CSM146, Winter 2018
Problem Set 5: Naïve Bayes, Hidden Markov Models
Due Mar 15, 2018 at 11:59 pm

1 Naïve Bayes over Multinomial Distribution [30 pts]

In this question, we will look into training a naïve Bayes classifier with a model that uses a multinomial distribution to represent documents. Assume that all the documents are written in a language which has only three words a, b, and c. All the documents have exactly n words (each word can be either a, b, or c). We are given a labeled document collection $\{D_1, D_2, \ldots, D_m\}$. The label y_i of document D_i is 1 or 0, indicating whether D_i is "`good`" or "`bad`".

This model uses the multinominal distribution in the following way: Given the i^{th} document D_i, we denote by a_i (respectively, b_i, c_i) the number of times that word a (respectively, b, c) appears in D_i. Therefore, $a_i + b_i + c_i = |D_i| = n$. We define

$$\Pr(D_i|y = 1) = \frac{n!}{a_i!b_i!c_i!}\alpha_1^{a_i}\beta_1^{b_i}\gamma_1^{c_i}$$

where α_1 (respectively, β_1, γ_1) is the probability that word a (respectively, b, c) appears in a "`good`" document. Therefore, $\alpha_1 + \beta_1 + \gamma_1 = 1$. Similarly,

$$\Pr(D_i|y = 0) = \frac{n!}{a_i!b_i!c_i!}\alpha_0^{a_i}\beta_0^{b_i}\gamma_0^{c_i}$$

where α_0 (respectively, β_0, γ_0) is the probability that word a (respectively, b, c) appears in a "`bad`" document. Therefore, $\alpha_0 + \beta_0 + \gamma_0 = 1$.

(a) **(3 pts)** What information do we lose when we represent documents using the aforementioned model? **Solution:** The aforementioned model loses the order of the words in a document and ignores the semantic meanings of the words.

(b) **(7 pts)** Write down the expression for the log likelihood of the document D_i, $\log\Pr(D_i, y_i)$. Assume that the prior probability, $Pr(y_i = 1)$ is θ. **Solution:** Notice that youre asked to generate the joint log-likelihood of the data and the label i.e. $logPr(D_i, y_i)$ and not just the data conditioned on the label. The joint probability $Pr(D_i, y_i)$ can be written as $Pr(D_i|y_i)Pr(y_i)$. Assume that the prior probability of generating $Pr(Y_i = 1)$ is ; naturally $Pr(Y_i = 0)$ is 1. Now using the fact that $y_i \in \{0, 1\}$, we can conveniently write down the joint probability as

$$\begin{aligned} Pr(D_i, y_i) &= Pr(Y = y_i)Pr(D_i|Y = y_i) \\ &= (Pr(Y = 1)Pr(D_i|Y = 1))^{y_i}(Pr(Y = 0)Pr(D_i|Y = 0))^{1-y_i} \end{aligned}$$

Using the Naïve Bayes model we're given, we can write $Pr(D_i, y_i)$ as

$$Pr(D_i, y_i) = \left(\eta\frac{n!}{a_i!b_i!c_i!}\alpha_1^{a_i}\beta_1^{b_i}\gamma_1^{c_i}\right)^{y_i}\left((1-\eta)\frac{n!}{a_i!b_i!c_i!}\alpha_0^{a_i}\beta_0^{b_i}\gamma_0^{c_i}\right)^{1-y_i}$$

Finally, we can express the log-likelihood of the document D_i, L_i, as

$$L_i = \log Pr(D_i, y_i) = y_i \left[\log \eta + \log\left(\frac{n!}{a_i! b_i! c_i!}\right) + a_i \log \alpha_1 + b_i \log \beta_1 + c_i \log \gamma_1\right]$$
$$+ (1 - y_i)\left[\log(1-\eta) + \log\left(\frac{n!}{a_i! b_i! c_i!}\right) + a_i \log \alpha_0 + b_i \log \beta_0 + c_i \log \gamma_0\right]$$

(c) **(20 pts)** Derive the expression for the maximum likelihood estimates for parameters α_1, β_1, γ_1, α_0, β_0, and γ_0. **Solution:** We derive the parameter values by maximizing the joint likelihood $L = \sum_i L_i$ derived in problem 1(b). However, notice that the parameters α_1, β_1, and γ_1 are not independent of each other (same goes for α_0, β_0, and γ_0) and so we cannot directly differentiate L to obtain MLE values. We shall have to substitute one of these values in terms of the others to obtain what we desire. Lets substitute $\gamma_1 = 1 - \alpha_1 - \beta_1$. We will derive the values of α_1 and β_1 and those will automatically give the value of γ_1 as per the above relation. First, we aim at deriving the expression for α_1. We have that $\frac{\partial \gamma_1}{\partial \alpha_1} = -1$. Differentiating L as given in previous problem, we get

$$\begin{aligned}
\frac{\partial L}{\partial \alpha_1} &= \sum_i y_i \left(\frac{a_i}{\alpha_1} + \frac{c_i}{\gamma_1}\frac{\partial \gamma_1}{\partial \alpha_1}\right) = \sum_i y_i \left(\frac{a_i}{\alpha_1} - \frac{c_i}{\gamma_1}\right) = \sum_i y_i \left(\frac{a_i \gamma_1 - c_i \alpha_1}{\alpha_1 \gamma_1}\right) = 0 \\
\therefore & \quad \gamma_1 \sum_i y_i a_i = \alpha_1 \sum_i y_i c_i \\
\therefore \alpha_1 &= \gamma_1 \frac{\sum_i y_i a_i}{\sum_i y_i c_i} \qquad (10)
\end{aligned}$$

$$\begin{aligned}
\text{Similarly,} \qquad \beta_1 &= \gamma_1 \frac{\sum_i y_i b_i}{\sum_i y_i c_i} \\
\text{But,} \qquad \alpha_1 + \beta_1 + \gamma_1 &= 1 \\
\therefore \gamma_1 \left[\frac{\sum_i y_i a_i}{\sum_i y_i c_i} + \frac{\sum_i y_i b_i}{\sum_i y_i c_i} + 1\right] &= 1 \\
\therefore \gamma_1 \left[\frac{\sum_i y_i (a_i + b_i + c_i)}{\sum_i y_i c_i}\right] &= 1 \\
\text{Since, } a_i + b_i + c_i = n, \quad \gamma_1 &= \frac{\sum_i y_i c_i}{n \sum_i y_i} \qquad (11) \\
\text{Substituting in Eq. 10,} \quad \alpha_1 &= \frac{\sum_i y_i a_i}{n \sum_i y_i} \qquad (12) \\
\text{and } \beta_1 &= \frac{\sum_i y_i b_i}{n \sum_i y_i} \qquad (13)
\end{aligned}$$

Doing a similar derivation as above with α_0, β_0, and γ_0, we can show that

$$\alpha_0 = \frac{\sum_i (1-y_i)a_i}{n\sum_i (1-y_i)} \tag{14}$$

$$\beta_0 = \frac{\sum_i (1-y_i)b_i}{n\sum_i (1-y_i)} \tag{15}$$

$$\text{and} \quad \gamma_0 = \frac{\sum_i (1-y_i)c_i}{n\sum_i (1-y_i)} \tag{16}$$

Note that this is not the only way to solve this. We can also work out the expressions using KKT conditions to add the constraint in L. Let

$$\begin{aligned}
L' &= \sum_i y_i \left[\log\eta + \log\left(\frac{n!}{a_i!b_i!c_i!}\right) + a_i\log\alpha_1 + b_i\log\beta_1 + c_i\log\gamma_1\right] \\
&\quad + \sum_i (1-y_i)\left[\log(1-\eta) + \log\left(\frac{n!}{a_i!b_i!c_i!}\right) + a_i\log\alpha_0 + b_i\log\beta_0 + c_i\log\gamma_0\right] \\
&\quad - \lambda_1(\alpha_1+\beta_1+\gamma_1-1) - \lambda_0(\alpha_0+\beta_0+\gamma_0-1) \\
\therefore \frac{\partial L'}{\partial \alpha_1} &= \sum_i y_i\left(\frac{a_i}{\alpha_1}\right) - \lambda_1 = \frac{\sum_i y_i a_i}{\alpha_1} - \lambda_1 = 0 \\
\therefore \alpha_1 &= \frac{1}{\lambda}\sum_i y_i a_i
\end{aligned}$$

$$\text{Similarly, } \beta_1 = \frac{1}{\lambda}\sum_i y_i b_i \quad \text{and} \quad \gamma_1 = \frac{1}{\lambda}\sum_i y_i c_i$$

$$\alpha_1+\beta_1+\gamma_1 = 1 = \frac{1}{\lambda}\left(\sum_i y_i(a_i+b_i+c_i)\right) \implies \lambda = n\sum_i y_i$$

Substituting λ gives the same expressions as Eq. (11) to (13).

Submission note: You need not show the derivation of all six parameters separately. Some parameters are symmetric to others, and so, once you derive the expression for one, you can directly write down the expression for others.

2 Hidden Markov Models [15 pts]

Consider a Hidden Markov Model with two hidden states, $\{1, 2\}$, and two possible output symbols, $\{A, B\}$. The initial state probabilities are

$$\pi_1 = P(q_1 = 1) = 0.49 \quad \text{and} \quad \pi_2 = P(q_1 = 2) = 0.51,$$

the state transition probabilities are

$$q_{11} = P(q_{t+1} = 1|q_t = 1) = 1 \quad \text{and} \quad q_{12} = P(q_{t+1} = 1|q_t = 2) = 1,$$

and the output probabilities are

$$e_1(A) = P(O_t = A|q_t = 1) = 0.99 \quad \text{and} \quad e_2(B) = P(O_t = B|q_t = 2) = 0.51.$$

Throughout this problem, make sure to show your work to receive full credit.

(a) **(5 pts)** There are two unspecified transition probabilities and two unspecified output probabilities. What are the missing probabilities, and what are their values?

Solution: The missing state transition probabilities are

$$a_{12} = P(q_{t+1} = 2 | q_t = 1) = 0 \quad \text{and}$$

and the missing output probabilities are

$$b_1(B) = P(O_t = B | q_t = 1) = 0.01 \quad \text{and}$$

(b) **(5 pts)** What is the most frequent output symbol (A or B) to appear in the first position of sequences generated from this HMM?

Solution: The probability of generating A in the first position is $P(O_1 = A) = \pi_1 b_1(A) + \pi_2 b_2(A) = (0.49)(0.99) + (0.51)(0.49) = 0.735$, and the probability of generating B in the first position is $P(O_1 = B) = \pi_1 b_1(B) + \pi_2 b_2(B) = (0.49)(0.01) + (0.51)(0.51) = 0.265$. (As a sanity check, these probabilities do sum to 1.) Thus, the most frequent output symbol to appear in the first position of sequences generated from this HMM is A.

(c) **(5 pts)** What is the sequence of three output symbols that has the highest probability of being generated from this HMM model?

Solution: Thanks to the transition rules for this HMM, every turn after the first one will find the system in state $q = 1$, and so the output probabilities for all turns after the first are independent of each other and of the first turn. For $q = 1$, A is 99 times more likely to be outputted than B. Thus, we see that in every turn, A is independently the most likely output, and so the most likely initial sequence of three output symbols for this HMM is AAA.

3 Facial Recognition by using K-Means and K-Medoids [55 pts]

Machine learning techniques have been applied to a variety of image interpretation problems. In this problem, you will investigate facial recognition, which can be treated as a clustering problem ("separate these pictures of Joe and Mary").

For this problem, we will use a small part of a huge database of faces of famous people (Labeled Faces in the Wild [LFW] people dataset[1]). The images have already been cropped out of the original image, and scaled and rotated so that the eyes and mouth are roughly in alignment; additionally, we will use a version that is scaled down to a manageable size of 50 by 37 pixels (for a total of 1850 "raw" features). Our dataset has a total of 1867 images of 19 different people. You will explore clustering methods such as k-means and k-medoids to the problem of facial recognition on this dataset.

Download the starter files from the course website. It contains the following source files:

- `util.py` – Utility methods for manipulating data.
- `cluster.py` – Code for the `Point`, `Cluster`, and `ClusterSet` classes.
- `faces.py` – Main code

[1] http://vis-www.cs.umass.edu/lfw/

Please note that you do not necessarily have to follow the skeleton code perfectly. We encourage you to include your own additional methods and functions. However, you are not allowed to use any `scikit-learn` classes or functions other than those already imported in the skeleton code.

We will explore clustering algorithms in detail by applying them to a toy dataset. In particular, we will investigate k-means and k-medoids (a slight variation on k-means).

(a) **(5 pts)** In k-means, we attempt to find k cluster centers $\boldsymbol{\mu}_j \in \mathbb{R}^d$, $j \in \{1, \ldots, k\}$ and n cluster assignments $c^{(i)} \in \{1, \ldots, k\}$, $i \in \{1, \ldots, n\}$, such that the total distance between each data point and the nearest cluster center is minimized. In other words, we attempt to find $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$ and $c^{(1)}, \ldots, c^{(n)}$ that minimizes

$$J(\boldsymbol{c}, \boldsymbol{\mu}) = \sum_{i=1}^{n} ||\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_{c^{(i)}}||^2.$$

To do so, we iterate between assigning $\boldsymbol{x}^{(i)}$ to the nearest cluster center $c^{(i)}$ and updating each cluster center $\boldsymbol{\mu}_j$ to the average of all points assigned to the j^{th} cluster.

Instead of holding the number of clusters k fixed, one can think of minimizing the objective function over $\boldsymbol{\mu}$, $\boldsymbol{c}$, and k. Show that this is a bad idea. Specifically, what is the minimum possible value of $J(\boldsymbol{c}, \boldsymbol{\mu}, k)$? What values of $\boldsymbol{c}$, $\boldsymbol{\mu}$, and k result in this value? **Solution:** The minimum objective value is 0. It is achieved when we have n clusters such that $c_i = i$ and $\boldsymbol{\mu}_j = \boldsymbol{x}^{(i)}$.

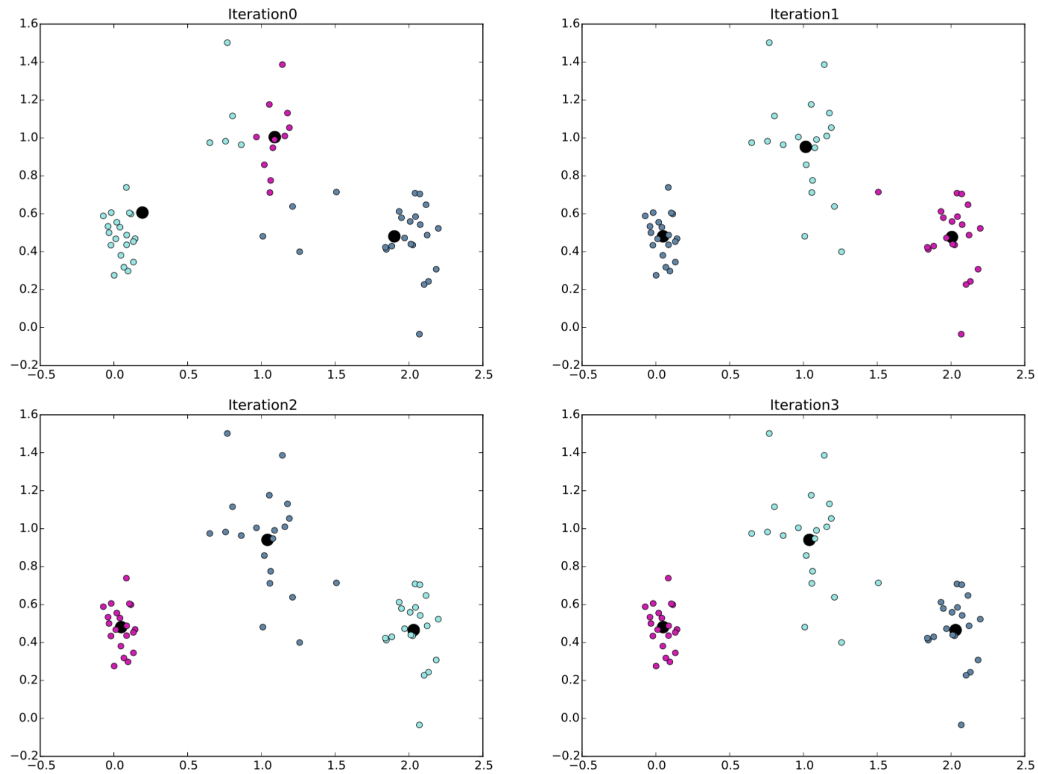
(b) **(10 pts)** To implement our clustering algorithms, we will use Python classes to help us define three abstract data types: `Point`, `Cluster`, and `ClusterSet` (available in `cluster.py`). Read through the documentation for these classes. (You will be using these classes later, so make sure you know what functionality each class provides!) Some of the class methods are already implemented, and other methods are described in comments. Implement all of the methods marked `TODO` in the `Cluster` and `ClusterSet` classes.

(c) **(20 pts)** Next, implement `random_init(...)` and `kMeans(...)` based on the provided specifications.

(d) **(5 pts)** Now test the performance of k-means on a toy dataset.

Use `generate_points_2d(...)` to generate three clusters each containing 20 points. (You can modify `generate_points_2d(...)` to test different inputs while debugging your code, but be sure to return to the initial implementation before creating any plots for submission.) You can plot the clusters for each iteration using the `plot_clusters(...)` function.

In your writeup, include plots for the k-means cluster assignments and corresponding cluster "centers" for each iteration when using random initialization. **Solution:**



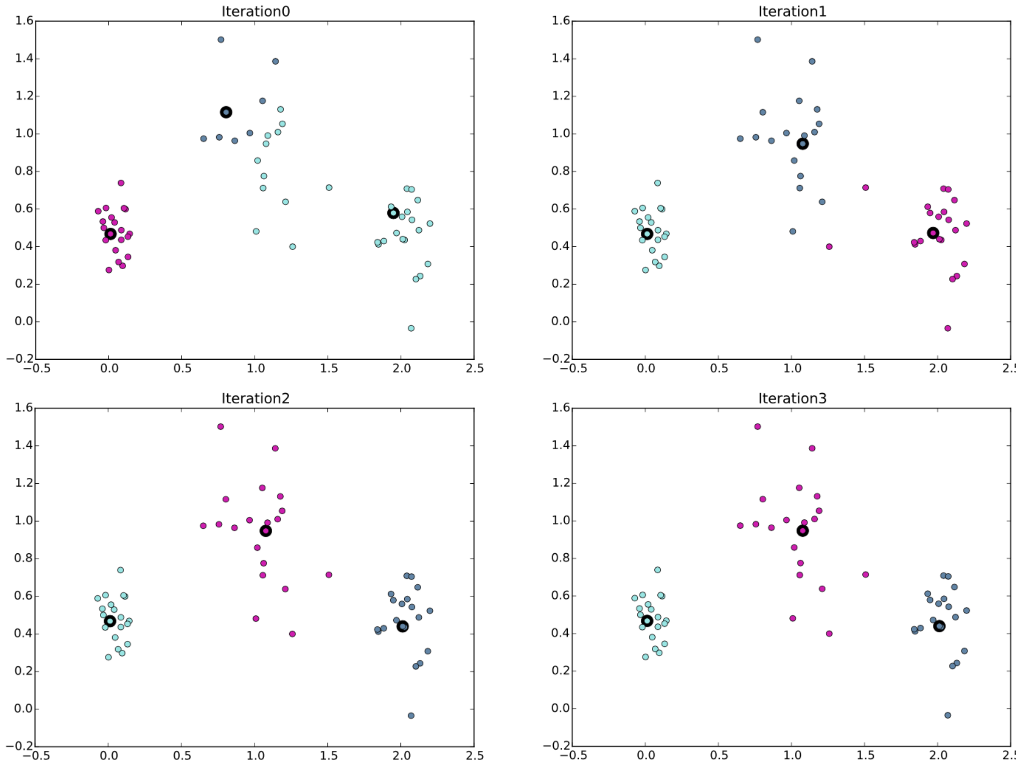
Note that you may have different results based on the randomly generated toy dataset and different random initialization.

(e) **(10 pts)** Implement `kMedoids(...)` based on the provided specification.

Hint: Since k-means and k-medoids are so similar, you may find it useful to refactor your code to use a helper function `kAverages(points, k, average, init='random', plot=True)`, where `average` is a method that determines how to calculate the average of points in a cluster (so it can take on values `ClusterSet.centroids` or `ClusterSet.medoids`).[2]

As before, include plots for k-medoids clustering for each iteration when using random initialization. **Solution:**

[2]In Python, if you have a function stored to the variable `func`, you can apply it to parameters `arg` by callling `func(arg)`. This works even if `func` is a class method and `arg` is an object that is an instance of the class.



(f) **(5 pts)** Finally, we will explore the effect of initialization. Implement `cheat_init(...)`. Now compare clustering by initializing using `cheat_init(...)`. Include plots for k-means and k-medoids for each iteration. **Solution:**

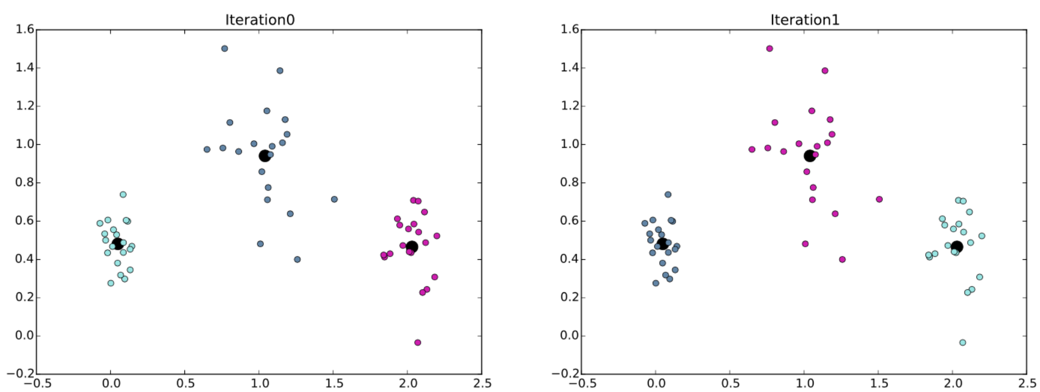


Figure 1: k-means with cheat initialization

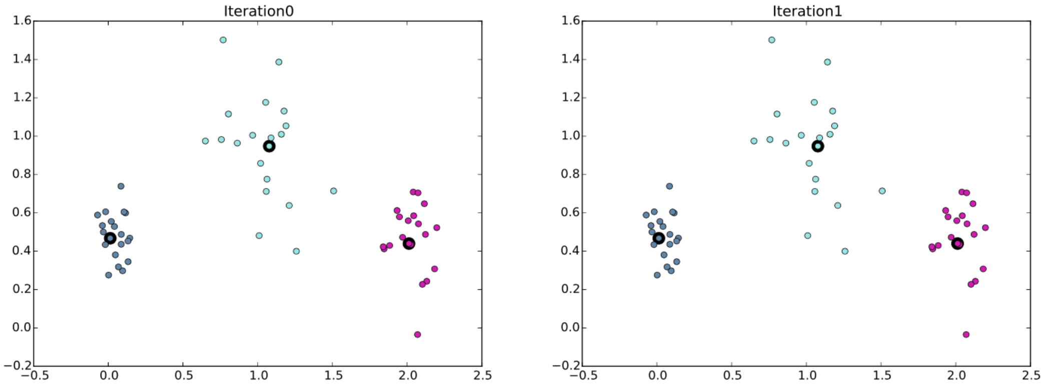


Figure 2: k-medoids with cheat initialization