DATA ENGINEERING KODEBEAGLE LABS

LDA: NLP and Code Analysis – Part II

by
Sachin Tyagi

posted on
September 14, 2015 (September 14, 2015)

In the previous post I discussed the utility of building a topic model on a code corpus. In this post I want to discuss a popular topic model called LDA. Subsequently, I will share a few notes on implementing LDA on Spark for KodeBeagle.

So What is LDA?

LDA stands for Latent Dirichlet Allocation. As already mentioned it is one of the more popular topic models which was initially proposed by Blei, Ng and Jordan in 2003. It is a generative model which, according to Wikipedia, "allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar."

Lot of words to unpack here – latent, dirichlet, generative model. So let's do that.

First, what do we mean when we say LDA is a generative model? It basically means that LDA starts with a certain story about how the data that is to be examined got generated. Crucially, the generative story makes certain assumptions. These assumptions are as follows:

1. Each document is a mixture of topics.
2. Each topic is a mixture of words.
3. A document is a "bag of words" i.e. the order of words in a document is not important.

Now with these assumptions the generative story proceeds as –

1. For each document, sample a topic from the underlying topic distribution.
2. From the topic sampled in the step above, sample a word from the word distribution for that topic.
3. Repeat 1 and 2 for all words of all topics.

If you think about it the assumptions of the model are pretty reasonable. Except perhaps #3, which is debatable because word order can affect the topic of a document. This also explains why many of the subsequent improvements on LDA are the ones that tighten this assumption. But for now let's continue with this assumption as it makes the things simple without (hopefully) sacrificing too much.

Now this is all well and good but how do we put this generative model to use? If you look at it carefully, you will notice that the generative story is essentially a causal story. It describes how the data is 'caused' from the given topic and word distributions. But we already know the data! So, our problem is opposite. Given the data (or evidence), what can we know about the *hidden* structure – i.e the topics, their distribution over documents and the word distribution in those topics – that is likely to have generated this evidence or data. (This also explains the 'L' in LDA – 'latent' is just a synonym for 'hidden'). To tackle this we will need to make use of Bayesian networks.

Bayesian networks are a kind of probabilistic graphical models and provide a principled way of representing and reasoning about the probabilistic relationships between random variables. They are directed a-cyclic graphs or DAGs and represent the conditional dependencies (represented as edges) among the random variables (represented as vertices) they model.

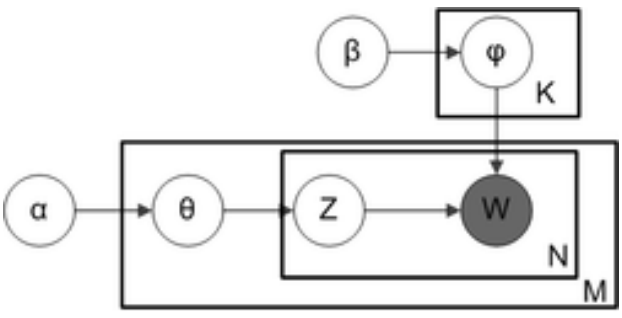
[For a very basic introduction to Bayesian networks see this wikipedia page. To dig deeper into graphical probabilistic models in general see Daphne Koller's excellent course on Coursera.]

So now let's build a Bayesian network for our LDA generative model. To begin with let's see how many nodes we will need in our Bayesian network. Let's say we have *M* documents with *N* words each. The vocabulary (distinct words) size is *V* and there are *K* latent topics.

- For each document we have to consider a topic distribution θ. So there will be *M* θs. Note that the θs are *K* dimensional vectors where each dimension represents the probability of a topic in a specific document.
- For each topic we have to consider a word distribution φ. So there will be *K* φs. Note that the φs are *V* dimensional vectors where each dimension represents the probability of a word occurring in a specific topic.
- Then for each word we have a topic assignment *z* (total *M* x *N* in number).
- Finally there will be observed variables – the actual words – *w*. Again total of *M* x *N* in number.

For any non-trivial corpus the large number of nodes (*K* + *M* + 2(*M* x *N*)) in the Bayes net for LDA will make it unwieldy. That's where the plate notation comes in. "**Plate notation** is a method of representing variables that repeat in a graphical model. Instead of drawing each repeated variable individually, a plate or rectangle is used to group variables into a subgraph that repeat together, and a number is drawn on the plate to represent the number of repetitions of the subgraph in the plate." [From Wiki]

Using plate notation our Bayesian network for LDA would look like this –



The biggest plate above has an index *M* which means that whatever is inside that plate is repeated *M* times. It is a plate which repeats for each document. Then the plate with index *N* repeats for each word (of each document as it's nested inside the *M* plate). The variables outside a plate are constant with respect to that plate. Also note that edge direction represents the 'causal' flow of influence. For instance, the topic distribution θ for a document causes a particular word instance to take a particular topic assignment *z*, then that *z* and word distributionφ for the topic selected 'causes' the actual identity of the word *w*.

Now, to summarize again here's what the notations used in the model above mean –

- As already mentioned *M, N, K* are the number of documents, words in each document and topics respectively. (Here it is assumed, for simplicity, that each document is of same size).
- *V*, as also mentioned above, is the size of the vocabulary or distinct words.
- θ is the topic distribution for a document. It is *K* dimensional.
- φ is the word distribution for a topic. It is *V* dimensional.
- *Z* is the topic assignment for each word in each document. It is *M x N* dimensional.
- *W* is the actual word observed. *M x N* dimensional.
- α and β are parameters for the φ and φ distributions respectively. We will discuss them in more detail later.

Now notice a few things –

- In the Bayesian network for LDA above the only observed variable is *W*, i.e. words in the documents. Everything else, including the variables of interest – $\theta, Z, \text{and } \varphi$ – are all hidden or unobserved in the data and need to be inferred.
- Also note that once we have *Z* (topic assignments for each word), calculating θ (distribution of topics in a document) and φ (word distribution in a topic) is straightforward.

So now let's put this all back together. That will need a bit of maths. I will intersperse the steps with comments to help make their meaning clear.

First let's begin with the full probability of the model which is just writing, notationally, how probable any combination of variable in the model is. This can be written as –

$$P(\boldsymbol{W}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{\varphi}; \alpha, \beta) = \prod_{i=1}^{K} P(\varphi_i; \beta) \prod_{j=1}^{M} P(\theta_j; \alpha) \prod_{t=1}^{N} P(Z_{j,t}|\theta_j) P(W_{j,t}|\varphi_{Z_{j,t}}),$$

On the left hand side are all the variables of the model. On the right is the chain rule applied according to the Bayesian network above. Let's unpack it. It says that the full probability distribution of the model is a product of –

- All the *K* word distributions (one for each topic).
- All the *M* topic distributions (one for each document).
- All the topic assignments *z*'s (one for each word in a document) given the topic distribution for that document.
- All the word identities *w*'s given the topic assignment *z* for that word.

Now, as noted above we can simply concentrate on *Z* and both θ and φ can be later constructed from it. (α and β are the inputs to the model and are not to be inferred here). So let's try to remove θ and φ from this equation. This can be done through marginalization or integration of θ and φ so that we have full probability distribution in terms of *Z* and *W* only.

$$P(\boldsymbol{Z}, \boldsymbol{W}; \alpha, \beta) = \int_{\boldsymbol{\theta}} \int_{\boldsymbol{\varphi}} P(\boldsymbol{W}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{\varphi}; \alpha, \beta)\, d\boldsymbol{\varphi}\, d\boldsymbol{\theta}$$

$$= \int_{\boldsymbol{\varphi}} \prod_{i=1}^{K} P(\varphi_i; \beta) \prod_{j=1}^{M} \prod_{t=1}^{N} P(W_{j,t} \mid \varphi_{Z_{j,t}})\, d\boldsymbol{\varphi} \int_{\boldsymbol{\theta}} \prod_{j=1}^{M} P(\theta_j; \alpha) \prod_{t=1}^{N} P(Z_{j,t} \mid \theta_j)\, d\boldsymbol{\theta}.$$

The second step above is just rearranging the θ and φ terms under their respective integrals. Also, since the distribution of topics in the documents –θ's- are independent of word distribution in topics – φ's- the two integration can be dealt with independently. (Also look at the Bayes net and note that there is no edge between θ and φ, which means the same thing that both are conditionally independent). So focusing only on the θ's, we can write the second half of the above equations as –

$$\int_{\boldsymbol{\theta}} \prod_{j=1}^{M} P(\theta_j; \alpha) \prod_{t=1}^{N} P(Z_{j,t} \mid \theta_j) d\boldsymbol{\theta} = \prod_{j=1}^{M} \int_{\theta_j} P(\theta_j; \alpha) \prod_{t=1}^{N} P(Z_{j,t} \mid \theta_j)\, d\theta_j.$$

Intuitively, the expression on the left represents the total probability of all *Z*'s given θ 's. Again refer to the Bayes net above and note that there're no edges between two documents, so they are conditionally independent. This means that we can simply calculate this value for the *j* th document and then multiply them all together to get the whole thing on the left. So we can just focus on the integral on the right –

$$\int_{\theta_j} P(\theta_j; \alpha) \prod_{t=1}^{N} P(Z_{j,t} \mid \theta_j)\, d\theta_j.$$

Now here's the important part – what should the $P(\theta_j; \alpha)$ be? Remember that the θ_j is a K dimensional vector whose each dimension represents how probable a topic is within a document. So for any given word in document *j*, θ_{ji} represents the probability that its topic is *i*. Now $P(\theta_j; \alpha)$ represents a distribution over probability vector θ_j, so this means that the following must be satisfied –

1. $\sum_{i=1}^{K} \theta_{ji} = 1$ for it to be valid probability distribution.
2. $\theta_{ji} \in (0, 1)$
3. And ideally, $P(\theta_j; \alpha)$ should also be able to encode the surety of my belief in the θ_{ji} 's. For example, let's say there are 4 topics. And you are asked the question how probable is each topic in a random document? You can say I am not sure so I will simply give them equal probability. So your θ_j will be $[0.25, 0.25, 0.25, 0.25]$. But suppose you have already analyzed lots of documents, and always find that all four topics are mostly equally likely.Your theta vector would still be $[0.25, 0.25, 0.25, 0.25]$, however your degree of belief in these values would be much higher. So your probability function should ideally capture this nuance.

This is exactly where the Dirichlet distribution comes in. It satisfies all of the above properties. For #3, the dirichlet distribution takes a single parameter α (which is also *K* dimensional) to represent the probability distribution of topics for a document. Higher the $\sum_i \alpha_i$, greater will be the concentration of θ's around the mean of the distribution. That is the reason why α is also referred to as concentration parameter.

Let's clarify this point through an example. Say *K*=4 and you choose your α as [1, 4, 4, 1]. So your θ will be distributed around [1/10, 4/10, 4/10, 1/10] (which is the mean of the distribution). But there's also a chance that your θ is something else –θ is random variable after all and distributed according to dirichlet distribution. Now the probability of theta being away from the mean will depends on the variance of the underlying dirichlet distribution. If the variance is more, there will be a relatively greater probability that the θ may be something other than the mean value. But here's the thing – the variance of a dirichlet distribution is inversely proportional to $\sum_i \alpha_i$. So if we provide lager α_i – remember it's called concentration parameter – our distribution will be concentrated around mean with lesser variance. Yet another way to think of alpha is as pseudo-counts – i.e. it is as if we have already made, before even looking at the current data, α_i number of observations for each of the ith topic. So, greater the α, more is our belief in the mean value of θ_j, and greater the number of new observations to the contrary that will be needed for us to believe in any another value of θ_{ji}.

So we substitute P(\theta_j;\alpha)$ in the integral with the well known probability density function for Dirichlet distribution. (Later we will do the same for φ as well. So now you also know what the 'D' in LDA stands for and why!)

$$\int_{\theta_j} P(\theta_j;\alpha) \prod_{t=1}^{N} P(Z_{j,t} \mid \theta_j)\, d\theta_j = \int_{\theta_j} \frac{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \prod_{i=1}^{K} \theta_{j,i}^{\alpha_i - 1} \prod_{t=1}^{N} P(Z_{j,t} \mid \theta_j)\, d\theta_j.$$

[Where Γ (called Gamma function) satisfies $\Gamma(n) = (n-1)!$.]

In the above equation we have simply substituted PDF for θ_j – $P(\theta_j;\alpha)$ assuming it is a Dirichlet distribution. Now let's tackle the last part $\prod_{t=1}^{N} P(Z_{j,t} \mid \theta_j)$. First, what does it mean? It stands for probability of topic assignments for all words in a document, given we already know the θ_j for the document. Now recall that θ_{ji} is the probability of i'th topic in the document. Now suppose there are 10 words in a document with θ as [0.5, 0.25, 0.25]. Now what is the probability that any 4 terms in the document are assigned to topic 1 and 3 each to topic 2 and 3? Well it is $(0.5)^4 * (0.25)^3 * (0.25)^3$. We express the exact same thing notationally as

$$\prod_{t=1}^{N} P(Z_{j,t} \mid \theta_j) = \prod_{i=1}^{K} \theta_{j,i}^{n_{j,(\cdot)}^{i}}.$$

where $n_{j,r}^{i}$ stands for number of words in document j assigned to topic i and taking the identity r from the vocabulary. So the $\theta_{j,i}^{n_{j,(\cdot)}^{i}}$ above means number of words (regardless of their identity) in document *j* assigned to topic *i*. Now we put the whole thing back together to get

$$\int_{\theta_j} \frac{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \prod_{i=1}^{K} \theta_{j,i}^{\alpha_i - 1} \prod_{i=1}^{K} \theta_{j,i}^{n_{j,(\cdot)}^{i}} \, d\theta_j$$

$$= \int_{\theta_j} \frac{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \prod_{i=1}^{K} \theta_{j,i}^{n_{j,(\cdot)}^{i} + \alpha_i - 1} \, d\theta_j.$$

Now the remarkable thing here is that the α_is for each topic simply add up with word count $\theta_{j,i}^{n_{j,(\cdot)}^{i}}$ for each topic. Didn't I mention that α_is are pseudo-counts! Think of them as topic word count we automatically assign to each document because of our 'prior' beliefs. Then my probabilities get updated once I see the actual data – i.e. the actual word counts $\theta_{j,i}^{n_{j,(\cdot)}^{i}}$.

Now we know from the properties of a dirichlet distribution, the following holds

$$\int_{\theta_j} \frac{\Gamma\left(\sum_{i=1}^{K} n_{j,(\cdot)}^{i} + \alpha_i\right)}{\prod_{i=1}^{K} \Gamma(n_{j,(\cdot)}^{i} + \alpha_i)} \prod_{i=1}^{K} \theta_{j,i}^{n_{j,(\cdot)}^{i} + \alpha_i - 1} \, d\theta_j = 1.$$

(Note that in the equation above α_i is replaced by $n_{j,(\cdot)}^{i} + \alpha_i$ but the form remains dirichlet so it still integrated to 1, as it is a valid PDF).

Now putting everything together we get the final equation with θ's integrated out. (That is what we initially set out to do, remember?)

$$\int_{\theta_j} P(\theta_j;\alpha)\prod_{t=1}^{N} P(Z_{j,t}\mid\theta_j)\,d\theta_j = \int_{\theta_j}\frac{\Gamma\big(\sum_{i=1}^{K}\alpha_i\big)}{\prod_{i=1}^{K}\Gamma(\alpha_i)}\prod_{i=1}^{K}\theta_{j,i}^{n^i_{j,(\cdot)}+\alpha_i-1}\,d\theta_j$$
$$=\frac{\Gamma\big(\sum_{i=1}^{K}\alpha_i\big)}{\prod_{i=1}^{K}\Gamma(\alpha_i)}\frac{\prod_{i=1}^{K}\Gamma(n^i_{j,(\cdot)}+\alpha_i)}{\Gamma\big(\sum_{i=1}^{K}n^i_{j,(\cdot)}+\alpha_i\big)}\int_{\theta_j}\frac{\Gamma\big(\sum_{i=1}^{K}n^i_{j,(\cdot)}+\alpha_i\big)}{\prod_{i=1}^{K}\Gamma(n^i_{j,(\cdot)}+\alpha_i)}\prod_{i=1}^{K}\theta_{j,i}^{n^i_{j,(\cdot)}+\alpha_i-1}\,d\theta_j$$
$$=\frac{\Gamma\big(\sum_{i=1}^{K}\alpha_i\big)}{\prod_{i=1}^{K}\Gamma(\alpha_i)}\frac{\prod_{i=1}^{K}\Gamma(n^i_{j,(\cdot)}+\alpha_i)}{\Gamma\big(\sum_{i=1}^{K}n^i_{j,(\cdot)}+\alpha_i\big)}.$$

Almost on similar lines, we integrate out the φ's as well. I will skip the steps (which you can see at wikipedia). But here's the final full probability distribution only in terms of Z and W with θ's and φ's integrated out.

$$P(\boldsymbol{Z},\boldsymbol{W};\alpha,\beta)$$
$$=\prod_{j=1}^{M}\frac{\Gamma\big(\sum_{i=1}^{K}\alpha_i\big)}{\prod_{i=1}^{K}\Gamma(\alpha_i)}\frac{\prod_{i=1}^{K}\Gamma(n^i_{j,(\cdot)}+\alpha_i)}{\Gamma\big(\sum_{i=1}^{K}n^i_{j,(\cdot)}+\alpha_i\big)}\times\prod_{i=1}^{K}\frac{\Gamma\big(\sum_{r=1}^{V}\beta_r\big)}{\prod_{r=1}^{V}\Gamma(\beta_r)}\frac{\prod_{r=1}^{V}\Gamma(n^i_{(\cdot),r}+\beta_r)}{\Gamma\big(\sum_{r=1}^{V}n^i_{(\cdot),r}+\beta_r\big)}.$$

(Where β is parameter for dirichlet distribution of words in a topic, pretty much analogous to α which was a parameter for topic distribution in documents.)

Now we need to infer Z, for which we can use the Bayesian rule as follows –

$$P(Z\mid W;\alpha,\beta)=\frac{P(Z;\alpha,\beta)P(W|Z;\alpha,\beta)}{\int_z P(Z,W;\alpha,\beta)}$$

Now the demoninator in the above equation is independent of any particular Z (as it is summation over all the Z's). So the above equation can be re-written as a proportionality.

$$P(Z\mid W;\alpha,\beta)\propto P(Z;\alpha,\beta)P(W\mid Z;\alpha,\beta)$$

But, $|\ Z\ |=K^V$ which makes the inference above analytically intractable. So we need to use some approximation. One such is Gibbs sampling. In it, every topic assignment z_i is handled one at a time while keeping z_{-i} (all other except z_i as constant). So we calculate a value for z_i and update it and then move to z_{i+1} (while now treating the new value of z_i as constant).

So the above equation for the n^{th} word in the m^{th} document, we can re-write the above equation as

$$P(Z_{m,n} = k \mid Z_{-(m,n)}, W_{-(m,n)}; \alpha, \beta)$$
$$\propto P(Z_{m,n} = k \mid Z_{-(m,n)}; \alpha, \beta) P(W_{(m,n)} \mid Z_{m,n} = k, Z_{-(m,n)}, W_{-(m,n)}; \alpha, \beta)$$

The above equation can be interpreted as –

- The left part represents how likely is it for $(m, n)^{th}$ word to belong to topic *k*.
- On the right hand side, there are two parts –
  - $P(Z_{m,n} = k \mid Z_{-(m,n)}; \alpha, \beta)$ represents how likely is it that the $(m, n)^{th}$ word, regardless of its identity, is derived from topic k. That is, how likely is topic k in document m.
  - $P(W_{(m,n)} \mid Z_{m,n} = k, Z_{-(m,n)}, W_{-(m,n)}; \alpha, \beta)$ represents how likely is the symbol at $(m, n)^{th}$ word location to come from topic *k* (regardless of the document).

[With the intuition behind the different parts of the equation stated, I am skipping the math and directly presenting the final expression. The math to final expression anyway involves mostly a clerical attention to all the indices and rearranging the terms without losing them].

$$P(Z_{m,n} = k \mid Z_{-(m,n)}, W_{-(m,n)}; \alpha, \beta)$$

$$\propto (n_{m,(.)}^{k,-(m,n)} + \alpha_k)\left(\frac{n_{(.),v}^{k,-(m,n)} + \beta_v}{\sum_{r=1}^{V} n_{(.),r}^{k,-(m,n)} + \beta_r}\right)$$

where,

- $n_{m,(.)}^{k,-(m,n)}$ is the number of words, excluding the $(m, n)^{th}$ word in document m that are assigned to topic *k*
- $n_{(.),v}^{k,-(m,n)}$ is the number of times the word identity v (at $(m, n)^{th}$ location) is assigned to topic k.

- $\sum_{r=1}^{V} n_{(.),r}^{k,-(m,n)}$ is the total number of words, in all documents, assigned to topic k.

Now, the above equation gives the probability for a particular word to be assigned to a topic given that the topic of every other word instance are known. But they are not!

So, here's the remarkable part – we can start with random topic assignment for each word, and then go through the whole corpus word by word, updating topic of each word according to the equation above, then after sufficient number of passes(also called a Gibbs iteration) through the whole corpus, we will converge to the actual topic assignments! Pretty cool, right? (Gibbs sampling is an algorithm belonging to the larger class called MCMC – Markov Chain Monte Carlo. You can look them up for the maths behind it).

Phew! So finally we are done with our LDA derivation. The algorithm, using Gibbs sampling can be summarized as follows –

- Tokenize the documents into collection of words.
- Fix sensible values for α, β and *K*.
- Initialize random topic for each word in the corpus.
- Repeat the following (Gibbs iteration) till happy. For each word of each document, calculate the following for each topic *k* –
  - $\hat{\theta}$ as $n_{m,(.)}^{k,-(m,n)} + \alpha_k$
  - $\hat{\phi}$ as $\left(\frac{n_{(.),v}^{k,-(m,n)} + \beta_v}{\sum_{r=1}^{V} n_{(.),r}^{k,-(m,n)} + \beta_r}\right)$
  - Assign a the topic *k* (from among total *K* topics) according to the proability – ($\hat{\theta}$ x $\hat{\phi}$)

P.S. In the next post, I will share the notes on implementing LDA with Gibbs sampling on Apache Spark.

Read Comments Leave a Comment

TAGGED IN

Gibbs Sampling | Latent Dirichlet Alloc... | LDA

Sachin Tyagi

2 COMMENTS

Rob Grzywinski

4 years ago

Hello! Just a quick poke to let you know that your LaTeX expansions aren't working in many cases making the post *very* hard to read.

Sachin Tyagi AUTHOR

4 years ago

Thanks for the feedback Rob. It's been fixed now. Further feedback welcome. ☺