Understanding and Applying the Latent Dirichlet Allocation model to first-order Markov Chains

Jean-Noël Rivasseau
Department of Computer Science
University of British Columbia
Vancouver, Canada
jnriva@cs.ubc.ca

Abstract

In this project I explored the implementation of latent Dirichlet allocation (LDA) for first order Markov chains. LDA is a generative probabilistic model; the main goal in this project was to compare it to a simpler mixture of multinomials model, and to understand its advantages over it. The specificity of the LDA model lies in its ability to associate a particular document (Markov chain, in my case), with *several* topics or clusters, whereas the mixture of multinomial makes the very restrictive assumption that every document is associated with a single cluster.

1 Introduction

This project started with the initial desire of being able to model music intelligently: given a collection of musical documents, how can these documents be clustered, according only to their intrinsic data? Several researchers have already explored the problem of musical classification. For example, musical data was used jointly with text and images in an experiment to cluster CD albums, and perform fast information retrieval [2]. It should be made clear from the beginning that the musical data we are interested in is expressed in terms of higher level abstract features, not the actual sound one may hear. Such abstract features are for example present in a music score; Western composers have used a standard conventional musical notation for several centuries. On the computer world side, similar data can be found in GUIDO notation files [4], or to a lesser extent MIDI files.

One convenient mean to express this data in a way that can be used in statistical models is to use Markov chains. Given a certain number of states, Markov chains represent a sequence of transitions from one state to another. Thus the probability of being in a given state depends only on the previous state. A matrix of probabilities P can be built, where p_{ij} represents the probability to get to the j state if you are currently in the i state. Note that for each i, $\sum_j p_{ij} = 1$. For music, one can see states as notes; then transitions in the Markov chain model the jumps from one note to an other in the music. I didn't exactly choose that model and added

one level to this approach; first a score is analyzed and translated in a sequence of transitions. Then the Markov chain is built from these *transitions*: thus states already represent transitions and the Markov chain's transitions represent one jump to a certain transition from another. One can easily persuade himself that in this model, you need 3 notes to build one Markov transition.

I want to emphasize that in the end, this project didn't focus that much on the specificity of musical data. In the following sections, the fact that the Markov chains we are dealing with eventually represent musical data can be easily forgotten. Actually, the results obtained are very general, and could be applied to any other application with Markov chains, such as games.

Once we model our data with a collection of Markov chains (one per document), the next step is to choose a statistical model, to be able to retrieve the essential statistical features of our corpus. In essence, the problem is to come up with a short, but meaningful, description of our huge initial data collection. Then one can achieve efficient classification, or even generation of new documents similar to the training set. This leads, in the case of music, to computer musical composition. There exist, in the machine-learning litterature, a variety of interesting models. This project outlines the differences between two of them, the "simple" mixture of multinomial model, and the more elaborate model of latent Dirichlet association, described in Blei and al. paper [1]. I found the LDA model to be much richer; it allows for a meaningful classification, and is much better suited for music generation. Thus this model could probably be used in many cases, although it requires more complex computations (because it can't be solved analytically). This project was very useful for me in the sense that it allowed me to actually understand *why* LDA performs better than the simple mixture of multinomials. Thus I think I have grasped some of the key issues underlying each model, which will undoublty prove very useful in the future. That project report will consequently emphasize how the results I obtained shed light on some of the questions I was asking myself.

2 Presentation of the two models

2.1 Notation and terminology

I will use the same terminology than the one used in Blei and al.'s paper [1]. Thus I refer the reader to section 2 of that paper, if he's not familiar with the notations.

Note that Minka and Lafferty use a different terminology in their article "Expectation-Propagation for the Generative Aspect Model" [5]. My implementation of LDA for Markov chains is based upon Minka's available code.

2.2 Mixture of Multinomials

It is assumed than there are z underlying topics, and each document is associated with *a single topic*. That assumption is crucial; we shall see why later. The probability of a document is:

$$p(w_d) = \sum_z p(z) \prod_{i=1}^{n_s} \prod_{j=1}^{n_s} (\beta_{i,j,z})^{N_{i,j,d}}$$

Then the complete likelihood for the corpus is just the product of probabilities for each document:

$$p(D) = \prod_{d=1}^{M} p(w_d)$$

Most of the times, the probability of a given document also depends upon the first state of the Markov's chain. In this project, I didn't implement that assumption, which is straightforward to add, if need be.

The β here can be viewed as the probabilities for the transitions, given a particular cluster. $p(z)$ is the probability of a given cluster (the mixture weights), and the matrix N is the data. A particularly interesting matrix appears in the mixture model, generally denoted $\xi_{d,z}$. These ξ are used as hidden variables when using the Expectation-Maximization algorithm to estimate the parameters of the model (specifically, $p(z)$ and $\beta_{i,j,z}$). They can be viewed as the probabilities for a given document d, to be in a certain cluster z. We shall explain that in more detail later.

2.3 Latent Dirichlet Allocation

Here again, it is assumed that there are z underlying topics, but this time a document can exhibit several aspects (topics) to different proportions. When generating a document, the topic is not sampled once, but one time for every transition. To understand better the LDA model, I once again refer the reader to Blei and al.'s paper [1]. Here I will only recall the main results.

The probability of a document is:

$$p(w_d) = \int p(\theta|\alpha) \prod_{i=1}^{n_s} \prod_{j=1}^{n_s} \left(\sum_z p(z|\theta) \cdot \beta_{i,j,z} \right)^{N_{i,j,d}} d\theta$$

The β play the same role as in the mixture model. However, here we have the corpus parameter α, that replace the parameters $p(z)$. α is a Dirichlet parameter that eventually control the proportions of the different clusters in a given document; when generating a new document, θ is sampled from α, and then the topic for each transition is sampled from a Multinomial distribution of parameter θ (thus in fact θ is the equivalent of $p(z)$).

Two basic computational tasks for the LDA model are performing inference and estimating parameters for a given corpus. These tasks can't be done analitically and thus we resort to various numerical methods. The one we will use is variational inference, described in Blei and al.'s paper [1]. A more recent article by Minka and Lafferty [5] describe an Expectation-Propagation algorithm for LDA, supposedly more efficient. Although I used Minka's code as the fundation for my implementation, I didn't use the Expectation-Propagation algorithm and sticked with the simpler variational bound method.

When using this method, two variational parameters (which are document specific) appear and must be estimated: ϕ and γ. Of particular interest to us is γ: it can be seen as the proportions of the topics in a given document. Thus it may be considered as an equivalent to ξ in the mixture model. The following section will show that these parameters are in fact quite different in their meaning, and that's the whole interest of the LDA model.

3 Applying both models to synthetical data

Let's consider the three following Markov chains:

(1) 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 ...

(2) 5 4 3 2 1 5 4 3 2 1 5 4 3 2 1 ...

(3) 1 3 5 2 4 1 3 5 2 4 1 3 5 2 4 ...

Obviously, we are dealing with 3 very simple clusters, one for each sequence. Cluster one would only have the transitions $1 \rightarrow 2$, $2 \rightarrow 3$, $3 \rightarrow 4$, $4 \rightarrow 5$ and $5 \rightarrow 1$, cluster two would have $5 \rightarrow 4$ and so on. Let's apply both models to this corpus.

3.1 Clustering the synthetical corpus

As expected, we find three clusters with EM methods applied to the mixture of models (both ML or MAP give good results, although with ML we have to initially give the correct number of clusters as an input parameter. MAP doesn't have that restriction and if we specify an initial number of 10 clusters, for example, seven of those will go to a zero probability and in the end we'll only have the three correct clusters). Each one of this cluster has a 1/3 probability, as shown in the Figure 1.

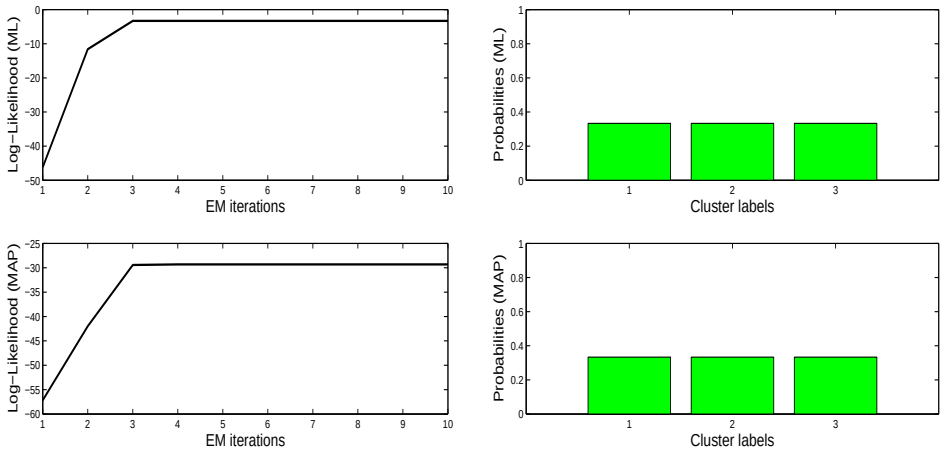

Figure 1: Typical Log-Likelihood graphs for the Mixture Model, and clustering of three Markov chains

When looking at the obtained β, everything is as expected. And the ξ map each sequence to its correct cluster (according to the β), with a probability of 1 for the correct cluster, and 0 for the others.

Applying LDA, we also obtain the expected results. The β are exactly the same as those obtained with the mixture model. Looking at γ, we also find what we were hoping for: the values map each sequence to their correct cluster.

3.2 Generating a new sequence

We are here at the heart of the problem. Now that we have trained our models with this simple training set, let's see how they will deal with a new sequence. We sample a new sequence in the following way: We start at the initial state 1. Next, for every

transition, we choose the transition corresponding to sequence 1 (1 2 3 4 5 1 ...) with probability p_1, and with probabilities p_2 and p_3 the transition corresponding to the one in sequence 2 and 3 respectively. Of course, $p_1 + p_2 + p_3 = 1$. So we have created a new sequence that contain elements coming from our 3 clusters, the proportions being the probabilities p_1, p_2, p_3.

We then expect that our models will correctly determine those proportions. Thus, computing the ξ would give the probabilities $P = [p_1, p_2, p_3]$, and the γ would verify the equation:

$$\gamma - \alpha = N \cdot P \qquad \text{(N is the length of the Markov chain)}$$

But, computing ξ gives an error! The algorithm assigns 0 to each element of ξ, thus, the new sequence doesn't belong to any cluster. Its log-likelihood would be $-\infty$. This is the perfect example to understand the crucial assumption being made in the mixture model: each document must "belong to" one cluster and *only one*. If we look at the formula to compute ξ_z, everything becomes clear:

$$\xi_z = p(z) \prod_{i=1}^{n_s} \prod_{j=1}^{n_s} (\beta_{i,j,z})^{N_{i,j}}$$

ξ_z is 0 because we can't *mix* the different clusters! In our example, only one transition is allowed for every cluster, thus for each i and z only one $\beta_{i,j,z}$ is non null. But, since for our new sequence we used transitions from all 3 clusters, the model is unable to map it to any of the three clusters (in the product, there is always one ${\beta_{i,j,z}}^{N_{i,j}}$ term which is null), and returns a $-\infty$ log-likelihood...

So it is essential to understand that the ξ don't represent the proportions of transitions from each cluster, in a given sequence. ξ really represent the probability that a given Markov chain belongs to a certain cluster. Under the simple mixture model, you can't allow the transitions distribution to vary between several different topics. This is very restrictive; for music, that means that if we have somehow obtained three different clusters, one for Jazz, the other for Classical and the last for Heavy Metal, we can't hope to create or recognize a musical work which is a blend of these three styles.

That's the essential difference with LDA, where the transitions distribution, on the other hand, can vary smoothly between the different topics. Evaluating γ for our new sequence produces the expected results, which means that using LDA, one can determine the values of p_1, p_2, p_3 without knowing them in the first place. This is quite powerful. Figure 2 reproduces the computed γ, and compares it to the actual values for p_1, p_2, p_3 (for 200 sampled Markov chains). Results are self-explicit: correct values are found.

4 Results on larger data set, generation of a new document

Once I was sure my implementation of LDA was correct, I did some tests, for the fun and also to see what results I would obtain with actual musical data. 21 scores were thus converted from MIDI format to GUIDO notation (see [4]). From GUIDO notation it was easy to extract the transitions, and to form the eventual Markov chains, with 49 states (this allows for transitions equal to 2 octaves - 24 semitones -, in either "direction"). For simplification, chords were reduced to one note, and rests were ignored. 3 topics (clusters) were choosen. A typical log-likelihood curve (for the EM variational bound algorithm) was obtained (Figure 3). Of course, EM is increasing at each iteration, which is good!

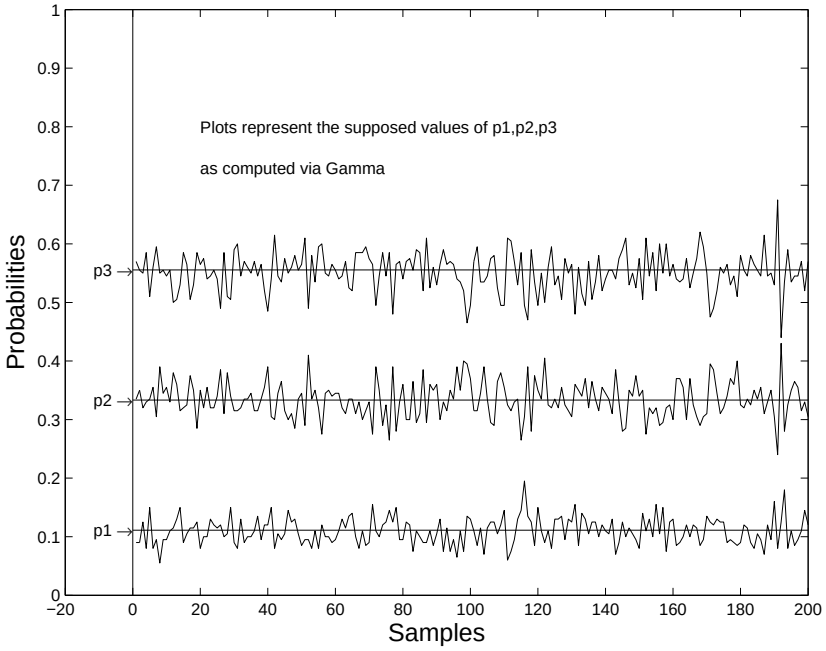

Figure 2: LDA detected topics proportions, for synthetical Markov Chain samples

The analysis of the results (which mainly consists in looking at the elements of γ) is a little bit tricky, since it is difficult to know, for a human, which fundamental aspects are captured by the model and translate into different styles (or clusters)! Still, I find some of the results encouraging. Of these 21 scores, 18 were guitar scores, and 3 belonged to other instruments. All these 3 other instruments are clustered into the same cluster, which leads me to believe that note transitions can somehow characterize instruments. I also deliberately chose many of these scores to be different guitar parts of the same song, and another worthy observation is that for one song, generally all parts go to the same cluster. So, definitely, LDA is able to capture some features of musical data.

The transition matrices are very sparse though, and that's why I believe that better results could be achieved by using higher order Markov chains. One important thing to note too is that we are dealing with documents that can be seen as "bags of words" (Blei and al.'s expression), or here, "bags of transitions". By this, I mean that the model doesn't take into account the order of the transitions, which is a severe restriction since musical works are generally built using several different "sections" (for exemple, intro, chorus, bridge, solo...), and it would be very valuable to be able to recognize them.

I finally tried generating a new music score. With LDA, the generative process is extremely simple: one just has to sample a Dirichlet parameter, and then repeatedly sample the topic (Multinomial distribution of the previously sampled Dirichlet parameter) and the transition (according to the β Multinomial distribution for the chosen cluster). I didn't change the parameter α, so none of the three clusters was favored for the sampling of θ (as a sidenote, α is in fact a parameter of the corpus, and can be trained like β. The computational methods required to do that are a little bit complex, thus I didn't train α and kept all its elements constant equal

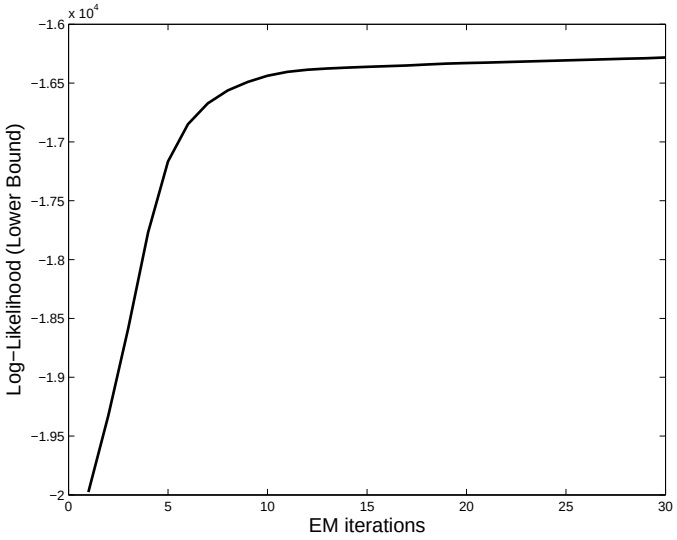

Figure 3: Typical Complete Log-Likelihood for the LDA model (Variational Bound Algorithm)

to 2.0). I ran an additional test, which consisted in recording the sampled θ and evaluating γ for the newly sampled scores. Like with the synthetical data, I again obtained good results and the correct θ were determined, this time with a much larger and complex data set!

Here are the first notes of my generated score:

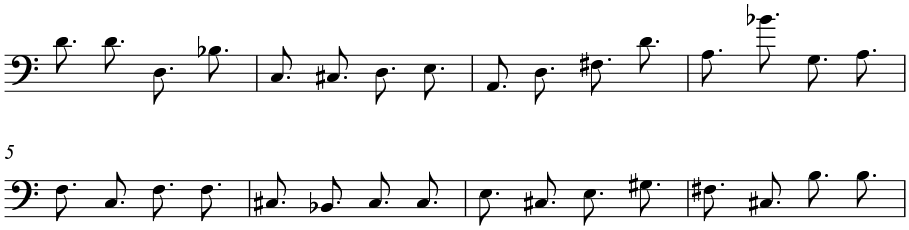

Figure 4: My very first 100% computer generated score! I hope it won't be the last...

5 Conclusions

This project allowed me to explore in depth the fundamental mechanisms behind two statistical generative models, the simple mixture of multinomials model, and the Latent Dirichlet Allocation model. The latest proves to be much richer because it possesses the ability to capture, for a single document, various topics in different proportions: this project mainly focused on proving that. Although computationally more complex, LDA could be used with success in many applications, notably classification and information retrieval. For example, useful extensions using LDA could be added to the work presented in Nando de Freitas and al. paper [2].

As far as music composition is concerned, I intend to explore some new ideas. For exemple, I believe one very interesting path is to try to determine "patterns" in the musical data, before applying a statistical model. A recent work [3] does just that; it uses a very simple statistical model (an unigram model) but combines that to a previous pattern analysis in order to generate new musical works. I think this is a successfull approach. In the near term, I hope to deepen my knowledge of the whole Machine-Learning field, which will be a definite prerequisite to start building a more complete statistical model for music, including not only information about notes, but also key elements such as tempo, different instruments...

References

[1] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. In *Neural Information Processing Systems 14*, 2001.

[2] E. Brochu, N. de Freitas, and K. Bao. The sound of an album cover: probabilistic multimedia and information retrieval, 2003.

[3] Shlomo Dubnov, Gerard Assayag, Olivier Lartillot, and Gill Bejerano. Using Machine-Learning Methods for Musical Style Modeling. *IEEE Computer Society*, 2003.

[4] Hoos H., Hamel K., Renz K., and Kilian J. The GUIDO Notation Format - a Novel Approach for Adequately Representing Score-Level Music. In *Proceedings ICMC*, pages 451–454, 1998.

[5] Thomas Minka and John Lafferty. Expectation-Propagation for the Generative Aspect Model. *Uncertainty in Artificial Intelligence (UAI)*, 2002.