Homework 5 (optional)
Latent Dirichlet Allocation

CMU 10-701: Machine Learning (Fall 2014)
https://piazza.com/cmu/fall2014/1070115781/home
OUT: Dec 3, 2014
DUE: Dec 15th, 11:59 PM

START HERE: Instructions

- **Late days:** The homework is due Monday Dec 15th, at 11:59PM. Late days are **NOT** eligible to be used on this homework.
- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get inspiration (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators and resources fully and completely (e.g., "Jane explained to me what is asked in Question 3.4" or "I found an explanation of conditional independence on page 17 of Mitchell's textbook"). Second, write up your solution independently: close the book and all of your notes, and send collaborators out of the room, so that the solution comes directly from you and you alone.
- **Programming:**
  - **NOTE: THIS HOMEWORK HAS NO WRITTEN PORTION.**
  - **Octave:** You must write your code in Octave. Octave is a free scientific programming language, with syntax identical to that of MATLAB. Installation instructions can be found on the Octave website. (You can develop your code in MATLAB if you prefer, but you *must* test it in Octave before submitting, or it may fail in the autograder.)
  - **Autograding:** All programming problems are autograded using the CMU Autolab system. The code which you write will be executed remotely against a suite of tests, and the results used to automatically assign you a grade. To make sure your code executes correctly on our servers, you should avoid using libraries which are not present in the *basic* Octave install.
- **Submitting your work:** All answers will be submitted electronically through the submission website: https://autolab.cs.cmu.edu/10701-f14.
  - Start by downloading the submission template. The template consists of a single sub-directory for the programming part of the assignment. *Do not modify the structure of these directories or rename these files.*
  - **Putting it all together:** Once you have provided each of the function stubs, compress the top level directory *as a tar file* and submit to Autolab online (URL above). You may submit your answers as many times as you like. You will receive instant feedback on your autograded problems.

Problem 1: Latent Dirichlet Allocation and EM [Abu - 40 pts]

Background: *Latent Dirichlet allocation* (LDA) is a probabilistic model of discrete data. It is the prototypical example of a *topic model*, which characterize documents based on the words contained within them. LDA makes use of the *Dirichlet distribution*, which is a continuous distribution over vectors that sum to 1. The Dirichlet distribution has one parameter vector $\alpha \in \mathbb{R}^n$, where each element is strictly positive: $\alpha_i > 0$. We write:

$$x \sim \text{Dir}(\alpha) \text{ where } x \in \mathbb{R}^n \text{ and } \sum_{i=1}^{n} x_i = 1,$$

$$p(x) = \frac{1}{B(\alpha)} \prod_{i=1}^{n} x_i^{\alpha_i - 1},$$

where the normalizing constant $B(\alpha)$ is the multinomial Beta function

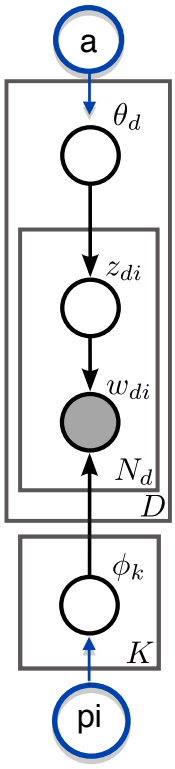
$$B(\alpha) = \frac{\prod_{i=1}^{n} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{n} \alpha_i)},$$

and $\Gamma(\cdot)$ is the gamma function, which can be understood as a continuous generalization of the factorial. When the parameter α is a vector of all ones, the Dirichlet distribution is uniform.

The Dirichlet distribution is the multivariate generalization of the Beta distribution. The following table outlines the relationships between the Dirichlet and other well-known probability distributions:

two outcomes	Bernoulli	binomial	Beta
many outcomes	categorial	multinomial	Dirichlet

Model: We will present LDA as a model of words in a collection of documents, but it can easily be extended to other kinds of discrete data. Each topic can be understood as a distribution over words in a dictionary. For example, you could have a "sports" topic that puts high probability on words like "game" or "ball." There could be a "machine learning" topic with high probability on words like "data" and "train." In a collection of D documents, each document is assigned a distribution of topics (a Science journal article will likely have low weight for the topic "sports"). Every word in each document is assigned a topic. The word is then distributed according to that topic.



The full model is given below, where D is the number of documents, K is the number of topics, and W is the number of words in a dictionary:

$$\begin{aligned}
\phi_k &\sim \text{Dir}(\pi), \text{ for each topic } k = 1, \ldots, K, \\
\theta_d &\sim \text{Dir}(\alpha), \text{ for each document } d = 1, \ldots, D, \\
z_{di} &\sim \text{Cat}(\theta_d), \text{ for each word } i \text{ in document } d = 1, \ldots, D, \\
w_{di} &\sim \text{Cat}(\phi_{z_{di}}), \text{ for each word } i \text{ in document } d = 1, \ldots, D.
\end{aligned}$$

For every topic k, a distribution over words $\phi_k \in \mathbb{R}^W$ is drawn from a Dirichlet with parameter $\pi \in \mathbb{R}^W$. Then, for every document, a distribution over topics θ_d is drawn from a Dirichlet with parameter $\alpha \in \mathbb{R}^K$. Then for every word in every document, a topic z_{di} is selected by drawing from a categorial distribution with parameter θ_d. Finally, the words themselves w_{di} are selected by drawing from a categorical distribution with parameter $\phi_{z_{di}}$. If the number of documents D is 1, then LDA becomes equivalent to a categorical mixture model.

Inference: Given a collection of documents with observed words $\boldsymbol{w}$, we want to learn about the unknown variables: $\boldsymbol{\phi}$, $\boldsymbol{\theta}$, and $\boldsymbol{z}$. To do so, we will use expectation maximization (EM). See the lecture and recitation for more information (and an example of how EM is applied to a Gaussian mixture model). We will divide the unknown variables into two classes: $\{\boldsymbol{\phi}, \boldsymbol{\theta}\}$ and $\{\boldsymbol{z}\}$. In the E step, we will update the distribution of $\boldsymbol{z}$ given that $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ are fixed. In the M step, we will update our belief of $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, using the conditional distribution of $\boldsymbol{z}$.

a. **[6 pts]** Derive the conditional probability that j is the topic assigned to word i in document d. That is, derive $p\{z_{di} = j|\boldsymbol{w}, \boldsymbol{\phi}, \boldsymbol{\theta}\}$. Computing this term constitutes the E step.

Use your result to implement the function `compute_z_conditional.m`. This function takes as input $\boldsymbol{w}$, $\boldsymbol{\phi}$, $\boldsymbol{\theta}$, and d. It computes the conditional for document d, and returns a matrix of dimension $K \times N_d$, where N_d is the number of words in document d. The element at row j and column i of this matrix is the conditional probability $p\{z_{di} = j|\boldsymbol{w}, \boldsymbol{\phi}, \boldsymbol{\theta}\}$. We provide the function `make_data.m` to generate a simple dataset for testing.

b. **[14 pts]** Derive the following energy function: $q(\boldsymbol{\phi}, \boldsymbol{\theta}) = \mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{w}, \boldsymbol{\phi}, \boldsymbol{\theta})}[\log p(\boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{z}|\boldsymbol{w})]$, omitting any terms constant with respect to $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$. A good first step is to write out the log posterior probability: $\log p(\boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{z}|\boldsymbol{w})$.

Use your result to implement the function `energy.m`. This function takes as input $\boldsymbol{w}$, $\boldsymbol{\phi}$, $\boldsymbol{\theta}$, and the conditional distribution of $\boldsymbol{z}$ (as computed by the above E step, for example). It returns the scalar $q(\boldsymbol{\phi}, \boldsymbol{\theta})$. Note that this function is not directly used by the EM algorithm. This question is intended to evaluate the correctness of your derivation.

c. **[8 pts]** Derive the M step for $\boldsymbol{\theta}$. That is, find the closed-form expression for $\boldsymbol{\theta}$ that maximizes q. Recall that each θ_d must sum to 1, and so you could use Lagrange multipliers to solve this constrained optimization problem.

Use your result to implement the function `maximize_theta.m`. This function takes as input $\boldsymbol{w}$, α, d, and the conditional distribution of $\boldsymbol{z}$. It returns a vector of dimension $1 \times K$ that contains the value of θ_d that maximizes the energy function q.

d. **[8 pts]** Derive the M step for $\boldsymbol{\phi}$. That is, find the closed-form expression for $\boldsymbol{\phi}$ that maximizes q. Similarly, each ϕ_k must sum to 1.

Use your result to implement the function `maximize_phi.m`. This function takes as input $\boldsymbol{w}$, π, k, and the conditional distribution of $\boldsymbol{z}$. It returns a vector of dimension $1 \times W$ that contains the value of ϕ_k that maximizes the energy function q.

e. **[4 pts]** Put everything together and implement the function `em_lda.m`. This function takes as input $\boldsymbol{w}$, α, π, and a specified number of iterations to run. It runs the specified number of iterations of the E step followed by the M step, iteratively improving the estimates of $\boldsymbol{z}$, $\boldsymbol{\phi}$, and $\boldsymbol{\theta}$. See the function documentation in the template code for details on initialization and the expected structure of the output variables.