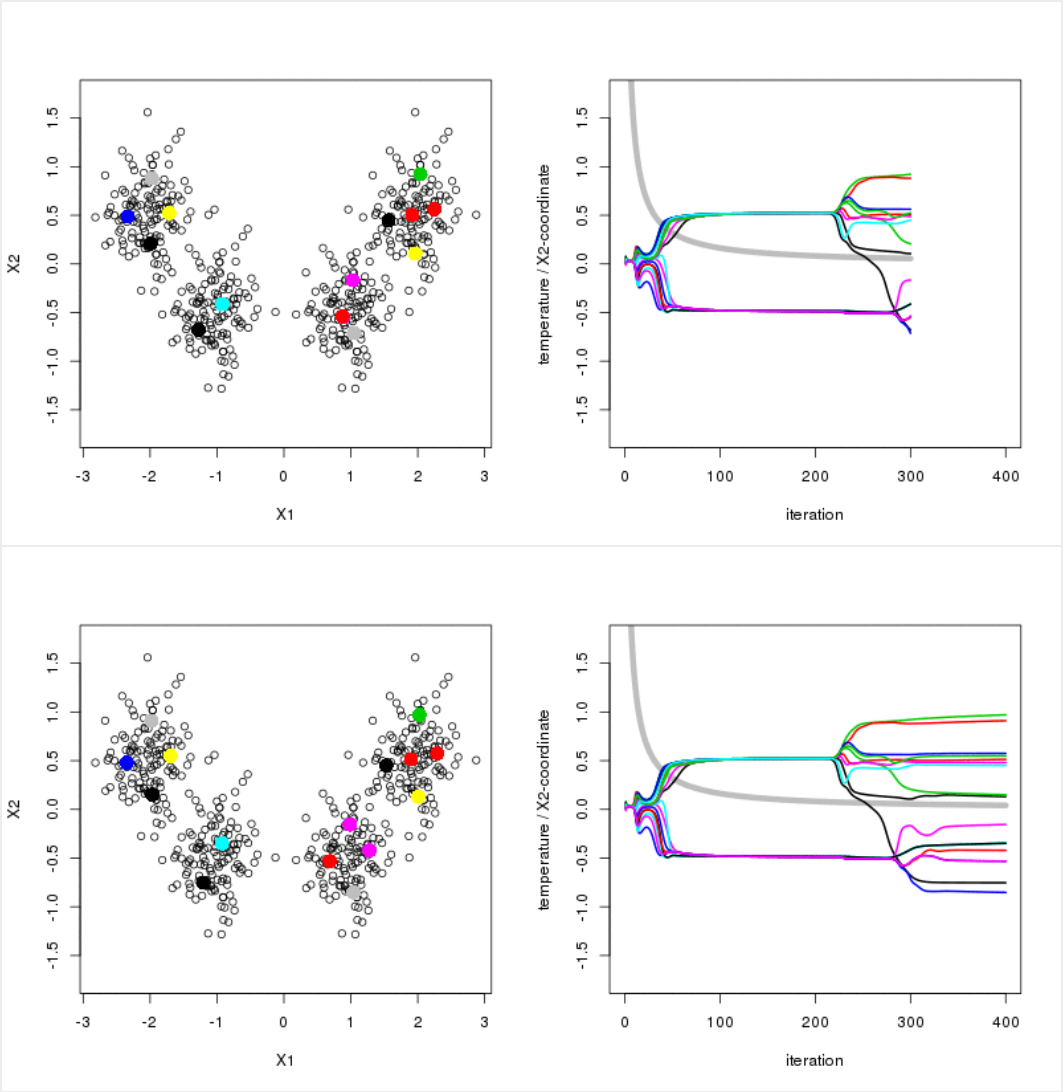
Solution

(c)



For the same and constant number of partitions M, larger values of β result in a more fine-grained partitioning of the data. The reason that we don't see all prototypes is that several overlap.
The cost function is proportional to β, using a larger β leaves more room to further lower the energy. The only way to further reduce the error when β is kept fixed, is to move the prototypes closer to the points assigned to it. This results in less prototypes overlapping and recruting points such that the data is partitioned into smaller clusters.
(e)


X2
X1
temperature / X2-coordinate
iteration



The results above demonstrate how annealing can modulate the "resolution" of the clustering algorithm over time. The number of prototypes M is set once in the beginning and is kept constant. Nonetheless,

1. after 100 iterations: several prototypes will converge to the same location. It appears as if the solution only contains 4 clusters.
2. after 200 iterations: β will have increased causing the cost to increase. In order to minimize the cost, the overlapping prototypes have to "separate". This leads to more and smaller clusters with smaller cluster-variances. This is the only response that the algorithm can perform to counteract the rising energy. The higher-resolution clustering, allows the energy to decrease until it flattens again.
3. after 300 iterations: the resolution of the clustering continues to increase. Maybe less noticeable because the prototypes that were overlapping and have to separate to recruit their own "sub" clusters only requires small changes and leads to small decrease in energy.

Total 10 points.