CS 498 AML HW7

Yuxi Gu(yuxigu2)
Songwei Feng(songwei3)
Xuanyi Zhu(xzhu42)

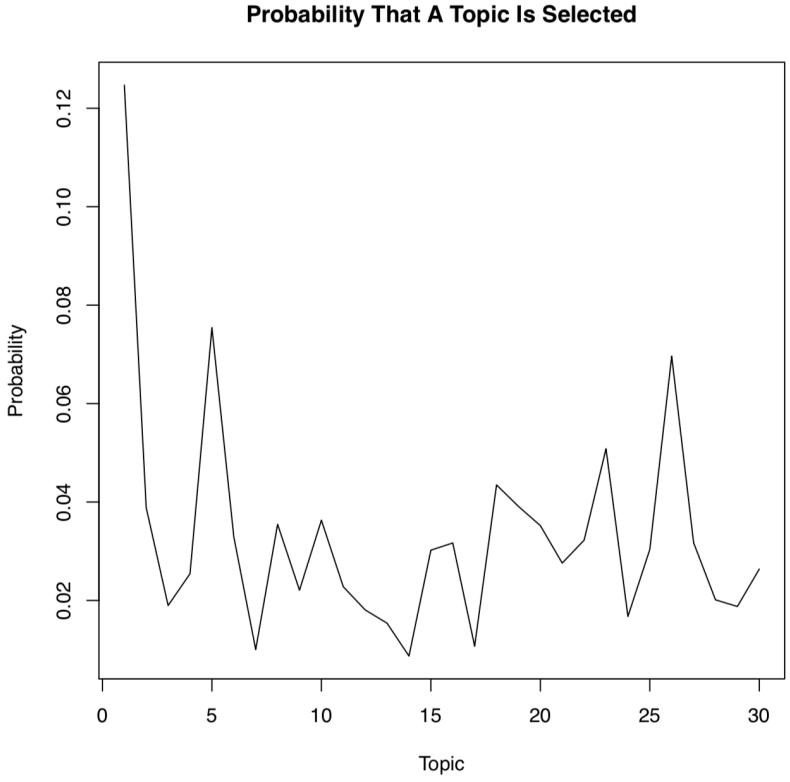
EM Topic Model

We used R as the programming language for this problem.

We used NIPS dataset from UCI machine learning repository for this problem. We first constructed a matrix from the dataset in docword.nips.txt. The row index for the matrix indicates the docID and the column index indicates the wordID. And each cell is the word count of the word (represented by wordID) in the document (represented by docID). Then we need to cluster 1500 documents into 30 different topics. And we used random distribution function, which was runif function in R, to get the initial value for each row of P_j_k, where each cell of P_j_k represents the probability of showing word k in topic j.

After data preparation, we did the EM steps stated in the textbook. And the result we got is showed as below.

A graph showing, for each topic, the probability with which the topic is selected.



A table showing, for each topic, the 10 words with the highest probability for that topic.
(For this question, we write the result to a csv file and below is a screenshot of the data in this csv file.)

	A	B	C	D	E	F	G	H	I	J	K
1		V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
2	1	point	distance	data	method	task	space	set	function	learning	image
3	2	model	data	point	function	distribution	algorithm	learning	method	network	output
4	3	network	neuron	cell	model	input	neural	system	circuit	unit	output
5	4	network	neural	pattern	neuron	order	input	learning	result	system	function
6	5	cell	model	input	information	object	network	rate	term	iiii	neural
7	6	network	input	algorithm	learning	neural	routing	noise	output	set	information
8	7	network	unit	input	weight	learning	pattern	output	hidden	function	set
9	8	network	input	data	unit	function	system	layer	output	neural	training
10	9	network	model	unit	system	direction	noise	set	function	error	channel
11	10	weight	charge	system	transfer	input	algorithm	output	gain	transistor	vor
12	11	network	function	neural	weight	input	algorithm	learning	set	data	output
13	12	learning	network	model	set	neural	system	algorithm	input	training	function
14	13	learning	algorithm	model	data	function	problem	set	unit	network	error
15	14	learning	error	task	network	input	field	training	noise	output	chip
16	15	model	neuron	data	parameter	circuit	current	likelihood	learning	song	response
17	16	learning	function	algorithm	model	method	vector	problem	set	parameter	result
18	17	network	learning	function	error	training	set	input	neural	result	weight
19	18	network	input	model	neural	spike	system	signal	neuron	function	noise
20	19	model	network	algorithm	data	mean	pattern	set	graph	field	nodes
21	20	data	network	algorithm	set	function	model	problem	input	training	weight
22	21	model	network	unit	learning	input	error	training	output	system	neural
23	22	object	color	image	algorithm	features	feature	network	developmen	layer	system
24	23	input	vector	output	pattern	model	network	algorithm	error	data	mlp
25	24	network	system	training	neural	speech	input	data	model	output	performance
26	25	network	object	training	set	neural	input	system	learning	recognition	hint
27	26	model	learning	mean	approximatic	field	input	decision	problem	machine	order
28	27	network	neural	unit	training	model	system	set	input	learning	recognition
29	28	network	model	function	system	data	learning	neural	algorithm	set	input
30	29	learning	model	input	network	rate	set	weight	neural	function	result
31	30	network	model	function	input	learning	neural	set	data	weight	algorithm

Problem 2

For this problem, we used Python.

We used Python with spicy.misc to read the image. We implemented color image segmentation using EM and segmented three images to **10**, **20** and **50** segments separately, then segmented the sunset image to **20** segments using **5** different start points.

In the segment process, first, we used kmeans to initialize centers and evenly initialize pi_j. Then we compute the wi_j based on the following formula in E step:

$$p(\delta_{ij} = 1|\theta^{(n)}, \mathbf{x}) = \frac{\left[\exp\left(-\frac{1}{2}(\mathbf{x}_i - \mu_j)^T(\mathbf{x}_i - \mu_j)\right)\right]\pi_j}{\sum_k \left[\exp\left(-\frac{1}{2}(\mathbf{x}_i - \mu_k)^T(\mathbf{x}_i - \mu_k)\right)\right]\pi_k} = w_{ij}.$$

Then we updated the μ and π using the formulas below in M step until met the stop criteria which means the distance between the recent π and prior π is less than 0.0001:

$$\mu_j^{(n+1)} = \frac{\sum_i x_i w_{ij}}{\sum_i w_{ij}} \qquad \pi_j^{(n+1)} = \frac{\sum_i w_{ij}}{N}.$$

To output the image, we mapped each pixel to the cluster center with the highest value of the posterior probability and reconstruct the image. At last, we got the following output:

Segment = 10

Segment = 20

Segment = 50

Segment = 10

Segment = 20

Segment = 50

Segment = 10

Segment = 20

Segment = 50

Then since in scipy.cluster.vq.kmeans, the initial k centroids are chosen by randomly selecting observations from the observation matrix. We run the segment process **5** times, in this way, we segment the sunset image to 20 segments using five different start points generated by kmeans, and display the result for each case:

Test1

Test2

Test3

Test4

Test5

From the five images we can see with different starting points, the output images after 20 image segment using EM are almost the same, which is reasonable since after enough iterations, the weights and parameters will be fit to the image and most pixels in image will be clustered properly to the right segments.