DIRECT AND INVERSE INFERENCE IN MUSIC DATABASES: HOW TO MAKE A SONG FUNK?

Patrick Rabbat
Sophis
patrick.rabbat@sophis.net

François Pachet
Sony CSL
pachet@csl.sony.fr

ABSTRACT

We propose an algorithm for exploiting statistical properties of large-scale metadata databases about music titles to answer musicological queries. We introduce two inference schemes called "direct" and "inverse" inference, based on an efficient implementation of a kernel regression approach. We describe an evaluation experiment conducted on a large-scale database of fine-grained musical metadata. We use this database to train the direct inference algorithm, test it, and also to identify the optimal parameters of the algorithm. The inverse inference algorithm is based on the direct inference algorithm. We illustrate it with some examples.

1. INTRODUCTION

Large databases of metadata are now available in many domains of digital content such as music (allmusic) or films (imdb). However, the scaling up of content databases makes it increasingly difficult and expensive to maintain manually these metadata databases. The apparition of collaborative tagging addresses this issue by exploiting the power of large distributed networks of users. However, the basic issue of maintaining these repositories of information and coping with their possible incompleteness remains open.

The aim of our study is to design algorithms to turn large-scale databases of musical information into knowledge bases. More precisely, we wish to exploit these databases to perform statistical inferences from partial descriptions of items in order to infer new information about the content.

The inferences we target must be both statistically coherent with the data and computationally tractable. The inferences we consider in this study are of two sorts: direct and inverse. Direct inference consists in inferring the most plausible value of an attribute given a set of "observed attributes". These inferences can be used typically to lower the maintenance cost of the database: when new items are encountered, only a subset of attributes would then be needed to be observed, and the other ones could be inferred with some degree of confidence. Inverse inference answers a different question: given a set of observed attributes (possibly complete, i.e. a fully described title), and a target attribute value, we wish to know the minimal modifications to apply to this initial observation set to reach this target attribute value. An application of direct inference is algorithms for automatic music categorization. Some attributes like 'Style Rap' or 'Distorted Guitar' can be relatively well estimated from the analysis of the acoustic signal [2]. However, acoustic classifiers fail to compute more "high-level" attributes like "Mood sad" or "Lyrics cynical" with a satisfactory performance. The idea of such a hybrid approach is to first compute the best acoustic attributes and, in a second stage, infer the remaining attributes using direct inference.

As an example of inverse inference, we can consider a computer-aided composition tool that suggests musicians how to modify their compositions: starting from an initial composition, the system would observe some "low-level" attributes. The musician could then ask the system what attributes to change, and how, in order to, say, make his title sound "Funk", or not "aggressive", or "harmonious". Inverse inference yields the minimal modifications of the initial title to increase optimally the corresponding probabilities.

In this paper, we propose an algorithm for direct and inverse inference, based on an efficient implementation of a kernel regression approach. We describe an evaluation experiment conducted on a large-scale database of fine-grained musical metadata. We use this database to train the direct inference algorithm, test it, and also to identify the optimal parameters of the algorithm. Inverse inference is based on the direct inference algorithm. We illustrate it with real world examples.

1.1. State-of-the-art

To perform our inferences, we would need ideally a statistical model of our attributes in the form of a Bayesian Network, yielding a compact representation of the underlying distribution of the attributes. If we had such a model inference would be carried out with approximate methods suitable for large networks.

We tried to learn such a model of our random variables with the SBNS algorithm proposed in [3]: since our data is sparse, we can use *frequent sets* (i.e. sets of attributes co-

occurring more than some threshold. For more details, see [1]) to build local models. Using frequent sets dramatically decreases the complexity of the structural learning of a Bayesian network model, as remarked in [5].

To carry out inferences in the large network we obtained, we used *Mean Field Theory* (MFT). MFT is a statistical physics approximation to solve the many-body problem. It consists in replacing all interactions to any one body with an average interaction, turning the N-body problem into $N(N-1)/2$ 2-body problems. When applied to Bayesian Networks for inference, MFT means estimating the real distribution of the inferred nodes given the observed nodes with a simpler distribution; this simpler distribution assumes independence between inferred nodes given observed nodes. An efficient implementation of MFT can be found in [8] and complete details about variational inference methods in [6].

Unfortunately, the network we obtained was too densely connected even for MFT to perform in reasonable time. As an example, the computation of the MFT equation for one title, given an observable set, would take approximately 5 minutes, so cross-validation on our database (described in the next section) would take about 5 months, which is not acceptable. In this context, inverse inference would be even less tractable. However, our database is probably not as sparse as those used in [3]. This paper is an attempt to address this challenge.

1.2. Description of the data

For this study we have used a music dataset comprising 37,000 popular songs of various styles. Each song is described by 948 binary ($0/1$ valued) attributes. Attributes describe low-level information (like the presence of "acoustic guitar", or the "tempo range" of the song) as well as high-level characteristics (like the "style" of a song, the mood emerging from it, etc.). The attributes are grouped in 17 categories: *Style*, *Genre*, *Musical setup*, *Main instruments*, *Variant*, *Dynamics*, *Tempo*, *Special*, *Era/Epoch*, *Metric*, *Country*, *Situation*, *Mood*, *Character*, *Language*, *Popularity*, and *Rhythm*. The complexity of the database makes it impossible to describe it in details here, but we are only concerned in this study by the number of attributes and the quality of the inferences obtained.

The database we considered for this study is *sparse*: The mean number of attributes set to true per song (occupation factor) is 4% (i.e. 40 on a total of 948). Sparseness suggests the dominant role of the true-valued attributes versus false-valued attributes for a given song. Therefore, in the analysis of the database, our inference algorithm should treat differently true values and false values. Another feature of the database is its *redundancy*. For instance, attributes like 'Country Greece' and 'Language Greek' are extremely correlated.

This justifies the presence of inter-attribute dependencies, that our inference algorithm will attempt to exploit.

1.3. Problem Statement

In the context of our Boolean multi-attribute database, we can formulate our problem as follows. Let D denote the number of attributes ($D = 948$), and $(A_i)_{1\le i\le D}$ the set of attributes modeled as $0/1$ valued random variables. We consider each song description as a realization of the random vector A and thus denote A_i^s the value of the i^{th} attribute ($1 \le i \le D$) for the s^{th} song ($1 \le s \le N$) where N is the number of songs in the database ($N = 37{,}000$). Let $\Pr(\cdot)$ denote the underlying probability distribution of the random vector A. We wish to accomplish two tasks:

Direct inference: Given an incomplete description of a music title, determine the most probable values of the remaining uninformed attributes. Formally, if for a subset $I \subset \{1, \dots, D\}$ the observed values of the attributes $\left(A_{I_1}, \dots, A_{|I|}\right)$ (denoted A_I) are $\left(a_{I_1}, \dots, a_{I_{|I|}}\right)$ (denoted a_I), we wish to estimate the most probable values of the non-observed attributes, i.e. compute:

$$e^* \in \operatorname{argmax}_{e\in\{0,1\}^{D-|I|}} \Pr(A_{\bar{I}} = e \mid A_I = a_I)$$

Inverse inference: Given an incomplete description of a music title *and* target values for a subset of target attributes, how can one modify the initial description to achieve the target values with maximum confidence, with a minimal modification? Formally, for a subset $I \subset \{1, \dots, D\}$ and $J \subset \{1, \dots, D\}$ such that $I \cap J$ is empty, the observed values of A_I are a_I and the target fields A_T are meant to be a_T, but are not, so we suppose that $\Pr(A_T = a_T \mid A_I = a_I) < 0.5$. Let then δ_A be an indicator function with value 1 if A is true, and 0 otherwise. We say that σ is a *flip* of the observed attributes A_I if:

$$\sigma: \{0,1\}^{|I|} \longrightarrow \{0,1\}^{|I|}$$
$$(u_i)_{1\le i\le |I|} \mapsto \left(\delta_{i\notin D(\sigma)} u_i + \delta_{i\in D(\sigma)}(1-u_i)\right)_{1\le i\le |I|}$$

The subset $D(\sigma) \in I$ characterizes the flip σ: this definition means that the flip will inverse its parameters which index belong to $D(\sigma)$ and leave the remaining parameters unchanged. The *order* of the flip is $d(\sigma) = |D(\sigma)|$. It represents the number of modifications on the parameter of the flip. Let S denote the space of all flips (its cardinality is $2^{|I|}$). The inverse inference problem consists in solving the following combinatorial optimization problem:

$$\sigma^* \in \operatorname{argmin}_{\sigma\in S}[d(\sigma) - \lambda \Pr(A_T = a_T \mid A_I = a_I)]$$

Here $\lambda > 0$ is a free parameter representing a trade-off between minimal flip order and maximum probability of achieving the target state. Since I could be of cardinality up to 947, the search space S is huge so an important feature of our algorithm is the evaluation speed of this functional form.

2. DIRECT INFERENCE

In this section we describe our direct inference algorithm and evaluate its performance.

2.1. Description of the algorithm

Computing $\Pr(A_{\bar{I}} = e | A_I = a_I)$ for all $e \in \{0,1\}^{D-|I|}$ is combinatorial and computationally intractable for the dimensions we are dealing with ($D - |I| \approx 700$), so the basic assumption to break the intractability is that the variables $A_{\bar{I}_1}, \dots, A_{\bar{I}_{|\bar{I}|}}$ are independent given A_I. It then suffices to compute separately $\Pr(A_{\bar{I}_k} = 1 | A_I = a_I)$ for all $k \in \{1, \dots, |\bar{I}|\}$ to determine the most probable values of the inferred attributes $A_{\bar{I}}$. Although this assumption might seem contradictory with our primary aim to model dependencies, the intuition is that the inferred $A_{\bar{I}}$ attributes only depend on observed attributes A_I, meaning that subjective attributes emerge solely from objective (possibly acoustic) attributes that will form our observation I. This independence assumption is less strong as the size of the observation set grows. However, the choice of the attributes A_I will have to be dealt with carefully, and we discuss this issue later on.

To compute $\Pr(A_{\bar{I}_k} = 1 | A_I = a_I)$ we use a kernel regression estimator [4] and [7]:

$$\hat{p}_k(a_I) = \frac{\sum_{s=1}^{N} \omega_\theta\big(d(a_I, A_I^s)\big) A_{\bar{I}_k}^s}{\sum_{s=1}^{N} \omega_\theta\big(d(a_I, A_I^s)\big)}$$

where $d(a_I, A_I^s)$ is a distance function indicating the 'proximity' between our observed attributes a_I and the corresponding values of attributes A_I for the s-th song in the database. For instance, $d(\cdot,\cdot)$ could be the Euclidian distance in $\mathbb{R}^{|I|}$ counting the number of differences between two binary strings passed as arguments. However, the Euclidian distance treats equivalently both values 0 and 1. Yet the sparseness of the data indicates clearly that two binary strings are more similar if they have common 1s than if they have common 0s. To emphasize the important role of common ones, we rather use the Jaccard distance:

$$d(u,v) = d\big((u_i)_{1\le i\le p}, (v_i)_{1\le i\le p}\big) = \frac{\sum_{i=1}^{p} \delta_{u_i \neq v_i}}{\sum_{i=1}^{p} \delta_{u_i \neq v_i} + \sum_{i=1}^{p} \delta_{u_i=1}\delta_{v_i=1}}$$

The numerator is the number of differences between the two arguments, and the denominator is composed of two terms: the first one is again the number of differences and the second term is the number of "1" the strings have in common. Notice that for all $u, v \in \mathbb{R}^p$ $0 \le d(u,v) \le 1$.

$\omega_\theta(\cdot)$ is a weighting function with a real parameter $\theta \in \mathbb{R}^+$. An obvious property of this function is that it decreases towards 0 as its parameter increases. Intuitively, this means that the closer a song is from our observation a_I (using distance d) the greater its weight should be. The integral of the weighting function $\omega_\theta(\cdot)$ should also be finite. Notice that the weighting function needs not be normalized since the estimator $\hat{p}_k$ is already normalized. A simple choice for ω_θ is a Gaussian filter $\omega_\theta(x) = \exp\left(-\frac{x^2}{\theta^2}\right)$ but since θ is a free parameter that we have to tune, a nice property would be to have an intuitive understanding of the value θ. For example, with a simple linear decreasing function $\omega_\theta(x) = \left(1 - \frac{x}{\theta}\right)_+$, the value of θ is just a threshold distance above which songs have a weight set to 0, and are thus not taken into account in the estimation $\hat{p}_k$. Note that if $\theta \to \infty$ our estimator is an empirical mean on all the songs of the database, each song having identical importance, no matter the distance. If $\theta \to 0^+$ then $\omega_\theta(x) = \begin{cases} 1 \text{ if } x = 0 \\ 0 \text{ if } x \neq 0 \end{cases}$ and we are only estimating the empirical distribution on the data.

2.2. Evaluation & Experiments

This section introduces experiments to assess the performance of our algorithm. We also study the influence of several parameters of the algorithm.

2.2.1. Performance criterion

Our database contains about 4% of attributes set to 1. This unbalancedness forbids the use of precision or recall to assess our classifiers (a dumb classifier that always predicts an attribute $A_{\bar{I}_k}$ to 0 would have an accuracy of 96%). Instead, we use the more suited F-measure. For a given attribute, *F-measure* (for the true class) is defined as the mean of precision and recall, $F = \frac{2\pi\rho}{\pi + \rho}$. A classifier always responding false has a F-measure of 0.

2.2.2. Observed Attributes

In this paragraph we discuss the different choices for the observation set I. The choice can either be driven by applications, or by performance: if the goal is to complete an acoustic automatic categorization system, then there is a canonical order for all the attributes defined by the decreasing performance of some acoustic classifier. We call this order the *acoustic order*. Another possible choice for I is the set of *modifiable attributes*. This set of observations will be used in the inverse inference problem and has been determined by hand: modifiable attributes are low-level attributes that a musician can easily modify (like instruments, tempo, etc.). On the other hand, if what we are looking for is to increase the performance of our inference algorithm, the choice for I is not obvious. The simplest way to formulate the problem is to choose I as one of the subsets of $\{1, \dots, D\}$ of fixed cardinality m maximizing some global performance criterion $\mathcal{L}$ for inference:

$$I_m^* \in \underset{\substack{I \subset \{1,\dots,D\} \\ |I| = m}}{\operatorname{argmax}} \mathcal{L}\left(\left[\hat{p}_k(a_I)\right]_{\substack{1 \le k \le |\bar{I}| \\ a_I \in \{0,1\}^m}} \right)$$

The arguments of $\mathcal{L}$ are all possible responses to an inference query. The global performance criterion should 'integrate' in a certain sense all possible inference responses to all $a_I \in \{0,1\}^m$. Of course, this optimization problem is completely intractable and we shall restrict our empirical study to the case of the *acoustic* and *modifiable* attributes.

2.2.3. Experiments

In this paragraph, we assess the influence of the free parameters of the algorithm, namely θ and ω_θ and the influence of the choice of the observation I. The performances were computed using leave-one-out cross-validation on the whole dataset.

2.2.3.1 Influence of the smoothing parameter

First, we evaluate the influence of the free parameter $\theta > 0$ on the overall performance. In this experiment, we thus fix the observation I to be the 100 first attributes in the acoustic order.

On Figure 1, we show the performance of our algorithm with θ varying in the interval $[0,1[$. The reason we restrict the search to this interval is because intuitively we do not want to take into account all the songs in the database: songs with large distance (close to 1) should be ignored and, as we mentioned earlier, θ can be interpreted as a threshold distance. We plotted the F-measure for several individual attributes versus the smoothing parameter θ on the range $[0,1[$.

All of the attributes exhibit a bell-shaped F-measure vs. θ curve. This shows that there exists a maximum of the performance for some optimal value of the smoothing parameter. The figure also shows that this optimal value is different from one attribute to another. If the smoothing parameter was unique for all attributes and chosen to maximize the performance of, say, attribute 'Country France', the performance of the attribute 'Variant Wah-wah' would be zero whereas it could be near 80% with an individual fine smoothing tuning. This suggests that we should use one smoothing parameter per attribute to achieve maximum performance as opposed to the classic approach of maximizing a global performance criterion over all attributes. On figure 1 we plotted the cumulative distribution function (cdf) for the set of F-measures (over all inferred attributes) obtained for several choices of the smoothing parameter. A perfect classifier would have a cdf starting at the origin straight to point $(1,0)$ and from then straight to point $(1,1)$.

The worst classifier would have a cdf curve starting at the origin, straight to point $(0,1)$ and then straight to point $(1,1)$. Visually, the best classifier among our choices for the smoothing parameter is thus the closest curve to that of the perfect classifier. Figure 2 confirms that choosing individually optimized smoothing parameters for each attribute largely outperforms a global optimized parameter.

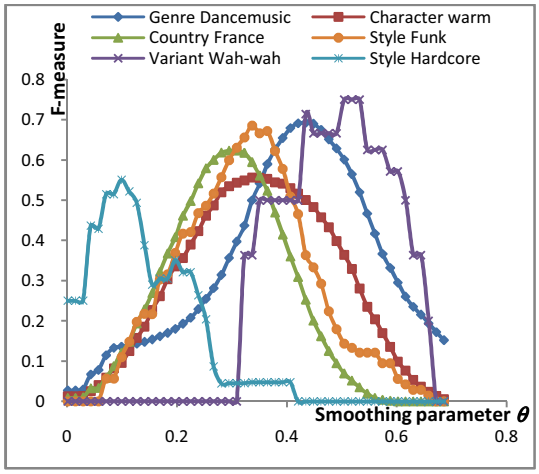

Figure 1: Influence of the smoothing parameter on the performance of several classifiers.

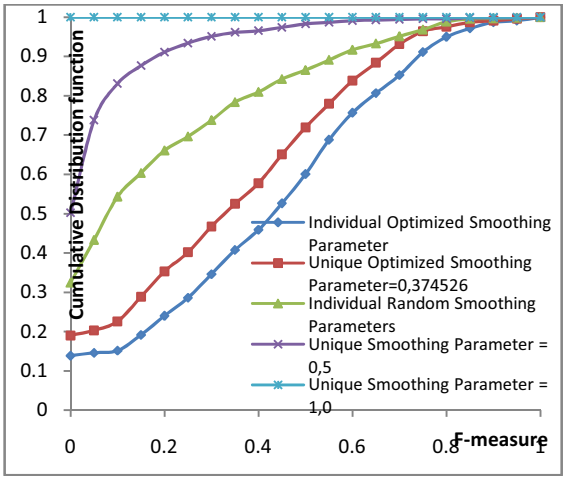

Figure 2: CDF of the F-measures over all inferred attributes for several choices of smoothing parameters. We select a unique parameter (for all attributes) that maximizes the sum of F-measures.

2.2.3.2 Influence of the weighting function

A well-known fact of kernel regression is that the choice of the smoothing parameter θ is more important than the choice of the weighting function. To confirm this in our case, we computed the performances for all inferred attributes for several functional forms for ω_θ and where θ was optimally chosen for each attribute as discussed above. The observation I is still the first 100 attributes in the acoustic order. On Figure 3, we plotted the cdf for each set of F-measures obtained for several choices of the weighting function.

Figure 3 shows that although there is a slight performance increase for the Gaussian function over other functional forms (with individually optimized smoothing parameters),

the improvement is negligible compared to the performance of a classifier with Gaussian weighting function and random choice for the smoothing parameters. Clearly, the choice of the smoothing parameters is critical and no particular effort should be dedicated to the choice of the weighting function. We thus choose to use the linear weighting function, as discussed in the algorithm description, since exponentials are expensive to compute. Linear weighting functions are a rather good tradeoff between performance and time.

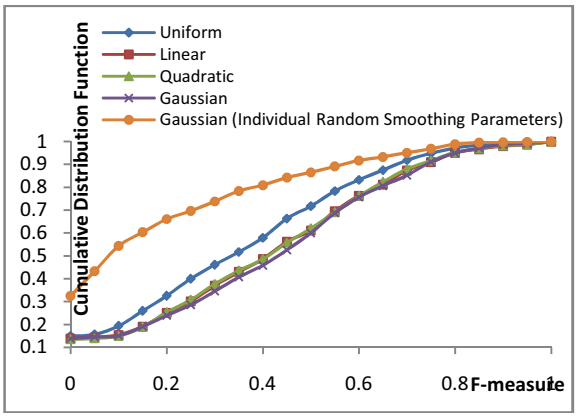

Figure 3: CDF of F-measures for several functional forms of the weighting function and for individual optimized smoothing parameters. We also plotted the cdf of the F-measures for random choice of the smoothing parameters.

2.2.3.3 *Choice of the distance*

To assess the choice of our distance, we compare the performance of our algorithm using the Jaccard distance and the Euclidian distance. Although computing an Euclidian distance is slightly faster than computing a Jaccard distance, the gain in performance is worth it: on Figure 4 we plotted the cdf for the Hamming and Jaccard algorithms with individual optimal smoothing parameters and a Gaussian weighting function.

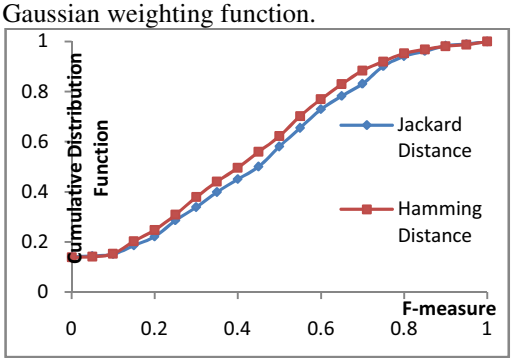

Figure 4: CDF of the F-measures of inferred attributes for the Hamming or Euclidian distance and the Jaccard-based distance.

2.2.3.4 *Choice of the observation*

The last parameter that may impact the quality of the inference is the observation set I. For the acoustic order, we would like to confirm the intuition that as $|I|$ increases, the overall performance converges towards an upper limit. On Figure 5 we plotted the cdf of the F-measures over all inferred attributes for several values of $|I|$. It can be seen that although there is an improvement of the performance for small values of $|I|$, there is a 'glass ceiling' for the classifier's performance.

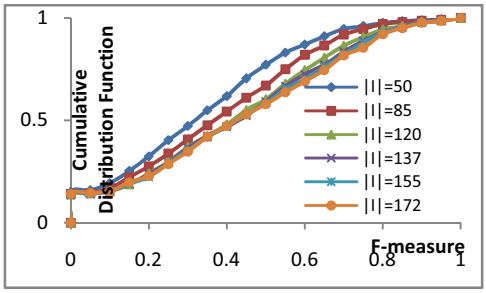

Figure 5: CDF of the F-measures of all inferred attributes for several values of the size of the observation I.

3. INVERSE INFERENCE

3.1. Description

As mentioned earlier, the observation is fixed when performing inverse inference so I is the set of modifiable attributes, meaning all low-level attributes and of which we can easily think of as attributes that a musician can easily change in a song: instruments, tempo, language, etc. Some other attributes are clearly not that easily modifiable like the "mood" of the song, or its "style". These more subjective attributes are supposed to "emerge" from the observation. The modifiable attributes set is of cardinality 232. So there are 2^{232} possible flips of the observation. To reduce this search space, we introduce the following heuristic: recall that there are categories in which all attributes can be grouped. For instance, all sorts of moods (calm, aggressive, dreamy, etc.) can be grouped in one category. In each category we define a limit to the number of true attributes. This limit is determined using the statistical distribution of the number of true attributes per category on the songs of the database. The limit we fix is the 90%-quantile of this distribution for each category. If the initial state a_I does not respect these limits once flipped, the flip will not be evaluated. We call a_I-*authorized flips* such flips and S_{a_I} the subset of authorized flips. The problem we will solve as an approximation of the general inverse inference problem we mentioned earlier is the following:

$$\sigma^* \in \operatorname{argmin}_{\sigma \in S_{a_I}} \left[d(\sigma) - \lambda \prod_{k \in T} \hat{p}_k(a_I)^{a_k} \left(1 - \hat{p}_k(a_I)\right)^{1-a_k} \right]$$

The product is just the estimated value of $\Pr(A_T = a_T | A_I = a_I)$.

```
Do
  flip = get next authorized flip
  If ( fitness(flip) is one of the M best
fitness's of order 1 )
  Then add flip to M best flip list
Until no more authorized flips of order 1
For k=2..max order Do
    flip = get next authorized flip
    If flip contains same attributes as at least
an element of M best flip list
    And fitness(flip) is one of the M best
    Then add flip to temporary M best flip list
  Until no more authorized flips of order k
  M best flip list = temporary M best flip list
End For
```

Table 1. The inverse inference algorithm.

We finally explore the search space with the following strategy: we scan through the a_I-authorized order 1 flips and evaluate their fitness; we select the M best authorized order 1 flips ($M$is a parameter of the approximation method). We then loop on the order of the flip, and only consider a_I-authorized flips that contain the same attributes as at least one of the M best flips of previous order, and update the M best flip list (Table 1).

3.2. Examples

We illustrate here inverse inference with an example. We tried to make "Black or White" by Michael Jackson sound 'Funk' (its style was initially tagged as 'Pop' and 'Rock/Pop'). We set tradeoff parameter $\lambda = 1000$ so that in the optimization process, an additional modification is only accepted if it increases the probability of being 'Funk' by at least 0.01. This 'Probability of Target' is the probability of being 'Funk' given the 232 modifiable attributes of the song "Black or White" (or its flipped version). Initially, this probability is about 0.05 and even though we are not sure the flip we found is optimal, it yields a probability of being 'Funk' of about 0.87 which is much better. The set of attributes to flip is the following:

Main Instruments Vocals (Choir)=false
Variant forming/shaping=true
Main Instruments Voice (Effects/Sample)=true
Main Instruments SFX (Sound Effects)=true
Rhythm funky=true

To confirm the funkiness of the song has been increased, we can verify that within the nearest neighbors (NN) of the modified song there are more 'Funk' songs than initially. Among the 20 NN of the initial "Black or White", 4 songs are tagged as "Funk". Among the 20 NN of its flipped version, 11 songs tagged as "Funk". So the algorithm is indeed able to translate the high-level query "more funk" into a minimal set of low-level modifications.

4. CONCLUSION

We have presented an attribute inference algorithm for large-scale Boolean databases. The algorithm produces optimally plausible inferences in reasonable time. The algorithm can also be used to perform inverse inference, i.e. answering questions about how to modify a description to make it fit optimally a given attribute. The algorithm is particularly well suited to the exploitation of metadata databases, both for traditional search applications, and for applications that exploit these databases for creation purposes. As such, it can be seen as a first step in turning these databases into knowledge bases.

5. ACKNOWLEDGEMENT

This work was partially supported by the TAGora project (contract IST-34721), funded by the Future and Emerging Technologies program of the European Commission.

6. REFERENCES

[1] Agrawal, R., & Srikant, R. (1994). Fast Algorithms for Mining Association Rules. *Proc. 20th Int. Conf. Very Large Data Bases (VLDB)*, pp. 487-499, Morgan Kaufmann.

[2] Aucouturier, J.-J., Pachet, F., Roy, P., & Beurivé, A. (2007). Signal + Context = Better Classification. *Proc. of ISMIR 07*, Vienna, Austria.

[3] Goldenberg, A., & Moore, A. (2004). Tractable Learning of Large Bayes Net Structures from Sparse Data. Proc. of the 21st int. conf. on Machine learning. New-York: ACM Int. Conf. Proc. Series.

[4] Nadaraya, E. (1964). On estimating regression. Theor. Probab. Appl., 9, 141-142.

[5] Pavlov, D., Mannila, H., & Smyth, P. (2001). Beyond independence: Probabilistic models for query approximation on binary transaction data. Technical report ICS TR-01-09, Information and Computer Science Department. UC Irvine.

[6] Wainwright, M., & Jordan, M. (2003). Graphical models, exponential families, and variational inference. Technical report, UC Berkeley, Department of Statistics, No. 649, September.

[7] Watson, G. (1964). Smooth regression analysis. Sankhya, Series A (26), 359-372.

[8] Winn, J. M., & Bishop, C. M. (2005). Variational Message Passing. Journal of Machine Learning, 6, 661-694.