LIGHT ON MATH MACHINE LEARNING

Intuitive Guide to Latent Dirichlet Allocation

Thushan Ganegedara
Aug 23, 2018 · 11 min read ★

Topic modelling refers to the task of identifying topics that best describes a set of documents. These topics will only emerge during the topic modelling process

(therefore called latent). And one popular topic modelling technique is known as *Latent Dirichlet Allocation* (LDA). Though the name is a mouthful, the concept behind this is very simple.

To tell briefly, LDA imagines a fixed set of topics. Each topic represents a set of words. And the goal of LDA is to map all the documents to the topics in a way, such that the words in each document are mostly captured by those imaginary topics. We will systematically go through this method by the end which you will be comfortable enough to use this method on your own.

This is the fourth blog post in the series of ***light on math machine learning A-Z***. You can find the previous blog posts linked to the letter below.

A B C D* E F G H I J K L M N O P Q R S T U V W X Y Z

*denotes articles behind Medium Paywall.

Why topic modeling?

What are some of the real world uses topic modelling has? Historians can use LDA to *identify important events in history* by analysing text based on year. Web based libraries can use LDA to *recommend books based on your past readings*. News providers can use topic modelling to *understand articles quickly or cluster similar articles*. Another interesting application is *unsupervised clustering of images*, where each image is treated similar to a document.

What's unique about this article? Is this another fish in the sea?

The short answer is a big NO! I've swept through many different articles out there. And there are many great articles/videos giving the intuition. However most of them stop at answering questions like:

- What is the intuition behind LDA?
- What is a Dirichlet distribution?

Which I do talk about but don't believe we should stop there. The way these models are trained is a key component I find missing in many of the articles I read. So I try to answer few more questions like:

- What is the mathematical entity we're interested in solving for?

- How do we solve for that?

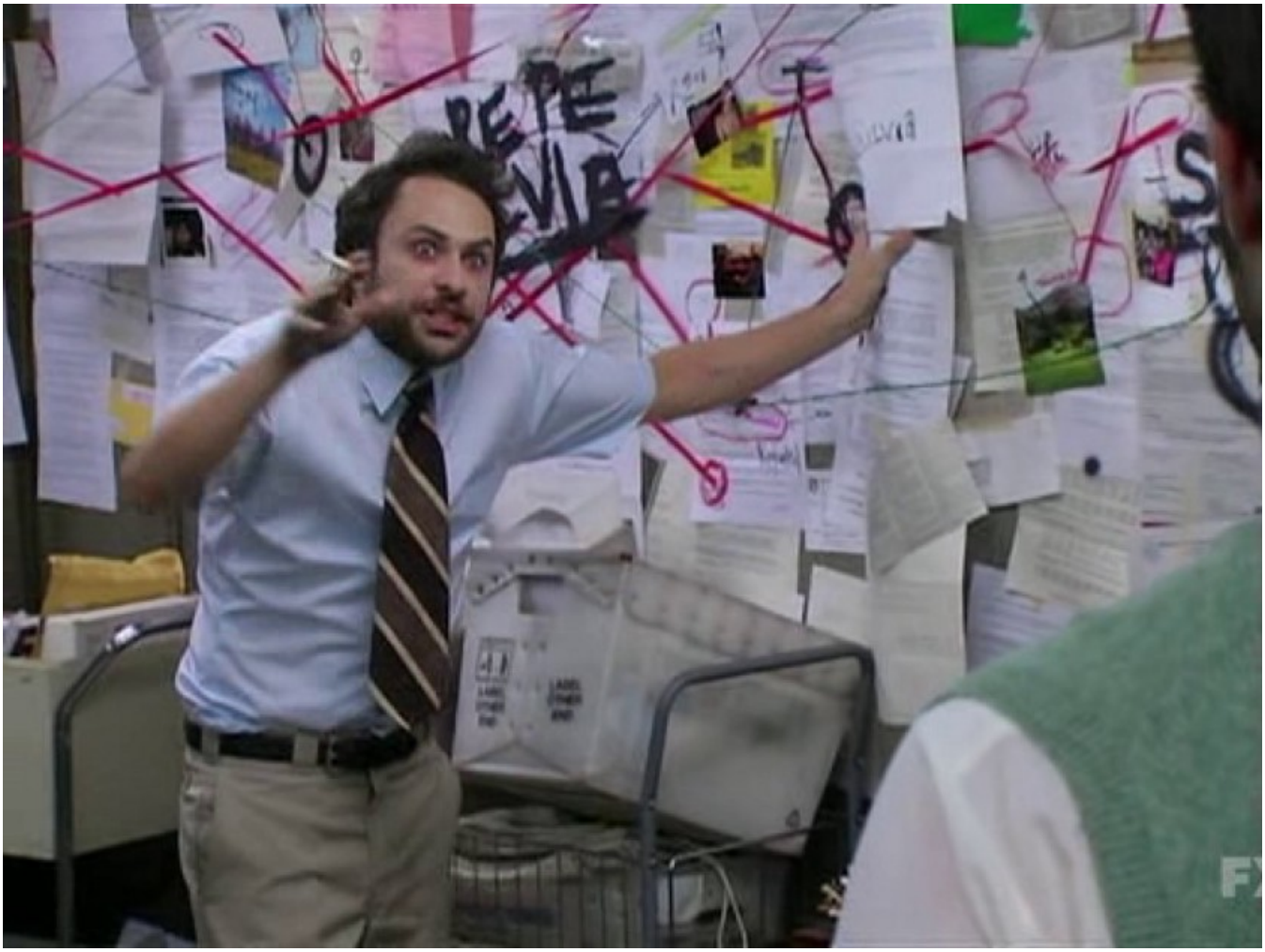
What is the big idea behind LDA?

Once you understand the big idea, I think it helps you to understand why the mechanics in LDA are the way they are. So here goes;

> Each document can be described by a distribution of topics and each topic can be described by a distribution of words

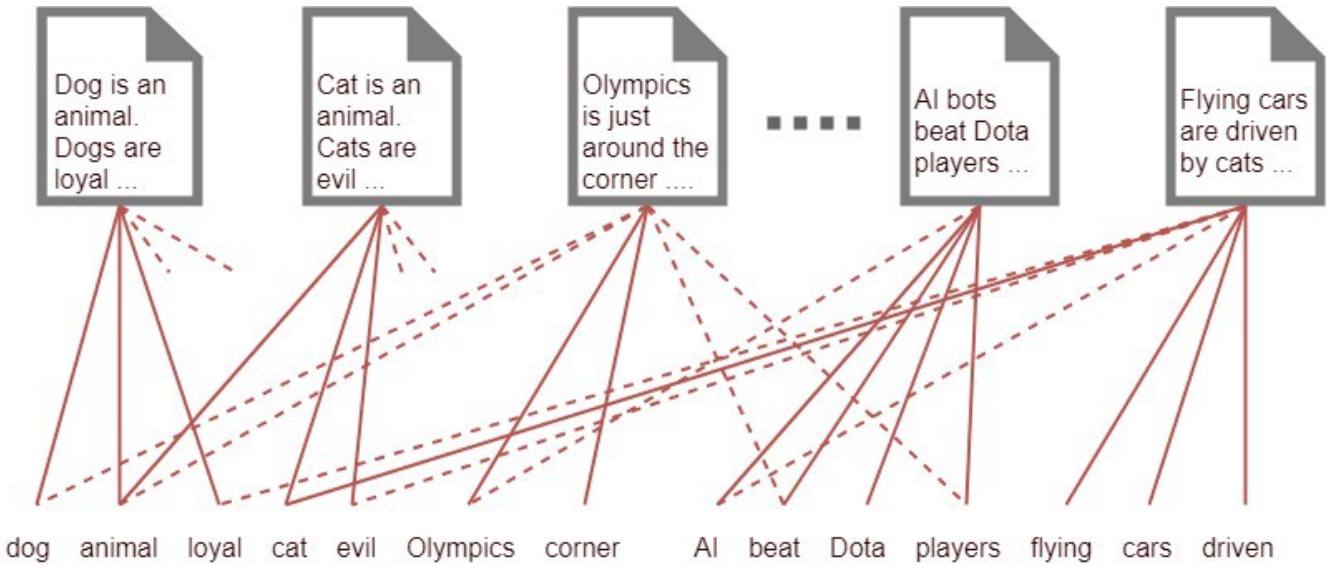
But why do we use this idea? Let's imagine it through an example.

LDA in layman's terms



Say you have a set of 1000 words (i.e. most common 1000 words found in all the documents) and you have 1000 documents. Assume that each document on average

has 500 of these words appearing in each. How can you understand what category each document belongs to? One way is to connect each document to each word by a thread based on their appearance in the document. Something like below.



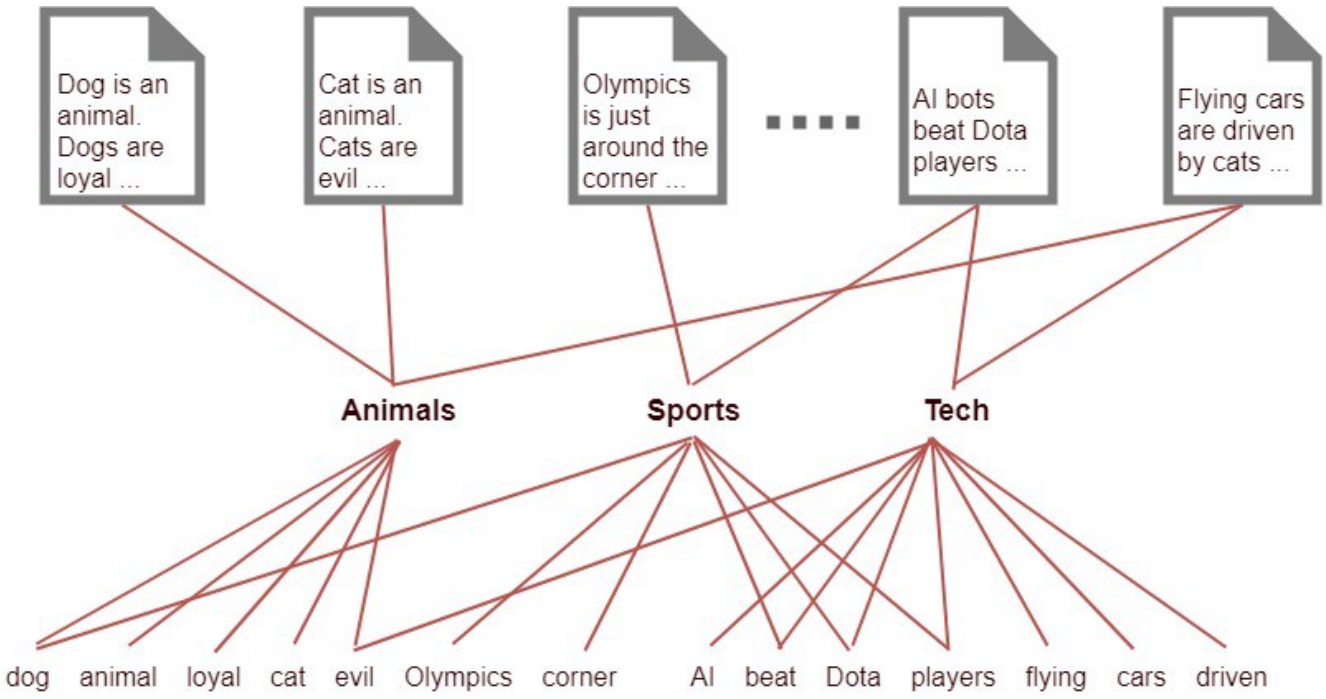
Modeling documents just with words. You can see that we can't really infer any useful information due to the large amount of connections

And then when you see that some documents are connected to same set of words. You know they discuss the same topic. Then you can read one of those documents and know what all these documents talk about. But to do this you don't have enough thread. You're going to need around 500*1000=500,000 threads for that. But we are living in 2100 and we have exhausted all the resources for manufacturing threads, so they are so expensive and you can only afford 10,000 threads. How can you solve this problem?

Go deeper to reduce the threads!

We can solve this problem, by introducing a latent (i.e. hidden) layer. Say we know 10 topics/themes that occur throughout the documents. But these topics are not observed, we only observe words and documents, thus topics are latent. And we want to utilise this information to cut down on the number of threads. Then what you can do is, connect the words to the topics depending on how well that word fall in that topic and then connect the topics to the documents based on what topics each document touch upon.

Now say you got each document having around 5 topics and each topic relating to 500 words. That is we need 1000*5 threads to connect documents to topics and 10*500 threads to connect topics to words, adding up to 10000.

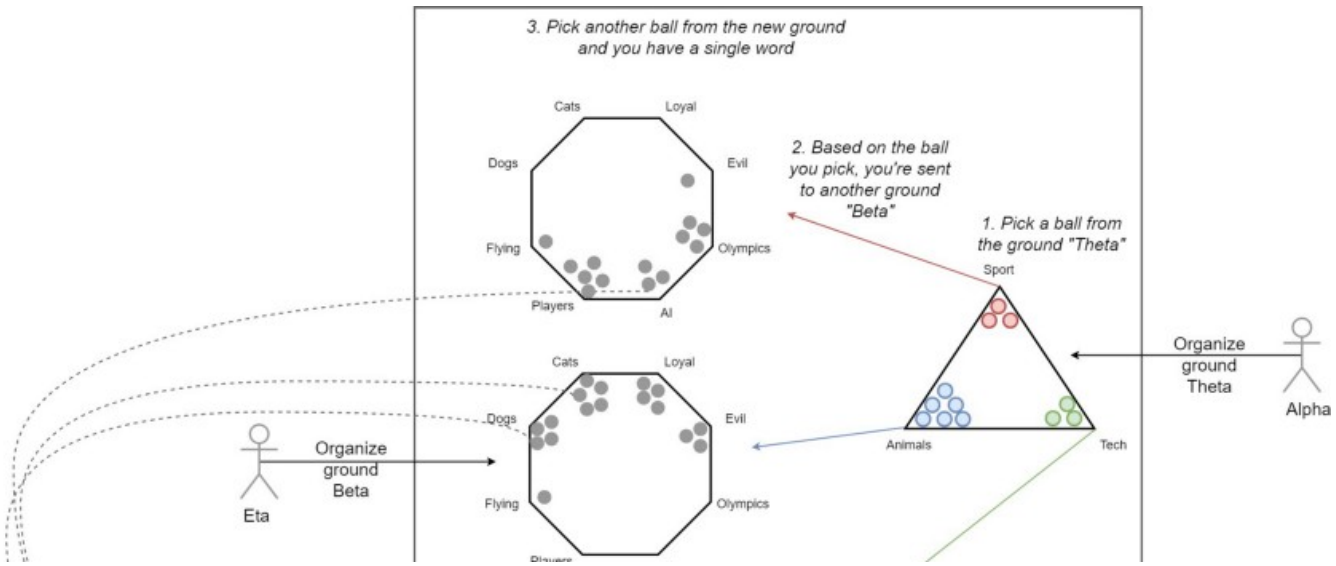


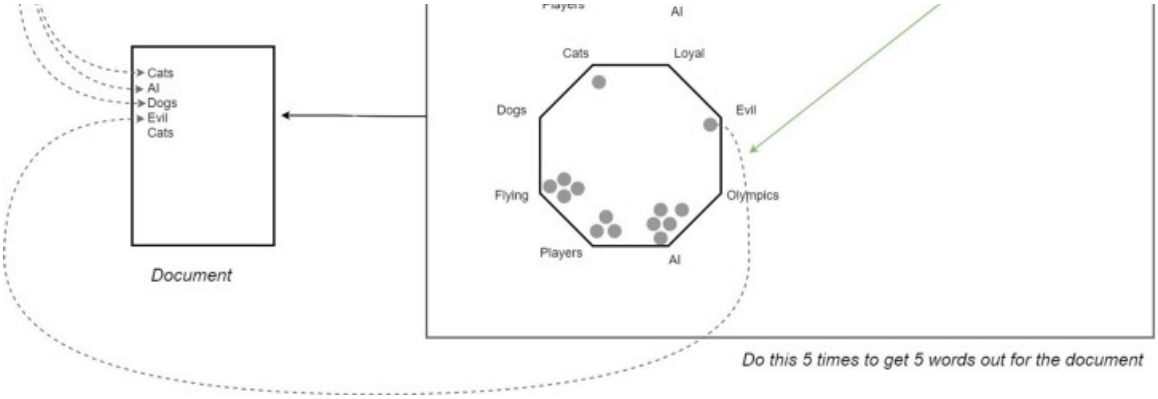
Words are modeled by a set of topics and documents are modeled by a set of topics. The relationships are clearer than the first example because there's a fewer connections than the first example.

Note: The topics I use here ("Animals", "Sports", "Tech") are imaginary. In the real solution, you won't have such topics but something like *(0.3*Cats,0.4*Dogs,0.2*Loyal, 0.1*Evil)* representing the topic "Animals". That is, as mentioned before, each document is a distribution of words.

A different view: how LDA imagine documents are generated?

To give more context to what's going on, LDA assumes the following generative process is behind any document you see. For simplicity let us assume we are generating a single documents with 5 words. But the same process is generalisable to *M* documents with *N* words in each. The caption pretty well explain what's going on here. So I won't reiterate.





How a document is generated. First α (alpha) organise the ground θ (theta) and then you go and pick a ball from θ. Based on what you pick, you're sent to ground β (beta). β is organised by η (Eta). Now you pick a word from β and put it into the document. You iterate this process 5 times to get 5 words out.

This image is a depiction of what an already-learnt LDA system looks like. But to arrive at this stage you have to answer several questions, such as:

- How do we know how many topics are there in the documents?
- You see that we already have a nice structure in the grounds that help us to generate sensible documents, because organisers have ensured the proper design of the grounds. How do we find such good organisers?

This will be answered in the next few sections. Additionally, we're going to get a bit technical this point onwards. So buckle up!

Note: LDA does not care the order of the words in the document. Usually, LDA use the bag-of-words feature representation to represent a document. It makes sense, because, if I take a document, jumble the words and give it to you, you still can guess what sort of topics are discussed in the document.

Getting a bit mathematical ...

Before diving into the details. Let's get a few things across like notations and definitions.

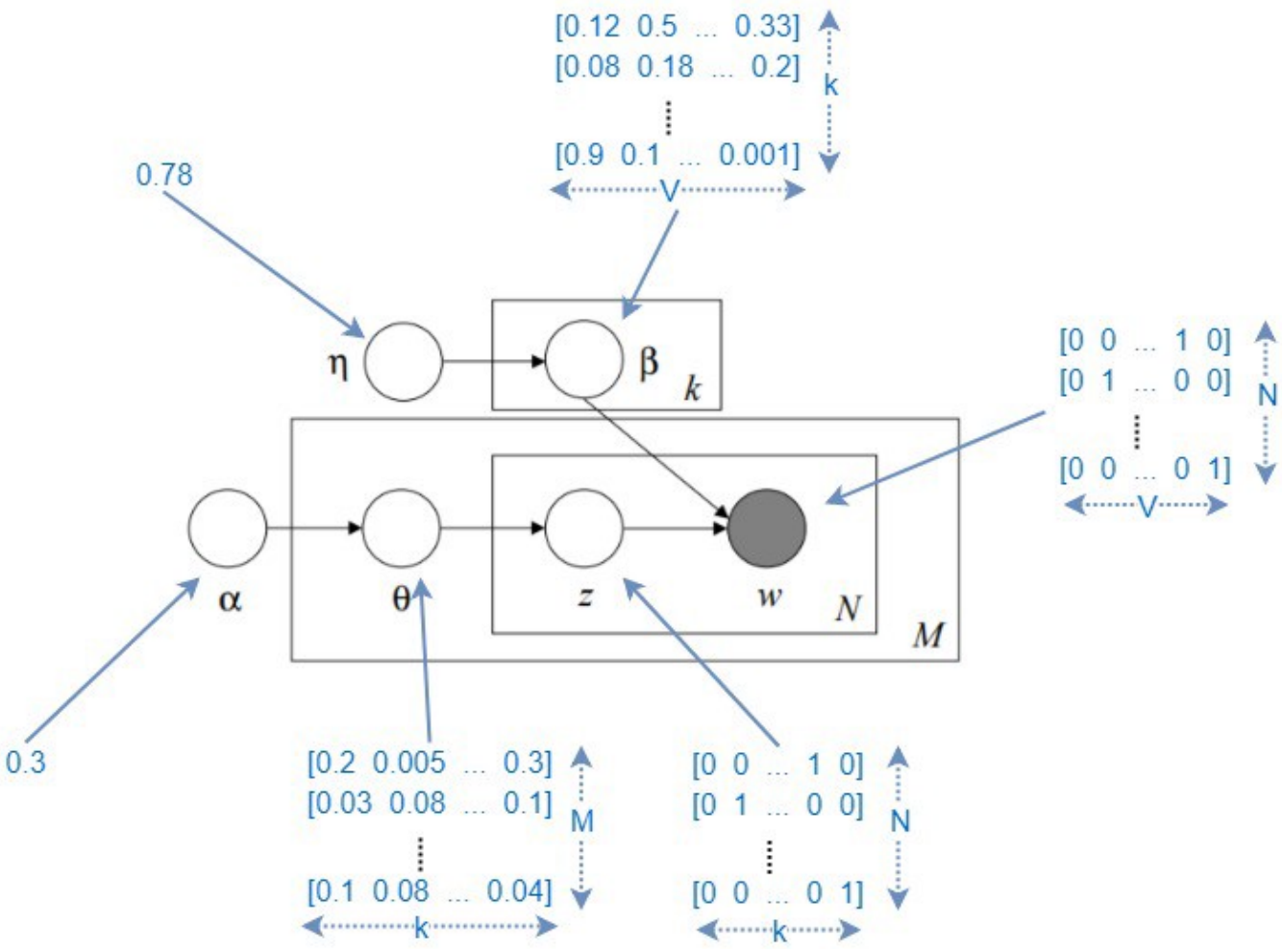
Definitions and notations

- k — Number of topics a document belongs to (a fixed number)
- V — Size of the vocabulary
- M — Number of documents
- N — Number of words in each document

- w — A word in a document. This is represented as a one hot encoded vector of size V (i.e. V — vocabulary size)
- $\boldsymbol{w}$ (bold w): represents a document (i.e. vector of "w"s) of N words
- D — Corpus, a collection of M documents
- z — A topic from a set of k topics. A topic is a distribution words. For example it might be, *Animal = (0.3 Cats, 0.4 Dogs, 0 AI, 0.2 Loyal, 0.1 Evil)*

Defining document generation more mathematically

First let's put the ground based example about generating documents above, to a proper mathematical drawing.



Graphical model of the LDA. Here I mark the shapes of the all the possible variables (both observed and hidden). But remember that θ, z, and β are distributions, not deterministic values

Let's decipher what this is saying. We have a single α value (i.e. organiser of ground θ) which defines θ; the topic distribution for documents is going to be like. We have M documents and got some θ distribution for each such document. Now to understand

things more clearly, squint your eyes and make that *M* plate disappear (assuming there's only a single document), woosh!

Now that single document has *N* words and each word is generated by a topic. You generate *N* topics to be filled in with words. These N words are still placeholders.

Now the top plate kicks in. Based on η, β has some distribution (i.e. a Dirichlet distribution to be precise — discussed soon) and according to that distribution, β generates k individual words for each topic. Now you fill in a word to each placeholder (in the set of N placeholders), conditioned on the topic it represents.

Viola, you got a document with *N* words now!

Why are α and η constant?

α and η are shown as constants in the image above. But it is actually more complex than that. For example α has a topic distribution for each document (θ ground for each document). Ideally, a *(M x K)* shape matrix. And η has a parameter vector for each topic. η will be of shape *(k x V)*. In the above drawing, the constants actually represent matrices, and are formed by replicating the single value in the matrix to every single cell.

Let's understand θ and β in more detail

θ is a random matrix, where θ*(i,j)* represents the probability of the *i* th document to containing words belonging to the *j* th topic. If you take a look at what ground θ looks like in the example above, you can see that balls a nicely laid out in the corners not much in the middle. The advantage of having such a property is that, the words we produce are likely to belong to a single topic as it is normally with real-world documents. This is a property that arise by modelling θ as a Dirichlet distribution. Similarly β(i,j) represents the probability of the *i* th topic containing the *j* th word. And β is also a Dirichlet distribution. Below, I'm providing a quick detour to understand the Dirichlet distribution.

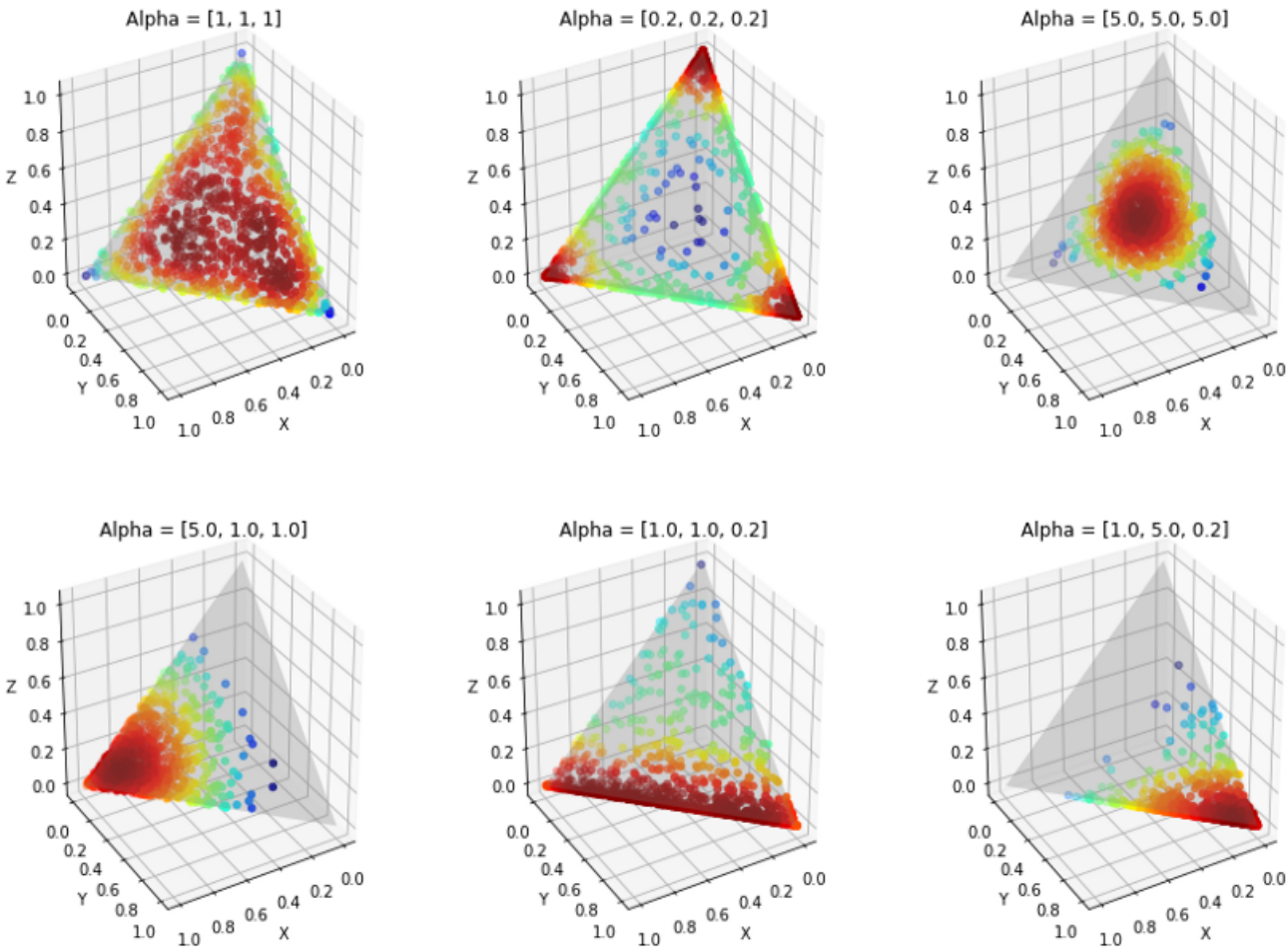
Quick detour: Understanding the Dirichlet distribution

Dirichlet distribution is the multivariate generalisation of the Beta distribution. Here we discuss an example of a 3-dimensional problem, where we have 3 parameters in α that affects the shape of θ (i.e. distribution). For an N-dimensional Dirichlet distribution you have a N length vector as α. You can see how the shape of θ changes

with different α values. For example, you can see how the top middle plot shows a similar shape to the θ ground.

The main take-away is as follows:

> Large α values push the distribution to the middle of the triangle, where smaller α values push the distribution to the corners.



How the distribution of θ changes with different α values

How do we learn the LDA?

We still haven't answered the real problem is, how do we know the exact α and η values? Before that, let me list down the latent (hidden) variable we need to find.

- α — Distribution related parameter that governs what the distribution of topics is for all the documents in the corpus looks like

- θ — Random matrix where θ(i,j) represents the probability of the i th document to containing the j th topic
- η — Distribution related parameter that governs what the distribution of words in each topic looks like
- β — A random matrix where β(i,j) represents the probability of i th topic containing the j th word.

Formulating what do we need to learn

If I'm to mathematically state what I'm interested in finding it is as below:

$$P(\theta_{1:M}, \mathbf{z}_{1:M}, \beta_{1:k} | \mathcal{D}; \alpha_{1:M}, \eta_{1:k})$$

It looks scary but contains a simple message. This is basically saying,

I have a set of M documents, each document having N words, where each word is generated by a single topic from a set of K topics. I'm looking for the joint posterior probability of:

- θ — A distribution of topics, one for each document,
- **z** — N Topics for each document,
- β — A distribution of words, one for each topic,

given,

- D — All the data we have (i.e. the corups),

and using parameters,

- α — A parameter vector for each document (document — Topic distribution)
- η — A parameter vector for each topic (topic — word distribution)

But we cannot calculate this nicely, as this entity is intractable. Sow how do we solve this?

How do I solve this? Variational inference to the rescue

There are many ways to solve this. But for this article, I'm going to focus on variational inference. The probability we discussed above is a very messy intractable posterior (meaning we cannot calculate that on paper and have nice equations). So we're going to approximate that with some known probability distribution that closely matches the true posterior. That's the idea behind variational inference.

The way to do this is to minimise the KL divergence between the approximation and true posterior as an optimisation problem. Again I'm not going to swim through the details as this is out of scope.

But we'll take a quick look at the optimization problem

$$\gamma^\star, \phi^\star, \lambda^\star = \text{argmin}_{(\gamma,\phi,\lambda)} D(q(\theta, \mathbf{z}, \beta|\gamma, \phi, \lambda)||p(\theta, \mathbf{z}, \beta|\mathcal{D}; \alpha, \eta)$$

γ , ϕ and λ represent the free variational parameters we approximate θ,**z** and β with, respectively. Here D(q||p) represents the KL divergence between q and p. And by changing γ,ϕ and λ, we get different q distributions having different distances from the true posterior p. Our goal is to find the γ* , ϕ* and λ* that minimise the KL divergence between the approximation q and the true posterior p.

With everything nicely defined, it's just a matter of iteratively solving the above optimisation problem until the solution converges. Once you have γ* , ϕ* and λ* you have everything you need in the final LDA model.

Wrap up

In this article we discussed about Latent Dirichlet Allocation (LDA). LDA is a powerful method that allows to identify topics within the documents and map documents to those topics. LDA has many uses to it such as recommending books to customers.

We looked at how LDA works with an example of connecting threads. Then we saw a different perspective based on how LDA imagine a document is generated. Finally we went into the training of the model. In this we discussed a significant amount of mathematics behind LDA, while keeping the math light. We took a look at what a Dirichlet distribution looks like, what is the probability distribution we're interested in finding (i.e. posterior) and how do we solve that using variational inference.

I will post a tutorial on how to use LDA for topic modelling including some cool analysis as another tutorial. Cheers.

References

Here are some useful references for understanding LDA is anything wasn't clear.

Prof. David Blei's original paper

An intuitive video explaining basic idea behind LDA

Lecture by Prof. David Blei
