10-701 Machine Learning, Fall 2012: Homework 3

Due Wednesday 11/14 at the beginning of class.

Instructions (important): Remember to submit your solution to **each problem separately** in **different piles** on the table at the front of the classroom at the beginning of class on the day the homework is due. At the *top* of the first page of each solution you hand in, clearly write the class number (10701), the **number of the problem** (i.e. "Problem 1", "Problem 2", etc.), your **first and last name** (assuming you have both), and your Andrew ID.

1 Clustering [35 points, Martin]

1.1 [25 points] K-means

Note: Remember to print and submit all your code along with the rest of your answers.

In K-means clustering, we are given points $x_1, ..., x_n \in \mathbb{R}^d$ and an integer $K > 1$, and our goal is to minimize the within-cluster sum of squares (also known as the k-means objective)

$$J(C, L) = \sum_{i=1}^{n} \|x_i - C_{\ell_i}\|^2$$

where $C = (C_1, ..., C_K)$ are the cluster centers ($C_j \in \mathbb{R}^d$), and $L = (\ell_1, ..., \ell_n)$ are the cluster assignments ($\ell_i \in \{1, ..., K\}$).

Finding the exact minimum of this function is computationally difficult. The most common algorithm for finding an approximate solution is Lloyd's algorithm, which takes as input the set of points and some initial cluster centers C, and proceeds as follows:

i. Keeping C fixed, find cluster assignments L to minimize $J(C, L)$. This step only involves finding nearest neighbors. Ties can be broken using arbitrary (but consistent) rules.

ii. Keeping L fixed, find C to minimize $J(C, L)$. This is a simple step that only involves averaging points within a cluster.

iii. If any of the values in L changed from the previous iteration (or if this was the first iteration), repeat from step i.

iv. Return C and L.

The initial cluster centers C given as input to the algorithm are often picked randomly from $x_1, ..., x_n$. In practice, we often repeat multiple runs of Lloyd's algorithm with different initializations, and pick the best resulting clustering in terms of the k-means objective. You're about to see why.

(a) **[3 points]** Briefly explain why Lloyd's algorithm is always guaranteed to converge (i.e. stop) in a finite number of steps.

(b) **[5 points]** Implement Lloyd's algorithm. Run it until convergence 200 times, each time initializing using K cluster centers picked at random from the set $\{x_1, ..., x_n\}$, with $K = 5$ clusters, on the 500 two dimensional data points in `http://www.cs.cmu.edu/~epxing/Class/10701/HW/hw3-cluster.csv`. Plot in a single figure the original data (in gray), and all 200×5 cluster centers (in black) given by each run of Lloyd's algorithm. You can play around with the plotting options such as point sizes so that the cluster centers are clearly visible. Also compute the minimum, mean, and standard deviation of the within-cluster sums of squares for the clusterings given by each of the 200 runs.

(c) **[4 points]** Kmeans++ is an initialization algorithm for K-means proposed by David Arthur and Sergei Vassilvitskii in 2007:

i. Pick the first cluster center C_1 uniformly at random from the data $x_1, ..., x_n$. In other words, we first pick an index i uniformly at random from $\{1, ..., n\}$, then set $C_1 = x_i$.

ii. For $j = 2, ..., K$:

- For each data point, compute its distance D_i to the nearest cluster center picked in a previous iteration:
$$D_i = \min_{j'=1,...,j-1} \|x_i - C_{j'}\|.$$
- Pick the cluster center C_j at random from $x_1, ..., x_n$ with probabilities proportional to $D_1^2, ..., D_n^2$. Precisely, we pick an index i at random from $\{1, ..., n\}$ with probabilities equal to $D_1^2/(\sum_{i'=1}^n D_{i'}^2), ..., D_n^2/(\sum_{i'=1}^n D_{i'}^2)$, and set $C_j = x_i$.

iii. Return C as the initial cluster assignments for Lloyd's algorithm.

Replicate the figure and calculations in part (b) using Kmeans++ as the initialization algorithm, instead of picking C uniformly at random.

Hopefully your results make it clear how sensitive Lloyd's algorithm is to initializations, even in such a simple, two dimensional data set!

Picking the number of clusters K is a difficult problem. Now we will see one of the most common heuristics for choosing K in action.

(d) **[3 points]** Explain how the exact minimum of the k-means objective behaves on any data set as we increase K from 1 to n.

A common way to pick K is as follows. For each value of K in some range (e.g. $K = 1, ..., n$, or some subset), we find an approximate minimum of the k-means objective using our favorite algorithm (e.g. multiple runs of randomly initialized Lloyd's algorithm). Then we plot the resulting values of the k-means objective against the values of K. Often, if our data set is such that there exists a natural value for K, we see a "knee" in this plot, i.e. a value for K where the rate at which the

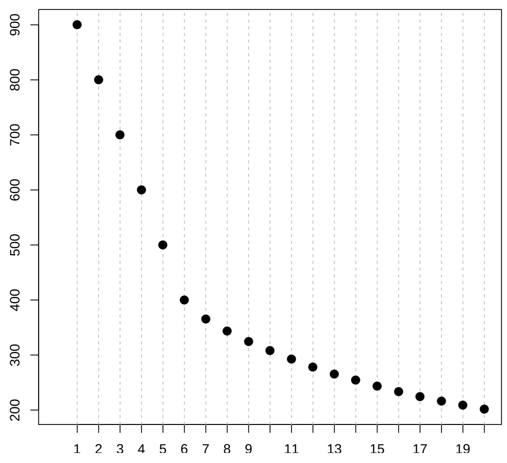


Figure 1: Picking the number of clusters (a toy example).

within-cluster sum of squares is decreasing sharply reduces. This suggests we should use the value for K where this knee occurs. In the toy example in Figure 1, this value would be $K = 6$.

(e) [**3 points**] Produce a plot similar to the one in Figure 1 for $K = 1, ..., 15$ using the data set in (b), and show where the "knee" is. For each value of K, run k-means with at least 200 initializations and pick the best resulting clustering (in terms of the objective) to ensure you get close to the global minimum.

(f) [**3 points**] Repeat part (e) with the data set in `http://www.cs.cmu.edu/~epxing/Class/10701/HW/hw3-cluster2.csv`. Find 2 knees in the resulting plot (you may need to plot the square root of the within-cluster sum of squares instead, in order to make the second knee obvious). Explain why we get 2 knees for this data set (consider plotting the data to see what's going on).

We conclude our exploration of k-means clustering with the critical importance of properly scaling the dimensions of your data.

(g) [**2 points**] Load the data in `http://www.cs.cmu.edu/~epxing/Class/10701/HW/hw3-cluster3.csv`. Perform k-means clustering on this data with $K = 2$ with 500 initializations. Plot the original data (in gray), and overplot the 2 cluster centers (in black).

(h) [**2 points**] Normalize the features in this data set, i.e. first center the data to be mean 0 in every dimension, then rescale each dimension to have unit variance. Repeat part (g) with this modified data.

As you can see, the results are radically different. You should not take this to mean that data should *always* be normalized. In some problems, the relative values of the dimensions are meaningful and should be preserved (e.g. the coordinates of earthquake epicenters in a region). But in others, the dimensions are on entirely different scales (e.g. age in years v.s. income in thousands of dollars). Proper pre-processing of data for clustering is often part of the art of machine learning.

1.2 Hierarchical clustering [10 points]

Agglomerative hierarchical clustering is a family of hierarchical clustering algorithms that, equipped with a notion of distance between clusters, form a binary tree with leaves for each original data point as follows:

i. Initialize by placing each data point in its own cluster (i.e. singleton trees).

ii. Find the two closest clusters, join them in a single cluster (by creating a new node and making it the parent of the roots of those two clusters).

iii. If there are more than one clusters (trees) left, repeat from step i.

iv. Return the final tree.

Some of the most common metrics of distance between two clusters $\{x_1, ..., x_m\}$ and $\{y_1, ..., y_p\}$ are:

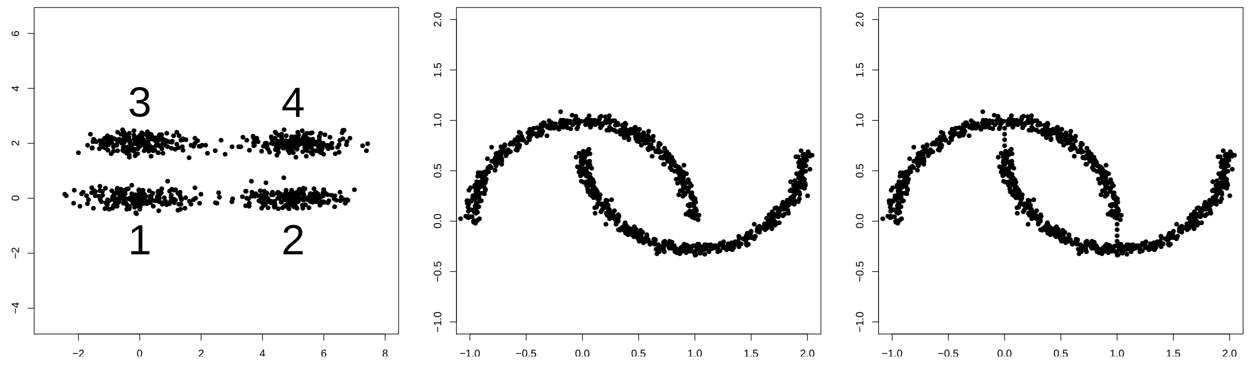
- *Single linkage*: Distance between clusters is the *minimum* distance between any pair of points from the two clusters, i.e.
$$\min_{\substack{i=1,...,m\\ j=1,...,p}} \|x_i - y_j\|;$$
- *Complete linkage*: Distance between clusters is the *maximum* distance between any pair of points from the two clusters, i.e.
$$\max_{\substack{i=1,...,m\\ j=1,...,p}} \|x_i - y_j\|;$$
- *Average linkage*: Distance between clusters is the *average* distance between all pair of points from the two clusters, i.e.
$$\frac{1}{m \cdot p} \sum_{i=1}^{m} \sum_{j=1}^{p} \|x_i - y_j\|.$$

Also, given a clustering tree, we can define a partitioning of the data into K clusters by "cutting" the tree some number of levels below the root. For example, if $K = 2$ we could define two clusters based on the left and right subtrees of the root of the clustering tree. If $K = 4$, we could use the subtrees of the children of the root, etc. (Note that if K is not a power of 2 we would need to come up with some way of deciding which subtree gets preference.)

(a) [**3 points**] Using this procedure for turning a hierarchical clustering into a partition, which of the three cluster similarity metrics described above would most likely result in clusters most similar to those given by k-means? (Assume K is a power of 2).

(b) [**4 points**] Consider the data in Figure 2a. What would be the result if we extracted $K = 2$ clusters from the tree given by hierarchical clustering on this data set using single linkage? (Describe your answer in terms of the labels $1 - 4$ given to the four "clumps" in the data.) Do the same for complete and average linkage.

(c) [**3 points**] Which of those three distance metrics (if any) would successfully separate the two "moons" in Figure 2b? What about Figure 2c? Briefly explain your answer.



(a) (b) (c)

Figure 2

2 Bayesian Network [25 points, Avi]

This problem will concern the Bayesian network in Figure 3.

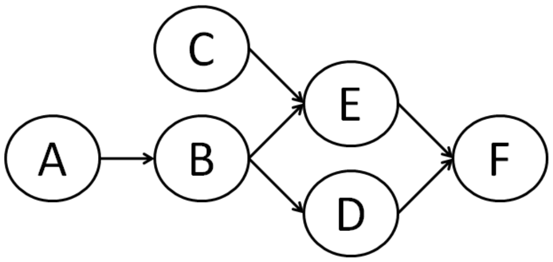


Figure 3

2.1 [3 points] Joint Probability

Write down the factorization of the joint probability distribution over A, B, C, D, E, F which corresponds to this graph.

2.2 [5 points] conditional independence

For every pair of nodes in the graph say whether each of them are independent of each other or not. Also test conditional independence for each pair except F when F is observed

2.3 [12 points] Inference

For this section we suppose that all the variables are binary, taking on the values $0, 1$. The conditional probability distributions on the graph have the following form:

- Nodes with a single parent take the value of their parent with probability $\frac{3}{4}$ otherwise they take the other value.

- Nodes with two parents take the value of the first parent with probability $\frac{1}{2}$ otherwise they take the value of the second parent.
- $P(A = 1) = p$, $P(C = 1) = q$.

All the assumptions made in this section holds for all the three questions below

1. [**2 points**] **CPT Tricks I**. If some node X has a single parent Y, and $P(Y = 1) = a$, what is a simple expression for $P(X = 1)$? Please assume that there is no child of X to worry about.
2. [**3 points**] **CPT Tricks II**. If some node X has a two independent parents Y, Z, and $P(Y = 1) = a,\ P(Z = 1) = b$, what is a simple expression for $P(X = 1)$? Please assume that there is no child of X to worry about.
3. [**7 points**] **Forwards Inference**. What is $P(F = 1)$ in the above graph?

2.4 [5 points] Conditional Inference

If $B = b, F = f$ are observed, what is the conditional probability that $E = 1$? For this question please leave your answer in terms of probability distributions e.g., $P(B = b|A = a)$ etc., but only those which could be computed directly from the local probabilities in the definition of the Bayes net.

3 Expectation Maximization [20 points, Derry]

In this question, you are going to derive the Expectation and Maximization equations of the EM algorithm for optimizing the latent variables involved in generating a text document.

Consider each word as a random variable w that can take values $1, ..., V$ from the vocabulary of words. Treat each w as a vector of V components such that $w(i) = 1$ if the w takes the value of the i^{th} word in the vocabulary. Hence, $\sum_i^V w(i) = 1$. The words are generated from a mixture of M discrete topics:

$$p(w) = \sum_{m=1}^{M} \pi_m p(w|\mu_m)$$

and

$$p(w|\mu_m) = \prod_{i=1}^{V} \mu_m(i)^{w(i)}$$

where π_m denotes the prior for the latent topic variable $t = m$ and $\mu_m(i) = p(w(i) = 1|t = m)$, thus $\sum_{i=1}^{V} \mu_m(i) = 1$.

Given a document containing words w_j, $j = 1, ..., N$, where N is the length of the document, **derive** the expectation and maximization step equations for the EM algorithm to optimize $\theta = \{\pi_m,\ \mu_m(i)\}$.

Note: Show all the steps in your derivation.

Hints:

- In the expectation step [**5 points**] , for each word w_j, compute $F_j(t_j) = p(t_j|w_j;\theta)$, the probability that w_j belongs to each of the M topic.
- In the maximization step [**15 points**], compute θ which is the set of parameters of this mixture model that maximizes the log likelihood of the data

  $$l(w;\theta) = log \prod_{j=1}^{N} p(w_j;\theta)$$

  Summing over the latent topic variable:

  $$l(w;\theta) = \sum_{j=1}^{N} log \sum_{t_j} p(w_j, t_j;\theta)$$

  $$l(w;\theta) = \sum_{j=1}^{N} log \sum_{t_j} F_j(t_j) \frac{p(w_j, t_j;\theta)}{F_j(t_j)}$$

  Using Jensen's inequality:

  $$l(w;\theta) \geq \sum_{j=1}^{N} \sum_{t_j} F_j(t_j)\ log\ \frac{p(w_j, t_j;\theta)}{F_j(t_j)} = \sum_{j=1}^{N} \sum_{t_j} F_j(t_j)\ log\ p(w_j;\theta) = log\ p(w_j;\theta) = l(w;\theta)$$

  Hence compute θ as:

  $$\theta := argmax_\theta\ \sum_{j=1}^{N} \sum_{t_j} F_j(t_j)\ log\ \frac{p(w_j, t_j;\theta)}{F_j(t_j)}$$

4 Hidden Markov Model [20 points, Zeyu]

As mentioned in class, we are going to derive the formula for learning HMM. Suppose x_t is the random observation at time t and $y_t^k = 1$ is the event where the state of time t is k. In class we derived the forward probability $\alpha_t^k = P(x_1...X_{t-1}, x_t, y_t^k = 1)$ as an iterative function:

$$\alpha_t^k = P(x_t|y_t^k = 1) \sum_i \alpha_{t-1}^i a_{i,k} \tag{1}$$

where $a_{i,k}$ is the transitional probability from state i to k. We also defined the backward probability as the future observations given the current state:

$$\beta_t^k = P(x_{t+1}...x_T|y_t^k = 1) \tag{2}$$

The following questions will guide you through the derivation of EM for HMM. Please keep your notation consistent with the following convention:

- Use uppercase letters to present random variables (e.g. Y_t); and use lowercase ones for particular values(e.g. x_t).
- Use $y_t^k = 1$ or just y_t^k to mean the event $Y_t = (Y_t^1, ..., Y_t^{k-1}, Y_t^k, Y_t^{k+1}, ..., Y_t^K) = (0, ..., 0, 1, 0, ..., 0)$ where Y_t is a vector that follows multinomial distribution and Y_t^k is the k-th element.
- Transition probability: $a_{i,j} = P(y_t^j = 1|y_{t-1}^i = 1)$
- Emission Probability: $b_{i,k} = P(x_t^k = 1|y_t^i = 1)$
- Assume there are K states and N types of observation.

For the following questions, show clear steps of your work.

1. [**4 points**] Prove $\beta_t^k = \sum_i a_{k,i} P(x_{t+1}|y_{t+1}^i = 1)\beta_{t+1}^i$.
2. [**2 points**] Now we are going to derive the EM algorithm for HMM. Before doing any work, answer the question: What are the parameters (i.e. Θ) we want to estimate? And what are the latent variables?
3. [**4 points**] Now Write the complete likelihood function $P(X, Y|\Theta)$. **Notes:** You may end up with some function that contains $P(Y_t = y_t|Y_{t-1} = y_{t-1})$ and $P(X_t = x_t|Y_t = y_t)$. However it is not useful form because $Y_t|Y_{t-1}$ is a huge random matrix containing $K \times K$ elements. So we need to write $P(Y_t|Y_{t-1})$ as indicator function. Suppose $a_{i,j}$ is the transition probability. Then $p(Y_t = i|Y_{t-1} = j)$ can be expressed as $\prod_{i=1}^K \prod_{j=1}^K a_{j,i}^{y_t^i y_{t-1}^j}$ where y_t^i is the i-th element of the vector y_t. The term $a_{j,i}^{y_t^i y_{t-1}^j}$ is zero unless y_t^i and y_{t-1}^j are one. Similarly, we can define emission probability $b_{j,k}$ for $p(Y_t = k|Y_{t-1} = j)$ and use the indicator functions to present $P(X_t = x_t|Y_t = y_t)$.
4. [**4 points**] Write the formula for expected log likelihood, namely $E[l(\Theta; X, Y)]$. You can also use "$< >$" to indicate expectation as shown in class. To check if you are on the right track, make sure you find the following terms in your derivation. (Note: the correct answer looks a little like the one on the slides but not exactly the same. So do the derivation yourself and do not copy and paste)
   - $\gamma_t^i \overset{def}{=} E[y_t^i] = p(y_t^i = 1|x_n)$
   - $\xi_t^{ij} \overset{def}{=} E[y_{t-1}^i y_t^j] = p(y_{t-1}^i = 1, y_t^j = 1|x_n)$

   [**0 points, do not hand it in**] Now we are one step close to getting the formula $Q(\Theta, \Theta^{old})$ for this EM problem. If you are interested you can follow the steps below to obtain the final E and M steps for learning a HMM. You may refer to the slides to check your answer.

   E step: calculate γ_t^i and ξ_t^{ij} based on Θ^{old}

(a) Recall the definition of forward and backward probability. Write γ_t^i as a function of α_t^k and β_t^k. (before doing this you may need to consider how to compute $p(X)$ which is a function of α_t^k)

(b) Show that $P(X|y_n, y_{n-1}) = P(x_1, ..., x_{n-1}|y_{n-1})p(x_n|y_n)p(x_{n+1}, ..., x_N|y_n)$

(c) Use Bayes theorem, and show $\xi(y_{n-1}, y_n) = \alpha(y_{n-1})P(x_n|y_n)p(y_n|y_{n-1})\beta(y_n)/\sum_k \alpha_N^k$

M step: show the update rules for $\pi_i, a_{i,j}$ and $b_{i,k}$ as a function of ξ_t^{ij}, γ_t^i, and x_t^k

5. **[6 points]** Now we are going to adjust the HMM model a little bit to make it more powerful for some particular circumstance. Suppose at each time point t, we know an additional variable Z_t, and for each different value $Z_t = z$ the transition probability $a_{i,j}$ is different (we can call it $a_{i,j|z}$). This model is very useful in practice. For example in recognizing hand gestures, Z_t defines a group of gestures with similar properties. The value of each z_t is computed using other algorithms (for example K-means). Having this additional variable decreases the complexity of hidden layers and thus increases the convergence rate when training this model. However, this model cannot over-perform HMM when the transitional matrices given $Z_t = z$ are similar for all values of z.

   (a) **[2 points]** Based on the information above, draw the Bayesian Networks for this new HMM model. (Assume the observation X_t does not depend on Z_t)

   (b) **[2 points]** What is the complete likelihood, $P(X, Y, Z)$ of this new model?

   (c) **[2 points]** Suppose the variable Z_i takes n different values and there are m states in the hidden layer. How many states does an ordinary HMM need to have to be equivalent to this new model? Provide your reasoning.