Project 1 – Implementing Switching Graphical Models

Advanced Natural Language Processing

Max Marks: 100

1. Introduction

We have learned how topic models (e.g., Latent Dirichlet Allocation, LDA [1]) work. In this project we will learn about a new graphical model which extends LDA. LDA only performs a weak semantic clustering of topical words. However, in most real-world documents, we find topics and intentions/sentiments jointly mentioned in them. Our goal is design a graphical model which can model mentions of two kinds of distributions (topics and intentions) from text. As modeling large scale document collections can be computationally expensive, for this project we will be simulating our results in the pixel space visualizing the state of the Markov Chain as it approaches its posterior distribution upon Gibbs sampling iterations.

Obtain the Java codebase (eclipse package) and reading materials for this project from links below:
http://www2.cs.uh.edu/~arjun/courses/GraphicalModelsInventory.zip
http://www2.cs.uh.edu/~arjun/courses/Topic models reading materials.zip

2. Jointly Modeling Topics and Intentions using Beta Switch Samplers

To jointly model topics and intentions in documents, let us first take a look how they appear in the real-world data. We choose the online debate/discussion domain which generates a large amount of Web traffic. Figure 1 shows mentions of topics and two intentions/sentiments (contention/disagreement, and concurrence/agreement) in debate discussions and online forums, where people often use topics in conjunction with sentiment/emotion intentions (e.g., contention/disagreement, and concurrence/agreement). To illustrate, consider the following excerpts of two actual debate posts where topics and intentions are color coded:

what a useless argument...cannabis...schizophrenia...**I don't see**...**doesn't prove**...marijuana...glaucoma...drug...**but the fact is**...tobacco...alcohol...**I don't think**...society...wine...legalising...ciggerette...legal...vice...weed...**you don't see**...heroin...**you don't know anything**...medical use...canadian government...US government...**i do not accept that argument**...**would you agree that**...deaths...affect on society...**which I argue**...**I disagree**...**I'm certain you'd find something wrong**...

I suppose it's a valid argument...philosophy...doxastic value...alethiological...evolution...faith based..creationism...doxology...***I agree with your***...scientists...national geographic channel...crop circles..religious-belief..dead..***of course***...god...jesus...christians...atheists...science...faith...believe in religion....***well put*** mirutsa...***you're right***...***of course***... historicity...christianity.

Figure 1: Topic/Intention color coded debate excerpts

i.e., all red/blue colored phrases/words indicate disagreement/agreement respectively. Further, we also have different topics ($drugs \rightarrow pink$, $society \rightarrow organge$, $christianity \rightarrow green$, etc.). Building on LDA, if we wish to model this kind of text data, we need a new graphical model. Essentially, we need to retain the topic word emission process of LDA but somehow need to "switch" between topics and intentions when generating the next word in the document. We also need to separate topics and intention models (i.e., topic distribution over words and intention distribution over words). Hence, we propose a new model, Joint Topic Intention (JTI) model.

2.1 Model Overview

The JTI model belongs to the family of generative models for text where words (unigrams)/phrases (n-grams) are viewed as random variables, a document is viewed as a bag of n-grams, and each n-gram (word/phrase) takes one value from a predefined vocabulary. In this work we refer both words (unigrams)/phrases (n-grams) by terms. We denote the entries in our vocabulary by $v = 1, \ldots, V$ where V is the number of unique terms in the vocabulary. A document (discussion post), d is represented as a vector of n-grams $\boldsymbol{w}_d$, with N_d entries. The entire corpus is comprised of $d = 1, \ldots, D$ documents. W is the set of all observed terms in the corpus with cardinality, $|W| = \sum_d N_d$. The JTI model is a hierarchical generative model which is motivated by the joint occurrence of intention types (disagreement, agreement) and topics in discussion posts. In general there could be a total of I intentions and T topics mentioned in our document corpus/collection. Our model is general and is designed to work with any value of I and T.

The observation in Figure 1 motivates the generative process of our model where documents are represented as random admixtures of *latent* topics and intentions. Each topic or intention is characterized by a distribution over terms (words/phrases). Assume we have $t = 1, \ldots, T$ topics and $i = 1, \ldots, I$ intentions in our corpus. Also, let ψ_d denote the distribution of topics and intentions in document d with $r_{d,j}$ denoting the binary indicator "switch" variable (topic or intention) for the j^{th} term of d, $w_{d,j}$. Let $z_{d,j}$ denote the appropriate topic or intention index to which $w_{d,j}$ belongs. We parameterize multinomials over topics using a stochastic matrix $\Theta^T_{D\times T}$ whose elements signify the probability that document d exhibits topic t. For simplicity of notation, we will drop the latter subscript (like t in this case) when convenient and use θ_d to stand for the d^{th} row of Θ^T. Similarly, we define multinomials over intentions using a stochastic matrix $\Theta^I_{D\times I}$. The multinomials over terms (words/phrases) associated with each topic are parameterized by a stochastic matrix $\Phi^T_{T\times V}$, whose elements denote the probability of generating the term v from topic t. Likewise, multinomials over terms associated with each intention are parameterized by a stochastic matrix $\Phi^I_{I\times V}$. We now formally define the generative process of JTI (Figure 2).

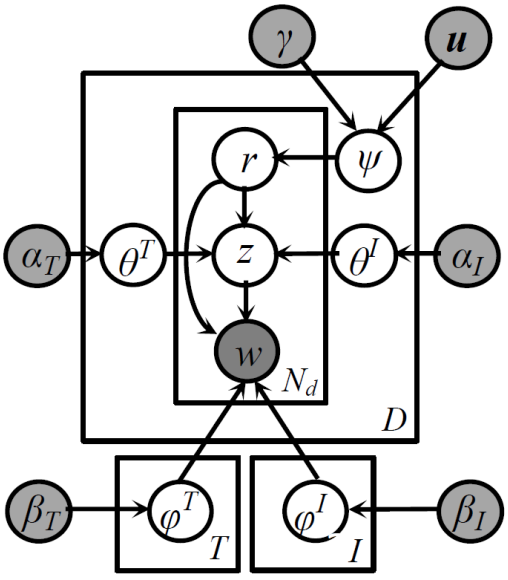

Figure 2: Plate notation of the Joint Topic and Intention Model

2.2 Generative process

1. For each intention i, draw $\varphi_i^I \sim Dir(\beta_I)$
2. For each topic, t, draw $\varphi_t^T \sim Dir(\beta_T)$
3. For each document post, $d \in \{1 \dots D\}$:
   i. Draw $\psi_d \sim Beta(\gamma \boldsymbol{u}) = Beta(\gamma_a, \gamma_b)$ // draw the switch prior
   ii. Draw $\theta_d^I \sim Dir(\alpha_I)$ // draw the document intention distribution
   iii. Draw $\theta_d^T \sim Dir(\alpha_T)$ // draw the document topic distribution
   iv. For each term $w_{d,j}; j \in \{1 \dots N_d\}$:
      a. Draw the topic/intention switch variable for this term, $r_{d,j} \sim Bernoulli(\psi_d)$
      b. if $(r_{d,j} = \hat{t})$ // the term at $w_{d,j}$ is a topical term
         Draw $z_{d,j} \sim Mult(\theta_d^T)$
      c. if $(r_{d,j} = \hat{\imath})$ // the term at $w_{d,j}$ is an intention term
         Draw $z_{d,j} \sim Mult(\theta_d^I)$
      d. Emit the corresponding topical/intention term from its corresponding (latent) topic/intention term distribution, $w_{d,j} \sim Mult(\varphi_{z_{d,j}}^{r_{d,j}})$

The graphical model corresponding to this process is shown in plate notation in Figure 2. Our goal in this project is to implement an approximate inference scheme, using MCMC collapsed Gibbs sampling for posterior inference.

3. Generating Synthetic Data

The class **IntentModelSyntheticData** generates a small document corpus of 1000 documents, each document containing 300 words. The entire vocabulary of the corpus is consisting only of 25 words which can be easily visualized as a 5 x 5 pixel matrix as follows (each pixel 0-24 is a word):

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

Figure 3: Pixel Matrix

Further, as we know form topic models [1], that topics are nothing but a distribution over words in the vocabulary, we can define various topics on this vocabulary using appropriate multinomial distributions. We can also make a distinction between two kinds of topics: (a) regular topics (b) intentions to simulate the documents in debate discussions and online forums. We can simulate the discussions data in a simplistic manner with just a 25 word vocabulary (i.e., there are no phrases, words can represent both topics and intentions). We can do this by generating about 10 topics and 2 intentions using the class **SyntheticData** and visualize the predefined topics and

intentions using the class **GibbsVisualizer**. We show the topics and intentions generated by **GibbsVisualizer** below:

Figure 3: Visualizing Topics/Intentions

However, it is important to note that documents generated using the class **SyntheticData** follow the following generate process (assuming a total of fixed $K = 12$ topics, 10 proper topics + 2 intentions):

For each $d \in \{1 \dots M = 100\}$

{

1. Draw the topic distribution for the document, d, $\theta_d \sim Dir(\alpha)$
2. Compute the weighted probability of a word appearing in document d, using $\theta_{d,t} \times \varphi_{t,w}$
3. This generates the document word distribution, $P(w|d) = \sum_t (\theta_{d,t} \times \varphi_{t,w})$.
4. Finally, emit actual words in this document, d by sampling words from the multinomial distribution over all words $\{P(w_i|d) \mid i = 0 \dots 24\}$.

}

These are clearly detailed in the methods `init_topics()` and `generateDocs()` in the class **SyntheticData**. You can also see the helper classes **DirichletSampler** and **SampleMultinomial** to understand "how exactly" it is done.

Going back to our **IntentModelSyntheticData**, we find that it further simplifies the problem by only considering 7 topics and 3 intentions. It employs a similar mechanism as in to generate the synthetic data, i.e., documents comprising of topics and intentions on a 25 word vocabulary. It is important to note that is specifically draws the (topic/intention) switch variable, r from a biased/asymmetric Beta distribution, $r \sim Beta(3.5, 1.0)$ which ensures that topic words appear in majority (to simulate the real-world distribution). This can be easily understood by following the details in the `generateDocs()` method in the class **IntentModelSyntheticData**. Further, the aggregate point estimates of r over all synthetic documents show that on average topic and intention words appear approximately 79% and 21% of the times respectively in documents.

4. Implementing the Markov Chain Monte Carlo (MCMC) Gibbs Sampler

Start with the template **IntentModelGibbsSampler**. This builds over the **LDAGibbsSampler** which was provided in the preparatory materials. The structure is very similar but adds more functionality to standard Latent Dirichlet Allocation. Use the following prior beliefs on the hyperparameters:

$$\alpha_T = \alpha = \frac{50}{T = 7} = 7.1428$$

$$\alpha_E = \alpha_i = \frac{50}{I = 3} = 16.667$$

$\psi \sim Beta(4,1) = Beta(\gamma = 5 \times \boldsymbol{u} = [\frac{4}{5}, \frac{1}{5}])$, where γ denotes the concentration parameter and $\boldsymbol{u}$ the base measure for Beta distribution.

Also note the representation used to jointly model topics and intents. To bootstrap the state of the Markov chain (i.e., assigning the latent variables $z_{d,j}$ to random topic/intention assingments), we use the following scheme. When $z_{d,j}$ takes on topic assignments (i.e., the corresponding $r_{d,j} = \hat{t}$) it is assigned values $z_{d,j} = 0, 1, ... T-1$. When $z_{d,j}$ takes on intentions (i.e., the corresponding $r_{d,j} = \hat{\imath}$), it is assigned values $z_{d,j} = -1, -2, ... -I$. This scheme allows us to capture both topic and intention assignments using exactly one variable. There are other splitting schemes possible too, but this is simple and does the job! Further, the topic/intention indicator switch variable $r_{d,j}$ for each word assignment, takes the value **true** for topics and **false** for intents. This is detailed in the method `initialState()`. **It is important to follow the same representation to ensure ease and correctness of implementation for the methods in Section 4.2**.

Recall that the machine learning goal is to mine the synthetic data (as documents) and estimate the original topics and intention distribution that generated the synthetic data. This results in learning a model which estimates the latent topic and intention distribution and can be broken into three sub problems:

4.1 Derivation [15 + 15 + 5 = 35 points]

Derive the following showing full details and using appropriate nomenclature for count variables.

(a) Complete joint distribution of JTI, i.e., $P(W, Z, R)$, where the capitalized random variables denote the family of all sub random variables mentioned with small cases in the above draft.

(b) Collapsed Gibbs Samplers, specifically, derive the following Gibbs conditional distribution:
   (i) $P(z_k = t', r_k = \hat{t} | w_k, W_{\neg k}, Z_{\neg k}, R_{\neg k}) \propto \cdots$ where $k = (d, j)$ denotes a specific token being assigned to topic t'.
   (ii) $P(z_k = i', r_k = \hat{\imath} | w_k, W_{\neg k}, Z_{\neg k}, R_{\neg k}) \propto \cdots$ where $k = (d, j)$ denotes a specific token being assigned to intention i'.

(c) The posterior distribution over the latent variables $\theta^I_{d,i}$, $\theta^T_{d,t}$, $\varphi^I_{i,v}$, $\varphi^T_{t,v}$

4.2 Implementing Collapsed Gibbs Samplers [25 + 6 + 3 + 3 + 3 = 40 points]

Implement the following methods in the driver class **IntentModelGibbsSampler**.

(a) `sampleFullConditional(int m, int n)`
Similar to LDA (as in **LDAGibbsSampler**), first carefully subtract the topic/intent counts form all the relevant count variables, perform sampling based on the samplers derived in the previous step. Then add back the sampled topic/intent to the appropriate count variables respecting the coding scheme detailed in `initialState()`.

(b) `updateParams()`
Similar to LDA (as in **LDAGibbsSampler**), correctly update the following variables: `thetasum[m][t]`, `thetasum_i[m][i]`, `phisum[t][w]`, `phisum_i[i][w]` for all documents m, topics, t, intents i and words w.

(c) Implement the posterior distribution of the latent variables θ_d , φ_t^T , and φ_i^I in the methods the `getTheta(),getPhi(),getPhi_i()` extending the version appearing in **LDAGibbsSampler** with suitable modification as applicable in this problem and its data structures/variable nomenclatures.

4.3 Visualizing the intermediate state of the latent topic/intentions during Gibbs sampling [5 points]

This has been already implemented in `printCurrentPhi()` which is called within the main Gibbs sampling method, `gibbs()`. You do not need to do anything but ensure that the implementation of `printCurrentPhi()` aligns well with your `sampleFullConditional()` and `updateParams()` method (i.e., they should respect the variables and the point estimates being taken in the implementation of `printCurrentPhi()`). If `sampleFullConditional()` and `updateParams()` methods are correctly implemented, then this would automatically generate the set of visualized topics and intents as they approach the posterior distribution. You will see that the original topics/intents (predefined in **IntentModelSyntheticData**) emerge from the raw data as Gibbs sampling estimation progresses. You are required to provide the set of images (which are also automatically generated by the existing implementation) to obtain full scores for this step with Synthetic data generated from **IntentModelSyntheticData** and Gibbs parameters in the class **IntentModelGibbsSampler** set to `model.configure(2000, 25, 50, 10)`.

Hint: One way to check you implementation is to compute the log-likelihood of the model. See the `getLogLiklihood ()` method in **LdaGibbsSampler** included in the codebase for this project, i.e., **GraphicalModelsInventory**. It computes the log value of the joint probability of the entire model. Also read [2] (text-est_heinrich.pdf included in the reading materials). It can be a bit tricky and require some thought to extend `getLogLiklihood` for **IntentModelGibbsSampler.** But if you can do it (i.e., derive both the Math and implement the code), it is a sure check that your implementation is correct. In fact part of this task is included in the next section.

4.4 Analyzing the variation of posterior distribution [5 x 4 = 20 points]: In this section we explore the posterior distribution of the model. For scoring in this section, you are required to: (1) Compute the complete model likelihood in its closed form, and (2) Generate a plot of model likelihood vs. Gibbs iteration number, (3) Compute the perplexity of the model in closed form using the ideas of query sampler presented in [2] (a sample implementation of perplexity computation of LDA is given in **LdaGibbsSampler** included in the codebase for this project, i.e., **GraphicalModelsInventory**), (4) Plot the perplexity vs. Gibbs iteration number.

5. Submission instructions

Submit the following for evaluation: (1) Typeset draft of derivation (4.1 (a, b, c), 4.4 (closed form of model's complete log-likelihood, and perplexity, plots of perplexity/model's log-likelihood vs. Gibbs iteration), (2) Codebase for (4.2, 4.3, 4.4).

References

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *J. Mach. Learn. Res.*, vol. 3, no. 4–5, pp. 993–1022, 2003.

[2] G. Heinrich, "Parameter estimation for text analysis," *Techincal Rep.*, 2005.