CS 181 Assignment 3: Clustering and Parameter Estimation

Mark VanMiddlesworth & Shuang Wu

3/11/11

Problem 1.

(a) We have that

$$\begin{aligned}\rho &= P\left(\max_j \left|x_j - y_j\right| \le \varepsilon\right)\\ &= P\left(\left|x_j - y_j\right| \le \varepsilon, \forall j\right)\\ &= P\left(\left|x_j - y_j\right| \le \varepsilon\right)^m\\ &= (2\varepsilon)^m.\end{aligned}$$

(b) Essentially, when we calculate the probability, we are multiplying the fraction of all possible values for each y_j which places it within ε of x_j. The only time when this calculation is not equal to the one in part (a) is if we have ε large enough or $\mathbf{x}$ close enough to the edge of the hypercube that the possible values for one or more y_j's are limited by the boundary of the hypercube. Hence the probability is at most ρ in any case.

(c) The Euclidean distance is given by

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_j (x_j - y_j)^2}.$$

Note that every term in the sum is positive, and that in particular, $d(\mathbf{x}, \mathbf{y}) \ge \sqrt{(x_j - y_j)^2} = \left|x_j - y_j\right|$ for every j. Hence $d(\mathbf{x}, \mathbf{y}) \ge \max_j \left|x_j - y_j\right|$. This implies that $d(\mathbf{x}, \mathbf{y}) \le \varepsilon$ only if $\max_j \left|x_j - y_j\right| \le \varepsilon$, and hence

$$P\big(d(\mathbf{x}, \mathbf{y}) \le \varepsilon\big) \le P\left(\max_j \left|x_j - y_j\right| \le \varepsilon\right) \le \rho$$

by part (b).

(d) From part (c), we see that the probability that a point $\mathbf{y}$ is not within ε of $\mathbf{x}$ is

$$1 - P\big(d(\mathbf{x}, \mathbf{y}) \le \varepsilon\big) \ge 1 - \rho.$$

For n points $\mathbf{y}_1, \ldots \mathbf{y}_n$, the probability that all n will lie beyond a radius ε of $\mathbf{x}$ is

$$\prod_{i=1}^{n} \Big(1 - P\big(d(\mathbf{x}, \mathbf{y}_i) \le \varepsilon\big)\Big) \ge (1 - \rho)^n,$$

and hence the probability that at least one point will be within the radius is

$$1 - \prod_{i=1}^{n} \Big(1 - P\big(d(\mathbf{x}, \mathbf{y}_i) \le \varepsilon\big)\Big) \le 1 - (1 - \rho)^n, \tag{1}$$

We wish to find a lower bound on n such that (1) is bounded below by $1-\delta$. We solve

$$\begin{aligned} 1-\delta &\le 1-(1-\rho)^n \\ \delta &\ge (1-\rho)^n \\ n &\ge \frac{\ln\delta}{\ln(1-\rho)} \end{aligned}$$

which gives us the desired lower bound.

(e) We see that the lower bound for n is given by

$$\frac{\ln\delta}{\ln\big(1-(2\varepsilon)^m\big)}$$

so that as $m\to\infty$, $n\to\infty$. This means that in high dimensional spaces, the number n of points needed to guarantee a non-zero probability that the nearest neighbor of a point $\mathbf{x}$ will be within a radius of ε becomes very large. This makes the HAC clustering algorithm ineffective. For example, suppose that there are m boolean attributes, such that the first l are correlated while the rest are uniformly distributed. We would like to cluster based on the first l attributes, but if we have $m \gg l$, the average distance between two points in of the same cluster is nearly as big as the average distance between points of different clusters, and HAC will have difficulty classifying the clusters correctly.

Problem 2.

(a) We have for ML,

$$P(\mathbf{X}=\mathbf{x}\mid\mathcal{D}) = P\Big(\mathbf{X}=\mathbf{x}\;\Big|\;\arg\max_\theta P(\mathcal{D}\mid\Theta=\theta)\Big);$$

for MAP,

$$P(\mathbf{X}=\mathbf{x}\mid\mathcal{D}) = P\Big(\mathbf{X}=\mathbf{x}\;\Big|\;\arg\max_\theta P(\mathcal{D}\mid\Theta=\theta)P(\Theta=\theta)\Big);$$

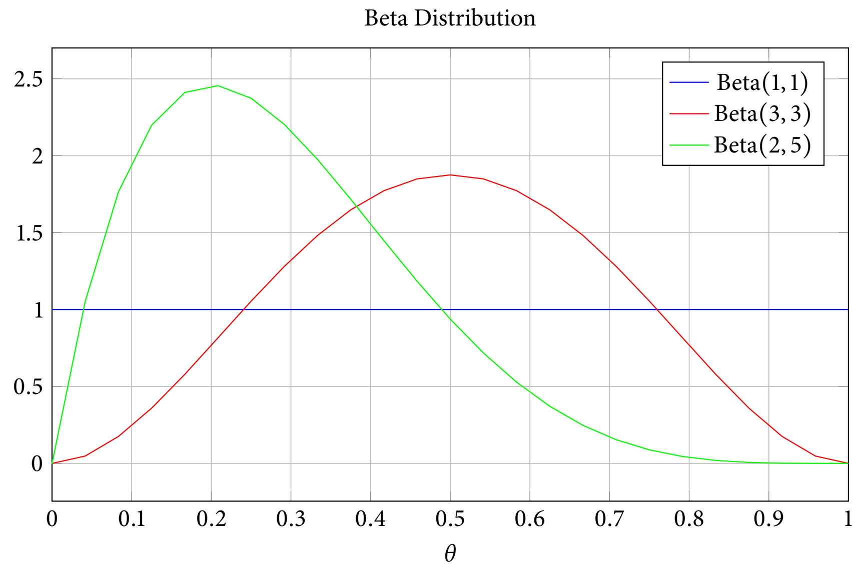
and for FB,

$$P(\mathbf{X}=\mathbf{x}\mid\mathcal{D}) = \int_\theta P(\mathbf{X}=\mathbf{x}\mid\Theta=\theta)P(\Theta=\theta\mid\mathcal{D})\,\mathrm{d}\theta;$$

(b) The MAP method can be considered to be Bayesian since it considers the prior $P(\Theta=\theta)$ on the parameters, which is updated based on observations. In the ML method on the other hand, parameters θ are simply chosen to maximize likelihood based on data, and not on any prior observations.

(c) Since the MAP method considers the prior $P(\Theta=\theta)$ on the parameters and gives us further knowledge about the distributions of parameters, while ML only considers the data in fitting the best parameters, MAP could help prevent overfitting of data which is a downfall of the ML approach. When compared to the FB method, the MAP method is less computationally intensive, and tends to provide a good approximation of the results of the FB method.

(d) We have



We see that Beta$(1, 1)$ is a uniform distribution, and adopting Beta$(1, 1)$ assumes no knowledge on the distribution of θ. Beta$(3, 3)$ is a symmetric density function, and hence adopting this beta as a prior places no bias on the team's probability of winning over losing, as the same numbers of wins or losses are observed. Finally, Beta$(2, 5)$ has a skewed density function, which puts a bias on the team's probability of losing when using this function as the prior since there are more observed losses than wins.

(e) Since Boolean variables have one parameter θ denoting the probability that the variable is 1, their distribution given θ is Bernoulli. Since we need a prior $P(\theta)$ for MAP estimation, Beta distribution is particularly useful in MAP because it is a conjugate prior for data that is Bernoulli distributed. This means that the posterior $P(\Theta = \theta \mid \mathcal{D})$ has the same functional form as the prior $P(\Theta = \theta)$, allowing us to interpret the hyperparameters in the prior.

PROBLEM 3.

(a) The parameters can be written as follows:

$$\begin{aligned}
P(\mathbf{Y} = T) &= \theta_c, \\
P(\mathbf{X}_1 = T \mid \mathbf{Y} = T) &= \theta_1^T, \\
P(\mathbf{X}_1 = T \mid \mathbf{Y} = F) &= \theta_1^F, \\
P(\mathbf{X}_j = T \mid \mathbf{X}_{j-1} = T, \mathbf{Y} = T) &= \theta_j^{TT}, \\
P(\mathbf{X}_j = T \mid \mathbf{X}_{j-1} = T, \mathbf{Y} = F) &= \theta_j^{TF}, \\
P(\mathbf{X}_j = T \mid \mathbf{X}_{j-1} = F, \mathbf{Y} = T) &= \theta_j^{FT}, \\
P(\mathbf{X}_j = T \mid \mathbf{X}_{j-1} = F, \mathbf{Y} = F) &= \theta_j^{FF}.
\end{aligned}$$

We see that for m Boolean attributes and a Boolean target class, the number of independent parameters for $P(\mathbf{Y})$ is 1, the number of independent parameters for $P(\mathbf{X}_j \mid \mathbf{Y} = y)$ is 1 given y, and the number of independent parameters for $P(\mathbf{X}_j \mid \mathbf{X}_{j-1} = x_{j-1}, \mathbf{Y} = y)$ is 1 given x_{j-1} and y. Hence, the total number of independent parameters is

$$1 + 2 \cdot 1 + (2 + 2) \cdot (m - 1) = 4m - 1.$$

(b) Analogously to the derivation in the notes (which confirms that our intuition indeed holds true), we have that

$$\begin{aligned}
\theta_{\text{ML},c} &= \frac{N_T}{n}, \\
\theta^T_{\text{ML},1} &= \frac{N^T_{1T}}{N_T}, \\
\theta^F_{\text{ML},1} &= \frac{N^F_{1T}}{N_F}, \\
\theta^{TT}_{\text{ML},j} &= \frac{N^{TT}_{jT}}{N_{TT}}, \\
\theta^{TF}_{\text{ML},j} &= \frac{N^{TF}_{jT}}{N_{TF}}, \\
\theta^{FT}_{\text{ML},j} &= \frac{N^{FT}_{jT}}{N_{FT}}, \\
\theta^{FF}_{\text{ML},j} &= \frac{N^{FF}_{jT}}{N_{FF}},
\end{aligned}$$

where n is the total number of instances, N_T is the number of instances y_i is T, N^T_{1T} is the number of instances with y_i is T and $x_i1 = 1$, N_{TT} is the number of instances y_i is T and $x_{i(j-1)} = 1$, N^{TT}_{jT} is the number of instances with y_i is T and $x_{i(j-1)} = 1$ and $x_{ij} = 1$, etc. That is, the maximum likelihood for each parameter is exactly the proportion of instances with positive classification among those with the proper required priors.

(c) Again analogously to the lecture notes, we have

$$\begin{aligned}
\langle N_T \rangle &= \sum_{\mathbf{x}_i} P(\mathbf{Y} = T \mid \mathbf{X} = \mathbf{x}_i), \\
\left\langle N^T_{1T} \right\rangle &= \sum_{\mathbf{x}_i} P(\mathbf{Y} = T \mid \mathbf{X} = \mathbf{x}_i), \\
\left\langle N^F_{1T} \right\rangle &= 1 - \sum_{\mathbf{x}_i} P(\mathbf{Y} = T \mid \mathbf{X} = \mathbf{x}_i), \\
\left\langle N^{TT}_{jT} \right\rangle &= \sum_{x_{ij}=T} P(\mathbf{Y} = T, x_{ij-1} = T \mid \mathbf{X} = \mathbf{x}_i), \\
\left\langle N^{TF}_{jT} \right\rangle &= 1 - \sum_{x_{ij}=T} P(\mathbf{Y} = T, x_{ij-1} = T \mid \mathbf{X} = \mathbf{x}_i), \\
\left\langle N^{FT}_{jT} \right\rangle &= \sum_{x_{ij}=T} P(\mathbf{Y} = T, x_{ij-1} = F \mid \mathbf{X} = \mathbf{x}_i), \\
\left\langle N^{FF}_{jT} \right\rangle &= 1 - \sum_{x_{ij}=T} P(\mathbf{Y} = T, x_{ij-1} = F \mid \mathbf{X} = \mathbf{x}_i),
\end{aligned}$$

where

$$\begin{aligned}
P(\mathbf{Y} = T \mid \mathbf{X} = \mathbf{x}_i) &= \frac{p_T}{p_T + p_F} \\
P(\mathbf{Y} = T, x_{ij-1} = T \mid \mathbf{X} = \mathbf{x}_i) &= \frac{p_{TT}}{p_T + p_F} \\
P(\mathbf{Y} = T, x_{ij-1} = F \mid \mathbf{X} = \mathbf{x}_i) &= \frac{p_{FT}}{p_T + p_F}
\end{aligned}$$

with

$$\begin{aligned}
p_T &= \theta_c(\theta_1^T)^{x_{ij}}(1-\theta_1^T)^{1-x_{ij}} \prod_{j=2}^{m} (\theta_j^{TT} + \theta_j^{FT})^{x_{ij}}(2 - \theta_j^{TT} - \theta_j^{FT})^{1-x_{ij}} \\
p_F &= (1-\theta_c)(\theta_1^F)^{x_{ij}}(1-\theta_1^F)^{1-x_{ij}} \prod_{j=2}^{m} (\theta_j^{TF} + \theta_j^{FF})^{x_{ij}}(2 - \theta_j^{TF} - \theta_j^{FF})^{1-x_{ij}} \\
p_{TT} &= \theta_c(\theta_1^T)^{x_{ij}}(1-\theta_1^T)^{1-x_{ij}} \prod_{j=2}^{m} (\theta_j^{TT})^{x_{ij}}(1 - \theta_j^{TT})^{1-x_{ij}} \\
p_{FT} &= \theta_c(\theta_1^T)^{x_{ij}}(1-\theta_1^T)^{1-x_{ij}} \prod_{j=2}^{m} (\theta_j^{FT})^{x_{ij}}(1 - \theta_j^{FT})^{1-x_{ij}}.
\end{aligned}$$

(d) The algorithm consists of two steps. We first initialize $\theta^{(0)}$ to some initial values, and then we repeat the following until convergence:

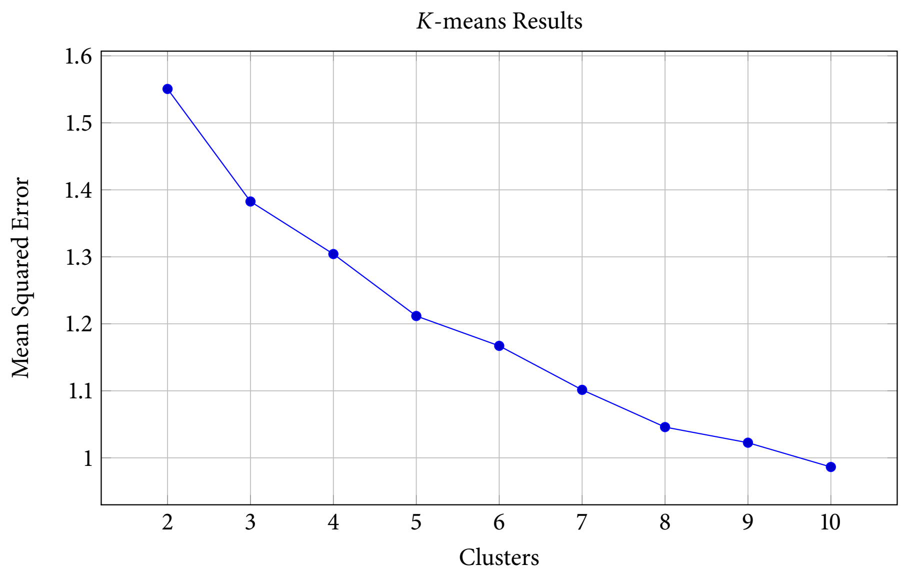
E step: Determine for each i p_T, p_F, p_{TT}, p_{FT} from the $\theta^{(\text{old})}$, and use this to calculate the posterior probabilities $P(\mathbf{Y} = T \mid \mathbf{X} = \mathbf{x}_i)$, $P(\mathbf{Y} = T, x_{ij-1}$, and $P(\mathbf{Y} = T, x_{ij-1}$.

M step: Using the probabilities from the E step, compute the expected values of each sufficient statistic as in part (c), and use these in the equations in part (b) to determine $\theta^{(\text{new})}$, and replace the old parameters with the new parameters.

(e) The advantage of using this model is that our classifier is no longer a linear separator, and so can learn more complex classifications like XOR. However, this comes at the cost of significant computation complexity compared to Autoclass. This model would be useful for clustering problems where we are trying to classify based on pairs of adjacent attributes (i.e., whether or not we have dots of size at least 2 pixels), rather than independent single attributes. Because the ordering of the attributes is important, we would expect that any reordering of the attributes would significantly effect the performance of the model negatively when used in such applications.

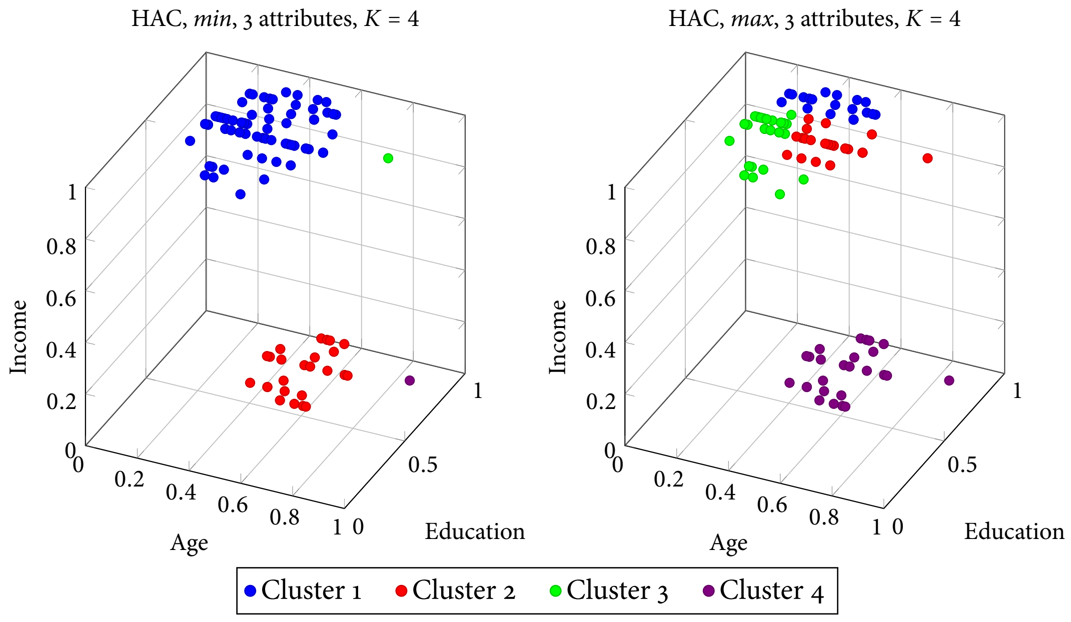
Problem 4.

(a) (i) We have the following plot:



(ii) With more clusters, the mean squared error falls. However, too many clusters also defeats the purpose of clustering in the first place. Hence we are interested in finding a good balance between minimizing the mean squared error and minimizing the number of clusters. Based on the plot above, we might choose $K = 5$ as it seems like this is the 'elbow' in the plot at which the marginal improvement in mean squared error decreases as the number of clusters continues to increase.

(b) (i)



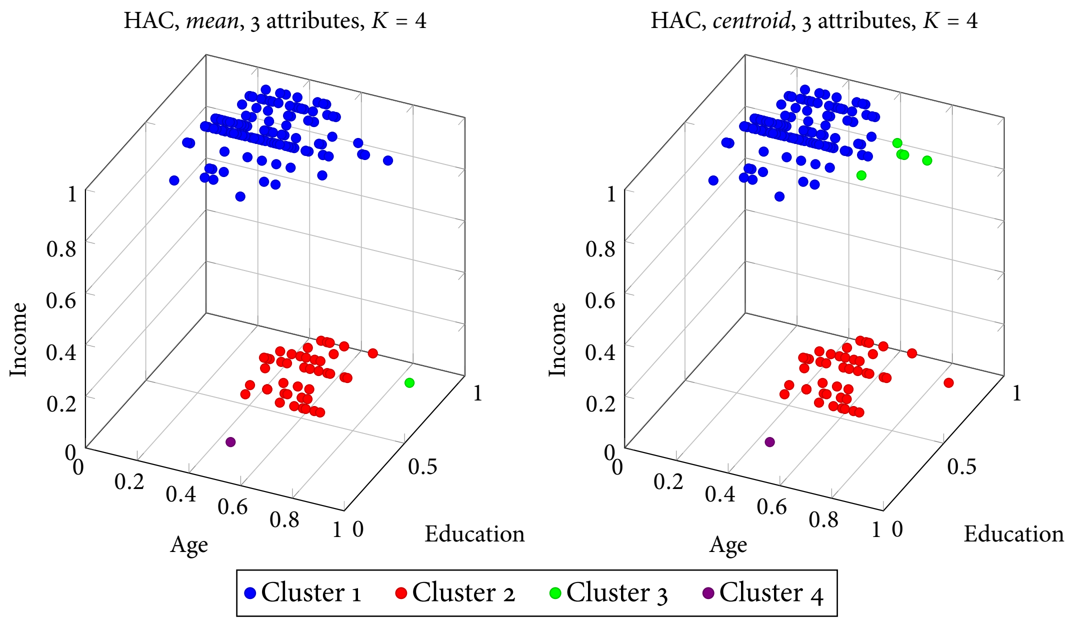
The number of instances in each cluster is given below:

Cluster	*min*	*max*
1	73	19
2	25	23
3	1	32
4	1	26

The *min* metric formed two large clusters and two single point clusters that appear to be outliers. This makes sense since the *min* metric grows clusters by chaining together points that are close to each other, and prefers points which are close to other points within a cluster. We also note that in this case we would probably pick these 4 clusters by hand.

The *max* metric seems to prefer clusters that are more compact, and split the large group of points for Income = 1 into three different clusters. This makes sense since the *max* metric tries to minimize the diameter of each cluster, and prefers more convex clusters. The number of points is more balanced compared to the *min* metric, which may be desirable in some problems.

(ii)



The number of instances in each cluster is given below:

Cluster	*mean*	*centroid*
1	152	147
2	46	47
3	1	5
4	1	1

Both *mean* and *centroid* formed roughly the same large clusters, and differed only in how they chose

to classify outliers. The *mean* metric formed two clusters of outliers of single points each, and two large clusters from the remaining data. This makes sense since the mean metric clusters points which are close together on average, and isolates the two outlying points.
The *centroid* metric formed two clusters of outliers, one of a single point, and one with five points. The boundaries of the clusters are more circular in shape, which makes sense since the *centroid* classifier attempts to collect groups of points which share similar centroids. This behavior seems like the most balanced compromise between the *min* and *max* metrics, preferring clusters that are slightly more balanced in size than *min*, but better at separating outliers in this particular case than *max*.