CM146 Homework#5

Zhaoxing Deng, 005024802

March 16, 2018

1 Naïve Bayes over Multinomial Distribution

(a) We lose the information about the order of words in D_i. We only know the numbers of each word in the documents.

(b) $\log Pr(D_i, y_i) = \log Pr(D_i \mid y_i)Pr(y_i) = \log((Pr(D_i \mid y_i = 1)Pr(y_i = 1))^{[y_i=1]}(Pr(D_i \mid y_i = 0)Pr(y_i = 0))^{[y_i=0]}) = \log((Pr(D_i \mid y_i = 1)Pr(y_i = 1))^{y_i}(Pr(D_i \mid y_i = 0)Pr(y_i = 0)^{1-y_i}) = \log(Pr(D_i \mid y_i = 1)^{y_i}\theta^{y_i}Pr(D_i \mid y_i = 0)^{1-y_i}(1-\theta)^{1-y_i}) = (\log(Pr(D_i \mid y_i = 1) + \log\theta)y_i + (\log(Pr(D_i \mid y_i = 0) + \log(1-\theta))(1-y_i) = y_i(\log\theta + \log\frac{n!}{a_i!b_i!c_i!}\alpha_1^{a_i}\beta_1^{b_i}\gamma_1^{c_i}) + (1-y_i)(\log(1-\theta) + \log\frac{n!}{a_i!b_i!c_i!}\alpha_0^{a_i}\beta_0^{b_i}\gamma_0^{c_i}) = y_i(\log\theta + \log\frac{n!}{a_i!b_i!c_i!} + a_i\log\alpha_1 + b_i\log\beta_1 + c_i\log\gamma_1) + (1-y_i)(\log(1-\theta) + \log\frac{n!}{a_i!b_i!c_i!} + a_i\log\alpha_0 + b_i\log\beta_0 + c_i\log\gamma_0)$

$\log Pr(D_i, y_i) = \log\frac{n!}{a_i!b_i!c_i!} + y_i(\log\theta + a_i\log\alpha_1 + b_i\log\beta_1 + c_i\log\gamma_1) + (1-y_i)(\log(1-\theta) + a_i\log\alpha_0 + b_i\log\beta_0 + c_i\log\gamma_0)$

(c) The maximum likelihood estimate is $\underset{\alpha_1,\beta_1,\gamma_1,\alpha_0,\beta_0,\gamma_0}{\operatorname{argmax}} \sum_{i=1}^m \log Pr(D_i, y_i)$.

Let $\frac{\partial}{\partial\alpha_1}\sum_{i=1}^m \log Pr(D_i, y_i) = 0$. Since $\alpha_1 + \beta_1 + \gamma_1 = 1$, we can choose α_1,γ_1 to be independent parameters, and $\beta_1 = 1 - \alpha_1 - \gamma_1$.

$\frac{\partial}{\partial\alpha_1}\sum_{i=1}^m \log Pr(D_i, y_i) = 0 \Rightarrow \frac{\partial}{\partial\alpha_1}\sum_{i=1}^m y_i(a_i\log\alpha_1 + b_i\log\beta_1) = \frac{\partial}{\partial\alpha_1}\sum_{i=1}^m y_i(a_i\log\alpha_1 + b_i\log(1-\alpha_1-\gamma_1)) = \frac{\partial}{\partial\alpha_1}\sum_{i=1}^m y_i(\frac{a_i}{\alpha_1} - \frac{b_i}{1-\alpha_1-\gamma_1}) = \frac{\partial}{\partial\alpha_1}\sum_{i=1}^m y_i(\frac{a_i(1-\alpha_1-\gamma_1)-b_i\alpha_1}{\alpha_1(1-\alpha_1-\gamma_1)}) = 0 \Rightarrow \alpha_1 = \frac{\sum_{i=1}^m y_i}{\sum_{i=1}^m y_i(a_i+b_i)}(1-\gamma_1)$

Using the same process, we can get

$\frac{\partial}{\partial\gamma_1}\sum_{i=1}^m \log Pr(D_i, y_i) = 0 \Rightarrow \frac{\partial}{\partial\gamma_1}\sum_{i=1}^m y_i(b_i\log\beta_1 + c_i\log\gamma_1) = \frac{\partial}{\partial\gamma_1}\sum_{i=1}^m y_i(\frac{c_i}{\gamma_1} - \frac{b_i}{1-\alpha_1-\gamma_1}) = 0 \Rightarrow \gamma_1 = \frac{\sum_{i=1}^m y_i}{\sum_{i=1}^m y_i(b_i+c_i)}(1-\alpha_1)$

Let $C_1 = \frac{\sum_{i=1}^m y_i}{\sum_{i=1}^m y_i(a_i+b_i)}$, $C_2 = \frac{\sum_{i=1}^m y_i}{\sum_{i=1}^m y_i(b_i+c_i)}$, then

$\alpha_1 = \frac{C_1(1-C_2)}{1-C_1C_2}$, $\beta_1 = \frac{(1-C_1)(1-C_2)}{1-C_1C_2}$, $\gamma_1 = \frac{C_2(1-C_1)}{1-C_1C_2}$

Similarly, we can get

$\alpha_0 = \frac{C_3(1-C_4)}{1-C_3C_4}$, $\beta_0 = \frac{(1-C_3)(1-C_4)}{1-C_3C_4}$, $\gamma_0 = \frac{C_4(1-C_3)}{1-C_3C_4}$ where $C_3 = \frac{\sum_{i=1}^m (1-y_i)}{\sum_{i=1}^m (1-y_i)(a_i+b_i)}$,
$C_4 = \frac{\sum_{i=1}^m (1-y_i)}{\sum_{i=1}^m (1-y_i)(b_i+c_i)}$
After simplification, we have the final results
$\alpha_1 = \frac{\sum_{i=1}^m y_i a_i}{\sum_{i=1}^m y_i n}$, $\beta_1 = \frac{\sum_{i=1}^m y_i b_i}{\sum_{i=1}^m y_i n}$,$\gamma_1 = \frac{\sum_{i=1}^m y_i c_i}{\sum_{i=1}^m y_i n}$,$\alpha_0 = \frac{\sum_{i=1}^m (1-y_i)a_i}{\sum_{i=1}^m (1-y_i)n}$,$\beta_0 = \frac{\sum_{i=1}^m (1-y_i)b_i}{\sum_{i=1}^m (1-y_i)n}$,$\gamma_0 = \frac{\sum_{i=1}^m (1-y_i)c_i}{\sum_{i=1}^m (1-y_i)n}$

2 Hidden Markov Models

(a) Two unspecified transition probabilities are
$q_{21} = P(q_{t+1} = 2 \mid q_t = 1) = 0$
$q_{22} = P(q_{t+1} = 2 \mid q_t = 2) = 0$
Two unspecified output probabilities are
$e_1(B) = P(O_t = B \mid q_t = 1) = 0.01$
$e_2(A) = P(O_t = A \mid q_t = 2) = 0.49$
(b) $P(A) = \pi_1 e_1(A) + \pi_2 e_2(A) = 0.735,\ P(B) = \pi_1 e_1(B) + \pi_2 e_2(B) = 0.265$
Since P(A) > P(B), A is the most frequent output symbol to appear in the first positon of sequences.
(c) There are 8 different sequences of three output symbols. That is AAA, AAB, ABA, ABB, BAA, BAB. BBA. BBB.
When t = 1, $q_{11} = q_{12} = 1$, which means at t = 2, the state will be transfered to 1. Then the states will stay in 1 afterwards.
So the only transition states will be 111 and 211.
$P(AAA) = P(AAA, 111)+P(AAA, 211) = \pi_1 p(A|1)p(A|1)p(A|1)+\pi_2 p(A|2)p(A|1)p(A|1) = 0.7203735$
Thus, AAA has the highest probability of being generated from this HMM model.

3 Facial Recognition by using K-Means and K-Medoids

(a) The minimum value of $J(\mathbf{c}, \mu, k)$ will be zero. Because we can have n class centers which just being assigned to each data point and the total distance will be zero. That is $\mathbf{c}^{(i)} = i$,$\mu_i = \mathbf{x}^{(i)}$, $k = n$. That is bad because it doesn't tell you anything about the data.
(b) In `Cluster`, I compute the centroids by taking the average fo all features of the data points. In `medoid` function, I calculate the total distances of each point to the other points and find the medoid of one cluster.

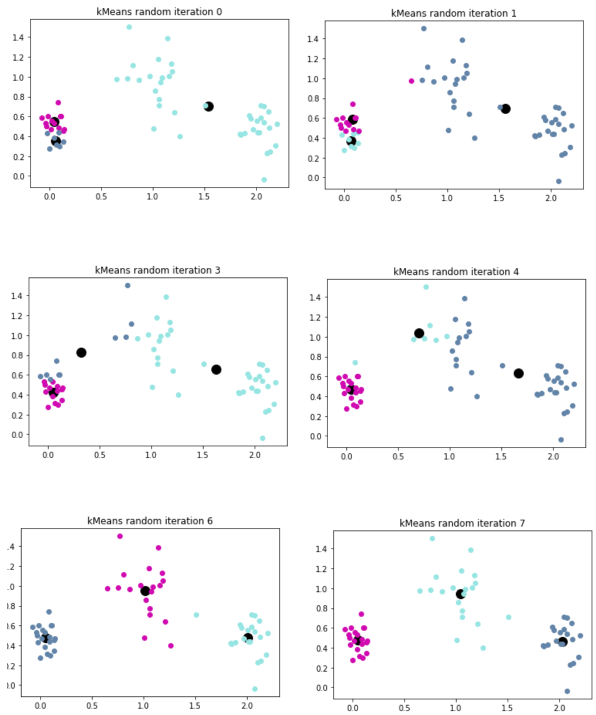

Figure 1: kMeans random iterations

In `Clusterset`, I just returen a set of centroids and medoids.
(c) In `random_init`, I choose k points by using np.random.choice and using a while loop to compare each ponit with other points to avoid duplication.
In `kMeans`, I first use either cheat_init or random_init to initialize clusters. Then I divide the data points in to k clusters by comparing the distances of each point to the class centers. Then on each iteration, using `ClusterSet.centroids` , create a new `ClusterSet` object and update the centroids and assign new cluster assignments to every points. This process will continue until the clustering no longer changes.
(d) Plots for the k-means clustering is shown below.

(e) Refactor `kMeans` by using a helper function `kAverages` to determine how to calculate the average of points in a cluster, whether to use `ClusterSet.centroids` or `ClusterSet.medoids`. Implement`kMedoids` and plot for each iteration.

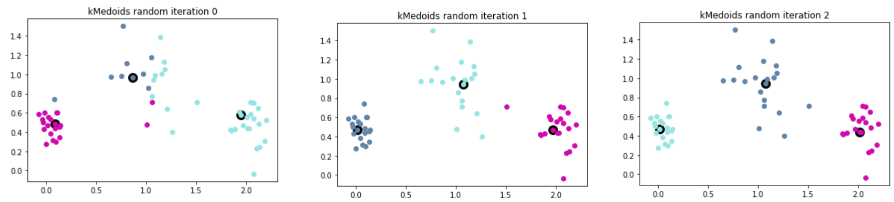

Figure 2: kMedoids random iterations

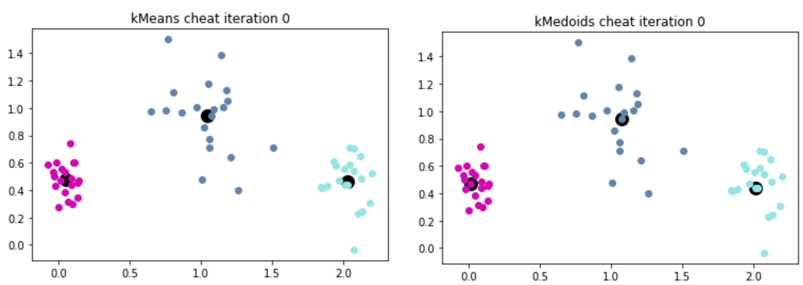

Figure 3: kMeans and kMedoids cheat inerations

(f) Implement `chet_init` by grouping points into k clusters based on label information and return medoid of each cluster as initial centers. The results are shown below. Both kMeans and kMedoids clustering terminate after one iteration. Because by cheating, we can find best class centers at the beginning.