Scalable Clustering and Keyword Suggestion for Online Advertisements

Anton Schwaighofer
Microsoft Research
7 JJ Thomson Ave
Cambridge CB3 0FB, UK
antonsc@microsoft.com

Joaquin Quiñonero Candela
Microsoft Research
7 JJ Thomson Ave
Cambridge CB3 0FB, UK
joaquinc@microsoft.com

Thomas Borchert
Microsoft Research
7 JJ Thomson Ave
Cambridge CB3 0FB, UK
tborcher@microsoft.com

Thore Graepel
Microsoft Research
7 JJ Thomson Ave
Cambridge CB3 0FB, UK
thoreg@microsoft.com

Ralf Herbrich
Microsoft Research
7 JJ Thomson Ave
Cambridge CB3 0FB, UK
rherb@microsoft.com

ABSTRACT

We present an efficient Bayesian online learning algorithm for clustering vectors of binary values based on a well known model, the mixture of Bernoulli profiles. The model includes conjugate Beta priors over the success probabilities and maintains discrete probability distributions for cluster assignments. Clustering is then formulated as inference in a factor graph which is solved efficiently using online approximate message passing. The resulting algorithm has three key features: a) it requires only a single pass across the data and can hence be used on data streams, b) it maintains the uncertainty of parameters and cluster assignments, and c) it implements an automatic step size adaptation based on the current model uncertainty. The model is tested on an artificially generated toy dataset and applied to a large scale real-world data set from online advertising, the data being online ads characterized by the set of keywords to which they have been subscribed. The proposed approach scales well for large datasets, and compares favorably to other clustering algorithms on the ads dataset. As a concrete application to online advertising we show how the learnt model can be used to recommend new keywords for given ads.

1. INTRODUCTION

Clustering data based on some notion of similarity is a problem that arises frequently in many data analysis tasks [3]. Our interest in clustering stems from the need to cluster online advertisements. Large online advertisers have repositories of ads available that subscribe to millions of different keywords to be matched to a given search query. When it comes to analyzing this data, it is useful to be able to group the individual data points into categories of related concepts. For example, advertisements could be grouped into categories such as automobiles, travel, financial services, and so on. Advertisers creating the ads are not required to specify which category an ad belongs to, instead they provide a set of keywords which describe the ad. An algorithm which can discover these categories and assign advertisements to them is therefore required in order to be able to explore the data in a structured way. In principle this categorization could be solved by a supervised classification scheme, but this would require manual labeling of a significant portion of the data, while an unsupervised clustering requires no labels at all. Furthermore, a supervised classifier would operate on a predefined and fixed set of possible labels, whereas unsupervised techniques are free to create whatever categories best fit the data. An unsupervised grouping thus seems beneficial for this problem.

In this paper we demonstrate a new way of clustering data that comes in the form of binary vectors. The method is particularly suitable for working on very large collections of ads. The aim is to develop an online clustering method that "touches" each data point (in our case, each ad) only once. Scaling behavior that goes beyond this bare minimum is too costly for the large corpora typical in web applications. Furthermore, the kind of data encountered in typical web applications is inherently ambiguous. Consider, for example, an ad about car insurances and the question of whether to assign it to the cluster of car related ads, or to the cluster of financial services ads. Hard assignments to clusters, be it during model learning or when assigning new data to clusters, will necessarily fail to capture such ambiguities, and hence probabilistic methods are called for.

The approach proposed here is based on a mixture of Bernoulli profiles (products of Bernoulli distributions) [10]. Traditionally the optimal value of the model parameters for mixture models is inferred by maximum likelihood [11], and a very popular technique is the expectation-maximization (EM) algorithm. A detailed treatment of the EM algorithm applied to mixtures of Bernoulli profiles can be found in [4,

ADKDD'09, June 28, 2009, Paris, France.
Copyright 2009 ACM 978-1-60558-671-7 ...$5.00.

Sect. 9.3]. Unfortunately, maximum likelihood learning is impractical for large scale datasets. Multiple passes through the entire dataset are required at a prohibitive computational cost. Additionally, inference by maximum likelihood requires a very careful initialization to avoid being trapped in local optima.

This work proposes using Bayesian inference [4]. Instead of estimating the point value of the model parameters that maximize the likelihood, the parameters of interest are treated as belief variables with associated distributions. Given the data, inference consists of computing the parameters' posterior distributions, which capture the uncertainty about their true values. The probabilistic nature of the underlying model has a number of advantages:

1. The quantification of uncertainty allows for a more careful interpretation of learnt parameter values.

2. Known model uncertainty can drive experimental design and active learning.

3. During online learning, the known uncertainty helps automatically adapt the effective learning rate for each parameter individually, and allows to control the memory consumption of the model by pruning non-informative parameters.

4. At any point data can be generated from the model by sampling.

The model is expressed using factor graphs, a convenient representation for factorizing probabilistic models. Inference is achieved by means of message passing [4, Chap. 8]. Message passing on factor graphs allows one to easily use an approximate "online" inference scheme. The datapoints are processed one by one, starting from an empty model, and only a single pass through the data is required. To be able to cope with very large datasets, several further computationally efficient approximations are proposed. For example, the posteriors for rare features (in the case of ads, this would be rarely used keywords) are represented by using shared parameters. Finally, the factor graph representation together with the "local" message passing lends itself to a straightforward paralellisation of inference across different subsets of the data.

The paper is organized as follows: The mixture of Bernoulli profiles model is described in Sect. 2. Bayesian inference with message passing on a factor graph is detailed in Sect. 3, as well as the online approximate inference scheme. Parallel inference is discussed in Sect. 4. Performance is evaluated in Sect. **??**. Finally, Sect. 6 explains how the model proposed can be used to suggest additional relevant keywords for ads to subscribe to.

2. PROBLEM SETTING AND MODEL

We consider a set of N objects, where the i-th object $\vec{x}_i$ is described by a D-dimensional vector of binary variables. In our concrete application, these objects are online ads in paid search, described by the set of keywords to which they subscribe. There are a total of D unique keywords, and vector $\vec{x}_i$ contains a 1 for those keywords that the i-th advertisement has subscribed to: If the i-th advertisement subscribed to the d-th keyword, then $x_{id} = 1$, else $x_{id} = 0$.

The model we propose assumes that the keyword vector of an ad is generated by one of K clusters, or mixture components. Each ad $\vec{x}_i$ has a variable $c_i \in \{1, \ldots, K\}$ associated with it that indicates the index of the cluster to which the ad belongs. If the i-th ad belongs to cluster j then $c_i = j$. Within a cluster, ads subscribe to keywords following *independent* Bernoulli probability distributions. If the i-th ad belongs to cluster j then the probability that it subscribes to the d-th keyword is given by $t_{jd} = p(x_{id} = 1|c_i = j)$. As a result, the probability that the i-th ad belongs to cluster j is given by a cluster-dependent Bernoulli profile:

$$p(\vec{x_i}|c_i = j) = \prod_{d=1}^{D} {t_{jd}}^{x_{id}} (1 - t_{jd})^{1-x_{id}}.$$

Which cluster an ad belongs to is unknown *a priori*, and that uncertainty is captured by the prior probability that the i-th ad (or in fact any other ad) belongs to cluster j: $\pi_j = p(c_i = j)$. If the global cluster assignment priors $\{\pi_j\}$ and the probabilities of subscribing to keywords $\{t_{jd}\}$ are known, the sampling distribution of the model is given by a mixture of Bernoulli profiles:

$$\begin{aligned} p(\vec{x_i}|\{t_{jd}\}, \{\pi_j\}) &= \sum_{j=1}^{K} p(c_i = j) \prod_{d=1}^{D} p(x_{id}|c_i = j, t_{jd}) \\ &= \sum_{j=1}^{K} \pi_j \prod_{d=1}^{D} {t_{jd}}^{x_{id}} (1 - t_{jd})^{1-x_{id}}. \quad (1) \end{aligned}$$

Sampling an ad from this model involves selecting first one of the K clusters by drawing it from a discrete distribution with parameter vector $\vec{\pi} = [\pi_1, \ldots, \pi_K]$. In a second step, keywords that the ad subscribes to are drawn from the selected cluster's Bernoulli profile.

The mixture of Bernoulli profiles is a well known model covered extensively in machine learning text books (for example, [4]). Typically the prior probabilities $\{\pi_j\}$ of belonging to a cluster and the probabilities of subscribing to the individual keywords $\{t_{jd}\}$ are treated as parameters of the model, and are estimated by maximum likelihood. Maximizing the likelihood is readily achieved by assuming the data is independently sampled from (1), and maximizing the resulting product of individual probabilities with respect to the parameters. Two common approaches are direct gradient-based maximization, or use of the Expectation Maximization (EM) algorithm. However, the maximum likelihood approaches suffers from a number of problems for the application we consider here:

- Both EM and direct gradient ascent are iterative algorithms and require several passes over the data in order to converge. Initialization is crucial due to the multiple modes of the likelihood, but very difficult for the high dimensional binary data we consider here.

- The optimization results in point estimates of the parameters $\{\pi_j\}$ and $\{t_{jd}\}$. Thus, no notion of uncertainty about the learned model is available. In a maximum likelihood framework, a value of 0.5 for a keyword probability can indicate that the keyword was present in 1 out of 2 ads, or in 5,000 out of 10,000.

2.1 Related Models

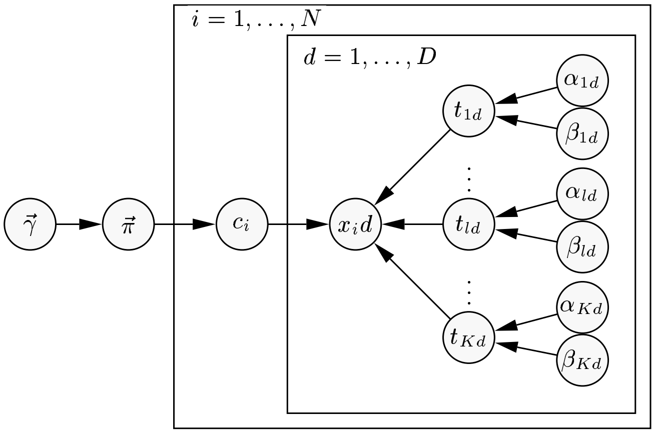


Figure 1: A directed graphical model representation of the Bayesian mixture of Bernoulli profiles.

Latent Dirichlet Allocation (LDA, [5]) is an unsupervised model that has been developed to model text corpora. LDA shares with the model presented here the fact that they both are unsupervised. Topics in an LDA model roughly correspond to the clusters in the model proposed above. The generative process, however, is quite different. In LDA, a new topic (cluster) is chosen each time before a word (here: keyword subscription) is chosen. A single word is subsequently sampled from a multinomial distribution that depends on the topic. In the clustering model described above, a cluster is chosen, after which all keyword subscriptions are sampled from the cluster's Bernoulli profile.

3. A BAYESIAN TREATMENT

An alternative approach to maximum likelihood is Bayesian inference. Rather than treating the unknown variables as model parameters and learning their optimal value, in the Bayesian framework these unknown variables are treated as belief variables, and beliefs about their values are represented by probability distributions to explicitly account for uncertainty. Before seeing any data, prior distributions can either be uninformative or encode prior knowledge about the problem domain. Inference reduces to using Bayes' rule given the prior distributions and the likelihood (1) to obtain the *posterior* distributions of the variables of interest.

For the mixture of Bernoulli profiles presented here, the Bernoulli probabilities of keyword subscription are given conjugate priors, which are Beta distributions $t \sim \text{Beta}(t; \alpha, \beta)$. The parameters α and β can be interpreted as pseudo-counts: α as the number of times the keyword was subscribed to and β as the number of times the keyword was not subscribed to. The probability density function (PDF) of the keyword subscription probability t is

$$p(t) = \text{Beta}(t; \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} t^{\alpha-1}(1-t)^{\beta-1}.$$

Figure 3(a) shows two examples of the Beta PDF for different choices of the parameters α and β. The higher the sum of the pseudo-counts, the smaller the uncertainty about the value of t.

The other unknown variables of interest are the prior cluster probabilities $\{\pi_j\}$; these are given a Dirichlet prior distribution, $\vec{\pi} \sim \text{Dir}(\vec{\pi}|\vec{\gamma})$ with parameter vector $\vec{\gamma}$. Similar to the Beta distribution, γ_j can be interpreted as a pseudo-count of the number of ads that belong to cluster j.

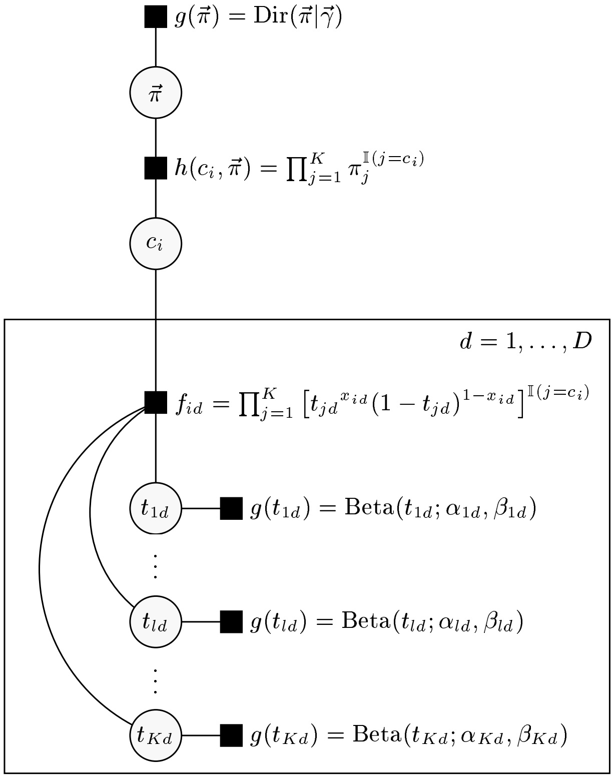

$g(\vec{\pi}) = \text{Dir}(\vec{\pi}|\vec{\gamma})$
$\vec{\pi}$
$h(c_i, \vec{\pi}) = \prod_{j=1}^{K} \pi_j^{\mathbb{I}(j=c_i)}$
c_i
$d = 1, \ldots, D$
$f_{id} = \prod_{j=1}^{K} \left[t_{jd}{}^{x_{id}} (1 - t_{jd})^{1-x_{id}} \right]^{\mathbb{I}(j=c_i)}$
t_{1d}
$g(t_{1d}) = \text{Beta}(t_{1d}; \alpha_{1d}, \beta_{1d})$
t_{ld}
$g(t_{ld}) = \text{Beta}(t_{ld}; \alpha_{ld}, \beta_{ld})$
t_{Kd}
$g(t_{Kd}) = \text{Beta}(t_{Kd}; \alpha_{Kd}, \beta_{Kd})$

Figure 2: The Bayesian mixture of Bernoulli profiles model represented as a factor graph, for the i-th training example $\vec{x}_i$.

Fig. 1 shows the directed graphical model corresponding to the full Bayesian model, including the parameters of the Beta and Dirichlet distributions. The parts of the graph enclosed in plates are replicated according to the index in the plate. For example, for a fixed value of i in the outer plate, the inner plate is replicated D times, once for each value of the keyword index d. The arrows indicate the dependencies between variables (see [4, Chapter 8] for a treatment of directed graphical models). The graph representation has the advantage of clearly revealing conditional independence between variables, which is important for computing the marginal posterior distributions efficiently. Fig. 2 shows the factor graph representation of a slice of the directed graph in Fig. 1 for a single datapoint indexed by i. Factor graphs [9] are bipartite graphs that represent joint probability distributions by means of variable nodes (circles) connected to factor nodes (shaded squares). Factor nodes express the functional relation among the variables connected to them, and the product of all factors corresponds to the joint probability distribution [7, 9]. Marginal distributions are obtained by computing *messages* from factor nodes to variable nodes: the marginal distribution of any given variable node is the product of its incoming messages. Inference in factor graphs is thus known as *message passing*, a detailed account of which is given in [4, Chapter 8]. The representation in

Fig. 2 absorbs the observed variables $x_{id},\ d = 1, \ldots, D$ into the factors f_{id}. The marginals of the cluster assignment probabilities $\vec{\pi}$ and of the keyword subscription probabilities t_{jd} obtained by message passing are thus the posterior distributions desired.

3.1 Online Learning

The factor graph in Fig. 2 represents only a single ad, but already contains on the order of $D \times K$ variables, with the number of keywords D potentially in the millions,[1] and the number of clusters K in the hundreds. The full graph further replicates this slice N times (number of training data), with N in the tens of millions. It is clearly impossible to store a graph that size in memory, or to compute and store the necessary messages.

To make the inference practical, we opt for an online learning scheme based on approximate inference with Assumed Density Filtering (ADF) [13]. Data points (ads) are processed one at a time, and the posterior distributions of $\vec{\pi}$ and t_{jd} obtained after processing one data point are passed as prior distributions for processing the next data point.

Because the factor graph is a *tree* in this online learning scenario, messages only need to be computed once from a root node to the leaves and back. A practical schedule for processing the i-th data point is the following:

1. Set the prior distributions $g(t_{ld})$ and $g(\vec{\pi})$ to the posterior marginals on t_{jd} and $\vec{\pi}$ obtained from processing the previous datapoint.
2. Compute the messages $\{m_{f_{id} \to c_i}(c_i)\}_{d=1}^{D}$ from the keyword factors f_{id} to the cluster assignment variable c_i.
3. Compute the message $m_{h \to \vec{\pi}}(\vec{\pi})$ from the cluster assignment factor $h(c_i, \vec{\pi})$ to the cluster assignment probability variable $\vec{\pi}$.
4. Compute the message $m_{h \to c_i}(c_i)$.
5. For each keyword factor f_{id} compute the outgoing messages $\{m_{f_{id} \to t_{ld}}(t_{ld})\}_{d=1}^{D}$.
6. Compute the new marginals $\{p(t_{ld}|\vec{x}_i)\}_{d=1}^{D}$ and $p(\vec{\pi})$.

Note that no messages need to be stored between the ADF steps, but only on the order of $D \times K$ marginal distributions.

The message from f_{id} to c_i is given by

$$m_{f_{id} \to c_i}(c_i) \quad = \quad \prod_{j=1}^{K} \left[{\mu_{jd}}^{x_{id}} (1 - \mu_{jd})^{1 - x_{id}} \right]^{\mathbb{I}(c_i = j)} \ , \quad (2)$$

where $\mu_{jd} = \frac{\alpha_{jd}}{\alpha_{jd} + \beta_{jd}}$ is the mean of $g(t_{ld})$, and $\mathbb{I}(\cdot)$ is the indicator function, equal to 1 if its argument is true, and to 0 if it is false. The message from c_i to factor h is simply $m_{c_i \to h}(c_i) = \prod_{d=1}^{D} m_{f_{id} \to c_i}(c_i)$, and therefore the message from factor h to $\vec{\pi}$ is

$$m_{h \to \vec{\pi}}(\vec{\pi}) = \sum_{l=1}^{K} \pi_l \prod_{d=1}^{D} {\mu_{ld}}^{x_{id}} (1 - \mu_{ld})^{1 - x_{id}} \ .$$

The message from h to c_i basically sends the (scaled) average cluster assignment probabilities under the Dirichlet prior $g(\vec{\pi})$

$$m_{h \to c_i}(c_i) = \prod_{j=1}^{K} \gamma_j^{\mathbb{I}(c_i = j)} \ .$$

It is useful to compute as an intermediate step the marginal distribution of c_i, given by the normalized product of its incoming messages. We adopt the shorthand

$$r_{il} = p(c_i = l | \vec{x}_i) = \frac{\gamma_l \ \prod_{d=1}^{D} {\mu_{ld}}^{x_{id}} (1 - \mu_{ld})^{1 - x_{id}}}{\sum_{j=1}^{K} \gamma_j \ \prod_{d=1}^{D} {\mu_{jd}}^{x_{id}} (1 - \mu_{jd})^{1 - x_{id}}} \ , \quad (3)$$

and refer to it as the *responsibility* of cluster l for advertisement i, with $0 \leq r_{il} \leq 1$ and $\sum_{j=1}^{K} r_{ij} = 1$.

The details of the computation of the message from f_{id} to t_{ld} are relegated to the appendix. Scaled appropriately, the message itself can be written as the linear combination of a Bernoulli distribution in t_{ld} and a uniform distribution:

$$m_{f_{id} \to t_{ld}}(t_{ld}) = r_{il} \frac{{t_{ld}}^{x_{id}} (1 - t_{ld})^{1 - x_{id}}}{{\mu_{ld}}^{x_{id}} (1 - \mu_{ld})^{1 - x_{id}}} + (1 - r_{il}) \ . \quad (4)$$

3.2 Beta and Dirichlet Approximations

Message passing is only efficient if a compact message representation can be assumed. Maintaining such a representation may require projecting the true message to a family of distributions thereby approximating it in the spirit of expectation propagation [13]. Given that the message (4) from f_{id} to the t_{ld} nodes is a mixture of a Beta distribution with a uniform distribution, the marginal distribution of t_{ld} is therefore not a Beta distribution either,

$$\begin{aligned} p(t_{ld}) \quad & \propto \quad m_{f_{id} \to t_{ld}}(t_{ld}) \cdot m_{g_{ld} \to t_{ld}}(t_{ld}) \\ & = \quad r_{il} \ \mathrm{Beta}\left(t_{ld}; \alpha_{ld} + x_{id}, \beta_{ld} + (1 - x_{id})\right) \quad (5) \\ & \qquad + (1 - r_{il}) \ \mathrm{Beta}\left(t_{jd}; \alpha_{jd}, \beta_{jd}\right) \ . \end{aligned}$$

Instead, it is the *convex combination* of the prior and the posterior Beta distributions on t_{ld} under the assumption that the current advertisement belongs to cluster l. The posterior has larger weight the larger the responsibility of cluster l.

In order to keep the message $m_{t_{ld} \to f_{id}}(t_{ld})$ in the Beta family, the marginal $p(t_{ld})$ itself is projected onto a Beta distribution by moment matching. For the first order moment of the marginal, we obtain

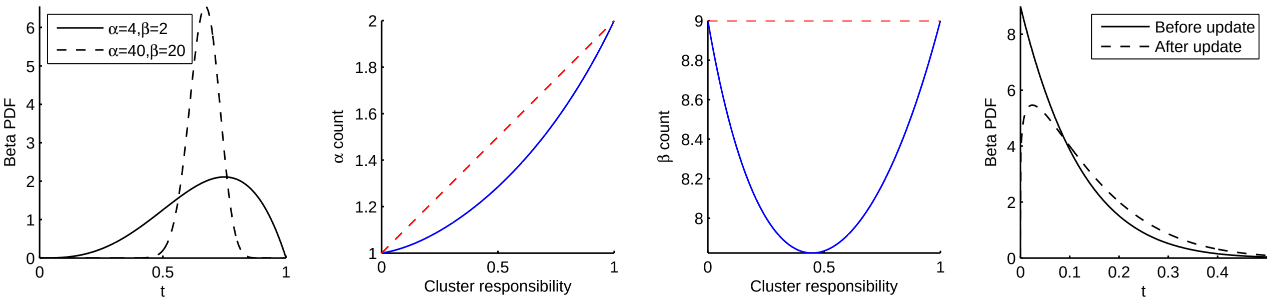
$$M_1(x_{id}) = r_{il} \frac{\alpha_{ld} + x_{id}}{\alpha_{ld} + \beta_{ld} + 1} + (1 - r_{il}) \frac{\alpha_{ld}}{\alpha_{ld} + \beta_{ld}} \ ,$$

and for the second non-central moment,

$$\begin{aligned} M_2(x_{id}) \quad = \quad & r_{il} \frac{(\alpha_{ld} + x_{id})(\alpha_{ld} + x_{id} + 1)}{(\alpha_{ld} + \beta_{ld} + 1)(\alpha_{ld} + \beta_{ld} + 2)} \\ & + (1 - r_{il}) \frac{\alpha_{ld}(\alpha_{ld} + 1)}{(\alpha_{ld} + \beta_{ld})(\alpha_{ld} + \beta_{ld} + 1)} \ . \end{aligned}$$

Note that the first order moment, i.e., the mean of the marginal, is a convex combination of the prior mean and the posterior mean under a full update of the Beta distribution (without taking the responsibility term r_{il} into account). Using the expressions of the parameters of a Beta

[1] Most keywords are actually key phrases, akin to typical search engine queries, which is why D can become so large.



(a) (b) (c) (d)

Figure 3: (a) Examples of Beta distributions with different parameters α and β. (b) through (d): An example of the effect of moment matching when updating a Beta distribution with $\alpha = 1$ and $\beta = 9$: Plot (b) shows the updated α as a function of the responsibility r, (c) shows the updated β as a function of r. (d) plots the Beta PDF before and after an update with $r \approx 0.4$ (leading to maximum loss of pseudo-count). Note that the variance of the updated distribution is larger than before the update.

distribution in terms of its moments, the parameters of the approximating Beta are computed as

$$\tilde{\alpha}_{ld} = M_1(x_{id})\tilde{N} \quad \text{and} \quad \tilde{\beta}_{ld} = [1 - M_1(x_{id})]\tilde{N} \ ,$$

where

$$\tilde{N} = \frac{M_1(x_{id}) - M_2(x_{id})}{M_2(x_{id}) - M_1(x_{id})^2} = \tilde{\alpha}_{ld} + \tilde{\beta}_{ld}$$

is the updated pseudo-count (including pseudo-count coming from the prior), roughly the total number of observed ads.

It is important to mention here that, due to moment matching, the updates may lead to a loss of pseudo-count. When learning a single Beta distribution, the total amount of pseudo-counts ($\alpha + \beta$) can never decrease, thus the variance of the Beta distribution can only shrink when observing more and more data. The effect of "forgetting" pseudo-counts is illustrated in 3(b)-(d).

The exact marginal distribution of $\vec{\pi}$ turns out to be a mixture of Dirichlet distributions,

$$p(\vec{\pi}) = \sum_{l=1}^{L} r_{il} \mathrm{Dir}(\vec{\pi}|\vec{\gamma} + \vec{e}_l) \ ,$$

where $\vec{e}_l$ is the i-th unit vector of length K. There is one Dirichlet mixture per cluster, and its value is the result of assuming that the corresponding cluster is fully responsible for the ad visited. The mixing coefficients are the actual responsibilities that the clusters had for the ad. Here again we need to take an approximation to stay in the family of Dirichlet distributions. We chose to preserve the means, and ensure that the sum of the γ_j is increased by one. This can be achieved by simply adding the cluster responsibilities to the corresponding parameters of the Dirichlet distribution, $\gamma_j^{\text{new}} = \gamma_j + r_{ij}$.

4. SCALING DETAILS

As described in Sec. 3.1, using ADF to process a single data point at a time leads to large savings in terms of computation time and memory use. Even within this online learning framework, clustering large datasets is computationally demanding. A typical dataset can contain millions of advertisements with millions of unique keywords. If every cluster contained one Beta distribution for every possible keyword then the memory requirements would be on the order of hundreds of gigabytes. In addition, the computation of (3) for each advertisement would involve tens of millions of terms, which would make training extremely slow. Several steps need to be taken to ensure that the model can run in a reasonable amount of time and use a reasonable amount of memory.

4.1 Sparse Representation

While there are potentially millions of unique keywords in a dataset, individual advertisements are very sparse, typically subscribing to on the order of ten keywords each. If one assumes that a cluster of similar ads should also be sparse, then that property can be exploited by using a sparse representation for the clusters. In this representation, only keywords that are "important" to a cluster are represented by explicit Beta distributions, and all other keywords are represented by the same single "default" Beta distribution for that cluster. "Important" here is a combination of 1) being contained in a significant number of the ads in the cluster, and 2) being sufficiently discriminative for that cluster. If every cluster contains hundreds of unique distributions instead of millions then the model will use a small amount memory, and computation of equation (3) can be done quickly.

Several steps are taken to ensure that the model remains sparse. First, in regular intervals, keywords are culled from the model using two different criteria:

- Keywords that have a similar probability (mean of the associated Beta distribution) across all clusters are irrelevant for distinguishing between clusters and can be removed (replaced in each cluster by the default value).
- Within a cluster, if replacing a keyword with the default value does not significantly change the responsibility profile (as measured using Kullback-Leibler divergence) then it can be removed.

Second, also in regular intervals, any clusters that explain only a tiny fraction of the data (i.e. have a small γ_i) are removed from the model. The data contained in any of these clusters is not discarded; instead it is treated like an advertisement (albeit one with fractional keyword subscriptions which are the mean of the keyword Beta distributions for the cluster to be removed) and applied to the model.

4.2 Parallelizing across Data

One strength of this clustering model is that it can be parallelized across data relatively easily. This parallelization is an extension of the factor graph model shown in Fig. 2, and is performed in four steps:

1. Given a prior model state, which could be a previously trained model or an empty model, an "equality" factor is used that creates multiple copies of the prior model.
2. Each of these child copies is trained in parallel using a different subset of the data.
3. After a child copy is finished training, we can compute the delta between the prior model and the child copy by dividing the child's posterior distribution by the prior distribution. This delta is a message that tells the prior how to update itself to be equal to the child's posterior.
4. All of these messages from the separate children are applied to the prior, giving a posterior distribution that contains all of the information learned by the parallel-trained copies.

This scheme is problematic because the model is extremely multi-modal. There is no guarantee that cluster i in one copy will describe the same natural cluster as cluster i in another copy, which would mean that step 4 would attempt to combine information from two disparate clusters into a single cluster. We take two steps to combat this problem. First, before training in parallel, the model is trained serially on a subset of the data form priming. This gives the clusters some initial definition before the parallel step, and reduces the freedom of the parallel copies to settle on different modes. Second, the full dataset is split into multiple batches. Parallel training is done one batch at a time, and after each batch the posterior produced in step 4 is used as the prior in step 1. This ensures that multiple copies of a single cluster cannot drift too far apart during the parallel training phase.

5. EVALUATION

The algorithm is evaluated using two datasets. The first is synthetic with known cluster assignments: 10,000 advertisements are sampled from a randomly generated model with 10 clusters whose prior probabilities are a random sample from uniform multipoint distribution, and 100 keywords whose Bernoulli probabilities are independently sampled from a uniform distribution.

The second dataset is derived from a corpus of almost 6 million advertisements and 19 million distinct keywords. The average advertisement subscribes to approximately 28 keywords. One can think of the dataset as a bipartite graph connecting ads to keywords, and groups of well-connected nodes in the graph would correspond to clusters of similar keywords and advertisements. The challenge is then to find these groups of well-connected nodes in the graph.

To get a feeling for the complexity of the problem, it is useful to analyze the connected components of the advertisement-keyword graph. For our dataset, this graph has one large component which contains 88% of the advertisements, with the remaining advertisements split into 388,000 tiny disjoint sub-graphs. 273,000 of these disjoint sub-graphs contain a single advertisement with only one keyword. In addition, some individual advertisements subscribe to huge numbers of keywords, sometimes as many as one million keywords for a single advertisement. These advertisements can help create strong connections between clusters that are undesirable. The large number of disjoint subsets and the over-subscribed advertisements combine to add enough noise to the input data that it would be difficult to find a good clustering solution.

We chose to apply some filters for on that dataset for two reasons: By its structure, we do not expect to obtain meaningful clustering solutions with any clustering method. Secondly, the dataset is of a size that we could easily handle using the online clustering method presented in this paper, but not with any other clustering method without substantial amounts of engineering. In order to obtain a more reasonable dataset, we first remove all keywords whose total number of subscribed ads is below a certain threshold t_a. Second, we remove all advertisements whose keyword subscription count is above a certain threshold t_k. This culling retains the most used keywords and can also significantly reduce the size of the input dataset, which has the added benefit of speeding up the training process.

Using thresholds of $t_a = 100$ and $t_k = 500$ leaves about 207,000 advertisements and 2,000 unique keywords. The keyword-advertisement graph for this dataset contains only a single connected component, and is of a size that can be handled using k-means or EM clustering.

We compare the proposed Bayesian clustering model with several other clustering methods: k-means, agglomerative clustering, and a maximum likelihood (ML) version of the inference for the mixture of Bernoulli profiles based on expectation-maximization (EM). Details about these algorithms can be found in the excellent review paper [3]. Comparing unsupervised clustering models is intrinsically difficult, because there is no ground truth from which we can measure the predictive ability of the model. The most straightforward comparison is to visually inspect the clusters, which is possible here because the items being clustered – Internet search keywords – have meanings that we can understand.

5.1 Qualitative Comparison and Training Time

Using the advertisement derived dataset, we visually inspect the resulting clusters for consistency in the meanings of the most prominent keywords. The results are shown in Table 1. Qualitatively, k-means and agglomerative clustering suffer from a collapse of most of the ads into a single cluster. This can be caused by the spurious connections between clusters introduced by ads that subscribe to incoherent sets of keywords. Both the Bayesian and ML mixture of Bernoulli profile models attain qualitatively better results, managing to identify many more meaningful clusters and spreading the ads more evenly across these.

We compare the training times of the four models on this dataset. k-means and agglomerative clustering both take approximately three hours to train. Because it requires visiting the whole dataset many times, ML inference with the EM algorithm is computationally very intense and takes 40 hours to train. The Bayesian mixture model using ADF that we propose in this paper trains in only 1 hour.

Table 1: Training time and subjective quality assessment of clustering methods on a dataset of 207,000 ads, when requiring all methods to create 100 clusters.

Method	Training time	Clustering
k-means	3h	90% of ads in one cluster. Remaining clusters are consistent.
Agglomerative	3.5h	90% of ads in one cluster. Remaining clusters are consistent.
ML inference with EM	40h	Ads evenly spread. Most clusters are consistent, some are mixtures of topics.
Bayesian inference	1h	Ads are evenly spread. Almost all clusters are consistent, few are mixtures of topics.

A larger dataset was generated from the corpus of advertisements using $t_a = 100$ and $t_k = 100$, which yields 1.3 million advertisements and 73,000 unique keywords. A parallel implementation (Sect. 4.2) of the Bayesian mixture model we propose in this paper takes seven hours to train on this larger dataset. However, none of the other benchmark methods had finished training after 3 days, hence we can not provide any performance comparisons on that large data set.

5.2 Quantitative Evaluation

For the synthetic dataset, we test the abibility of each of the clustering algorithms to identify whether two advertisements belong to the same cluster or not. We find that this evaluation criterion is most easy to interpret, can be computed for all clustering methods, and is closest to an actual application where our goal is indeed to use the clustering model to assign ads to categories.Every pair of advertisements is classified by the competing algorithms as belonging to the same cluster or to different clusters. For the probabilistic models, we compute for each pair of advertisements x_i and x_j the probability that they belong to the same cluster:

$$p(c_i = c_j | x_i, x_j) = \sum_{l=1}^{K} p(c_i = l | x_i) p(c_j = l | x_j) \,,$$

given by the dot product of the resposibility vectors (3). Naturally one would classify the pair of advertisements as belonging to the same cluster if $p(c_i = c_j | x_i, x_j) > 0.5$, but in the experiments we explore a variety of different thresholds in the [0.1, 0.9] range. We compute a true positive ratio (fraction of the pairs correctly classified as belonging to the same cluster) and a false positive ratio (fraction of the pairs incorrectly classified as belonging to the same cluster), and plot them against each other. For the probabilistic methods varying the threshold allows us to obtain an ROC curve [12]. For the non-probabilistic k-means and agglomerative clustering we obtain only a single point. The results are displayed in Figure 4. Agglomerative clustering and k-means have a true positive rate of 93.0% and 94.3% respectively, and both have a false positive ration of 2.28%. With a threshold of 50%, the ML mixture model has a true postive ratio of 97.4% and a false positive ratio of 1.67%, while the Bayesian mixture model has a true positive ratio of 99.5% and a false positive ratio of 1.66%. As can be seen in the figure, the Bayesian mixture model can consistently match the true positive ratio of the ML model with fewer false positives.

Figure 4: Evaluation on synthetic dataset. True positive ratio (fraction of the pairs correctly classified as belonging to the same cluster) versus false positive ratio (fraction of the pairs incorrectly classified as belonging to the same cluster)

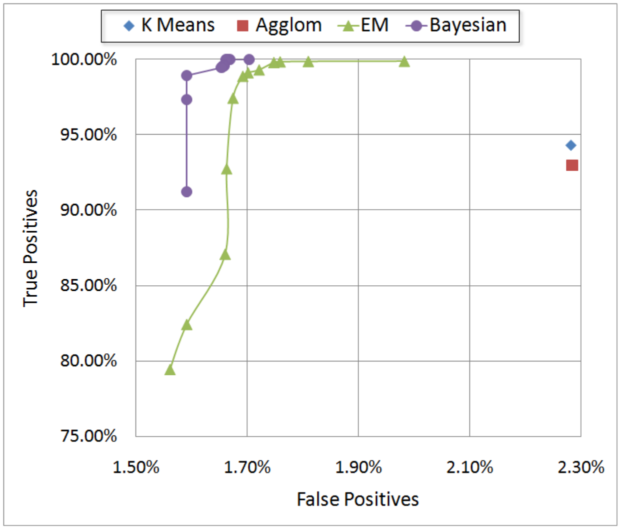


Table 2: Quantitative comparison based on the test negative log likelihood, and on the cluster and advertiser entropy scores when learning a 100 cluster model with all methods. Smaller numbers are better for all metrics.

	neg avg log likelihood	advertiser entropy score
Bayesian inference	17.97	1.18
Bayesian inference (discarding clutter)		0.96
ML inference with EM	12.98	2.61

The advertisement-based dataset does not offer a ground truth. For this reason, the evaluation is performed according to the following two metrics: the average negative log likelihood of the test set (closely related to the log perplexity, a quality criterion that has been used to evaluate, for example, the LDA model in [5]), and the advertiser entropy score. The second metric is based on the assumption that advertisements from a single advertiser most likely relate to the same concept, and thus should belong to as few clusters as possible. The advertiser entropy score measures the entropy of the distribution of advertisers across clusters. It is defined as:

$$S_A = \frac{1}{N}\sum_a N_a H(\vec{p}_a)\,,$$

where a is an advertiser index, N_a is the number of advertisements that belong to advertiser a, vector $\vec{p}_a$ contains the empirical probabilities of an advertisement from advertiser a of belonging to the different K clusters, and $H(\cdot)$ is the entropy function. By definition, a good clustering solution should achieve a low advertiser entropy score. Both of these metrics require probabilistic cluster assignments and are therefore not suitable for evaluating k-means and agglomerative clustering.

As shown in Table 2, a better average test log likelihood is achieved by the EM algorithm, which is probably attributable to the fact that it performs several passes through the training data, and does not enforce sparsity in its cluster representations. The advertiser entropy score is best for the Bayesian inference approach than for EM. This means that on average advertisers are spread across fewer clusters. A further improvement can be obtained from the Bayesian approach by letting the model learn a "clutter cluster". This cluster tracks an identical Beta distribution for all keyword subscription probabilities. As a result, this cluster tends to attract advertisements that subscribe to unrelated keywords and those that do not fit in any of the other clusters. This ultimately has the effect that these remaining clusters are more coherent, which explains the superior advertiser entropy score that can be obtained when discarding the clutter cluster.

5.3 Choosing the Number of Clusters

In the previous sections, our main goal was to benchmark methods, and we thus required all methods to learn the same number of clusters. In a practical application that uses the learned cluster model for, e.g., categorization, choosing the right number of clusters would be a necessary next step. This topic is extensively covered in the clustering literature, see [3] for pointers. However, we mainly use the clustering model for keyword suggestion, where it turned out that the number of clusters is rather uncritical, see the discussion below in sec. 6.

5.4 Benefits of Modeling Uncertainty

The topic of a cluster is determined by examining the keywords that have the largest probability of being subscribed to. Because of the noisy nature of the data, it is possible for certain unrelated keywords to spuriously have a high average subscription probability. These keywords might have been subscribed to by noisy ads that also simultaneously subscribe to some of the main thematic keywords of the cluster. The Bayesian treatment proposed allows one to deal with this problem by providing a measure of the uncertainty about the subscription probabilities. Table 3 shows an example of a very homogeneous cluster where the keyword with highest mean subscription probability μ – "pest control" – does not fit. However, this keyword was seen active in fewer ads attributed to this cluster. The total pseudo-count α of the Beta distribution represents the effective number of ads that were attributed to this cluster and subscribed to the keyword in question. Given two keywords with identical mean μ but with different α values, the model is more certain about the keyword with highest α. Sorting by α instead of by μ thus takes into account the uncertainty, and in Table 3 the benefits are evident: the spurious keyword is relegated to a lower position.

Table 3: Most prominent keywords in two different clusters for the Bayesian approach. Sorting by expected keyword subscription probability μ can place spurious keywords on top of the list. Sorting by the effective number of ads that subscribe to that keyword (parameter α of the Beta distribution) factors in the uncertainty and allows to get rid of the noisy keyword.

Sorting by mean:

Keyword	**Mean (μ)**	Alpha (α)	Beta (β)
pest control	**0.113**	44	343
nissan altima	**0.074**	84	1039
nissan maxima	**0.065**	75	1080
nissan quest	**0.065**	76	1090
nissan dealer	**0.051**	61	1136

Sorting by positive ad pseudo-count (α):

Keyword	Mean (μ)	**Alpha (α)**	Beta (β)
nissan altima	0.074	**84**	1039
nissan quest	0.065	**76**	1090
nissan maxima	0.065	**75**	1080
nissan dealer	0.051	**61**	1136
pest control	0.113	**44**	343

Table 4: Illustration of additional suggested keywords for an advertisement.

Subscribed Keywords	Suggested Keywords
window cleaner window cleaning cleaning company cleaning companies	carpet cleaning services home cleaning services floor cleaning services residential cleaning services commercial cleaning commercial cleaning services apartment cleaning services office cleaning services office cleaning cleaning tips

6. KEYWORD SUGGESTION

In our specific application, we are interested in *keyword suggestion*. The goal is to suggest to an advertiser a range of keywords that are semantically similar to ones that were already selected, in order to increase the reach of the ad. This is a challenging and commercially important task as pointed out by [8], and methods using semantic similarity [1] and concept hierarchies [6] as well as logistic regression and collaborative filtering [2] have been proposed. Our approach is similar to the latter in that makes keyword suggestions to one advertiser based on keyword subscriptions of other advertisers

The model described in Sect. 2 can be used in a generative form, following the directed graphical model shown in Fig. 1. For keyword suggestion, we assume that a specific ad represents partially observed data: An advertiser may have put some thoughts into which keywords to subscribe to, but still may have missed out on some important ones. Subscribed keywords thus act as an indicator of the advertiser's intent, but the (huge) set of non-subscribed keywords is treated as "not observed".

With this partially observed data, we can again perform message passing, in order to compute the probability of the unobserved keywords, given the subscribed keywords. In short, this works as follows: Let $S \subset \{1, \ldots, D\}$ be the set of all subscribed keywords in the i-th ad. All factors $\{f_{id}\}$, $d \in S$, send messages of the form (2) to node c_i, where it is combined with the incoming message from factor h. Similar to the update scenario in (3), a responsibility of clusters for the ad is computed, but this information is only based on the keywords that are actually subscribed:

$$\tilde{r}_{il} = p(c_i = l|\vec{x}_i) = \frac{\gamma_l \ \prod_{d \in S} {\mu_{ld}}^{x_{id}} (1 - \mu_{ld})^{1 - x_{id}}}{\sum_{j=1}^{K} \gamma_j \ \prod_{d \in S} {\mu_{jd}}^{x_{id}} (1 - \mu_{jd})^{1 - x_{id}}} \,. \tag{6}$$

As the last step, we can compute the expectation of the data (keyword) nodes that are implicitly attached to the factors f_{id} in Fig. 2, and obtain for the unobserved keywords $d \notin S$

$$p(x_{id} = 1|\{x_{ib}\}_{b \in S}) = \sum_{j=1}^{K} \tilde{r}_{ij} \mu_{jd} \,,$$

a linear combination of the Bernoulli profiles for the unobserved keywords, with weights based on the responsibilities computed from the observed keywords.

Using this method, keywords can be suggested to users with a clear ranking criterion (the above probability). Table 3 shows the additional keywords that are suggested for a specific advertisement related to cleaning services. The top 10 keywords with highest conditional subscription probability (given the existing subscribed keywords) are shown in the table.

Furthermore, the keyword suggestion can be refined by an iterative interactive process: from the list of suggestions, users can select keywords (or mark keywords as "I don't want to subscribe to that"). This updated information can be used to refine the computation of responsibilities in 6, and present the user a refined list of keywords.

7. DISCUSSION

We have proposed a Bayesian online clustering model for large datasets of binary vectors based on a mixture of Bernoulli profiles. The experiments conducted show that compared to the maximum likelihood treatment, the Bayesian approach proposed is both more accurate and dramatically faster to train. Whereas the maximum likelihood model is trained by EM in batch mode and requires several passes over the data, our approach uses an online training scheme requiring a single pass, and is suitable for streams of data. In contrast to the EM algorithm, training is incremental: adding one more data point does not require retraining the entire model.

Our approach is not only faster to train than k-means and agglomerative clustering; it also offers probabilistic cluster assignments and explicitly models the uncertainty about the learned model parameters, one advantage of which is resistance to noise. A straightforward application of our generative probabilistic model is the suggestion of additional keywords for advertisements.

Future work on this topic will benchmark the keyword suggestion algorithm described above to recommender systems. In particular, we plan to evaluate the performance of the Bayesian recommender system MatchBox [14] for keyword suggestion.

8. ACKNOWLEDGMENTS

All of the code used for this work was written in F#: we thank its creators at Microsoft Research Cambridge, UK, for their support. We thank Microsoft adCenter Labs for their continuing sponsorship and support, and the anonymous reviewers for their comments that helped to improve this paper.

9. REFERENCES

[1] V. Abhishek and K. Hosanagar. Keyword generation for search engine advertising using semantic similarity between terms. In *ICEC '07: Proceedings of the ninth international conference on Electronic commerce*, pages 89–94, New York, NY, USA, 2007. ACM.

[2] K. Bartz, V. Murthi, and S. Sebastian. Logistic regression and collaborative filtering for sponsored search term recommendation. In *Proceedings of the Second Workshop on Sponsored Search Auctions*, 2006.

[3] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[4] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[5] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[6] Y. Chen, G.-R. Xue, and Y. Yu. Advertising keyword suggestion based on concept hierarchy. In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, pages 251–260, New York, NY, USA, 2008. ACM.

[7] B.J. Frey, F.R. Kschischang, H.A. Loeliger, and N. Wiberg. Factor Graphs and Algorithms. In *Proceedings of the annual Allerton Conference on Communication, Control and Computing*, volume 35, pages 666–680, 1997.

[8] A. Joshi and R. Motwani. Keyword generation for search engine advertising. In *ICDMW '06: Proceedings of the Sixth IEEE International Conference on Data Mining - Workshops*, pages 490–496, Washington, DC, USA, 2006. IEEE Computer Society.

[9] F.R. Kschischang, B.J. Frey, and H.A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[10] P. F. Lazarsfeld and N. W. Henry. *Latent Structure Analysis*. Houghton Mifflin, 1968.

[11] G. McLachlan and D. Peel. *Finite Mixture Models*. Wiley, 2000.

[12] C. E. Metz. Basic principles of ROC analysis. *Seminars in Nuclear Medecine*, 4(8), 1978.

[13] T. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, MIT, 2001.

[14] David Stern, Ralf Herbrich, and Thore Graepel. Matchbox: Large scale bayesian recommendations. In *WWW'09: Proceedings of the 18th International Conference on World Wide Web*, pages 111–120, New York, NY, USA, 209. ACM Press.

APPENDIX

A. ALGEBRAIC DETAILS

The most involved messages to derive are those from factor f_{id} to the class membership variable c_i and to the leaf Bernoulli probability variables t_{ld}. Computing $m_{f_{id}\to c_i}(c_i)$ requires marginalizing over the Bernoulli probability variables, and the realization that the integral factorizes:

$$\begin{aligned}
m_{f_{id}\to c_i}(c_i) &= \int_{\{t_{jd}\}_{j=1}^{K}} f_{id}(c_i, x_{id}, \{t_{jd}\}) \prod_{j=1}^{K} m_{t_{jd}\to f_{id}}(t_{jd}) \mathrm{d}t_{jd} \\
&= \prod_{j=1}^{K} \int_{\{t_{jd}\}_{j=1}^{K}} \left[{t_{jd}}^{x_{id}} (1 - t_{jd})^{1-x_{id}} \right]^{\mathbb{I}(c_i=j)} m_{t_{jd}\to f_{id}}(t_{jd}) \mathrm{d}t_{jd} \\
&= \prod_{j=1}^{K} \left[{\mu_{jd}}^{x_{id}} (1 - \mu_{jd})^{1-x_{id}} \right]^{\mathbb{I}(c_i=j)} .
\end{aligned}$$

The computation of message $m_{f_{id}\to t_{ld}}(t_{ld})$ requires the computation of message $m_{c_i\to f_{id}}(c_i)$. This last message is easy to obtain by dividing the marginal $p(c_i|\vec{x}_i)$ by the incoming message $m_{f_{id}\to c_i}(c_i)$:

$$\begin{aligned}
m_{c_i\to f_{id}}(c_i) &= \frac{p(c_i|\vec{x}_i)}{m_{f_{id}\to c_i}(c_i)} \\
&= \prod_{j=1}^{K} \left[\frac{r_{ij}}{{\mu_{jd}}^{x_{id}} (1 - \mu_{jd})^{1-x_{id}}} \right]^{\mathbb{I}(j=c_i)} .
\end{aligned}$$

Message $m_{f_{id}\to t_{ld}}(t_{ld})$ can now be computed as:

$$\begin{aligned}
m_{f_{id}\to t_{ld}}(t_{ld}) &= \sum_{c_i=1}^{K} \int_{\{t_{jd}\}_{j\neq l}} f_{id}(c_i, x_{id}, \{t_{jd}\}) m_{c_i\to f_{id}}(c_i) \\
&\quad \cdot \prod_{j\neq l}^{K} m_{t_{jd}\to f_{id}}(t_{jd}) \mathrm{d}t_{jd} \\
&= \sum_{c_i=1}^{K} \left[r_{il} \frac{{t_{ld}}^{x_{id}} (1 - t_{ld})^{1-x_{id}}}{{\mu_{ld}}^{x_{id}} (1 - \mu_{ld})^{1-x_{id}}} \right]^{\mathbb{I}(l=c_i)} \\
&\quad \cdot \int_{\{t_{jd}\}_{j\neq l}} \prod_{j\neq l}^{K} \left[r_{ij} \frac{{t_{jd}}^{x_{id}} (1 - t_{jd})^{1-x_{id}}}{{\mu_{jd}}^{x_{id}} (1 - \mu_{jd})^{1-x_{id}}} \right]^{\mathbb{I}(j=c_i)} \\
&\quad \cdot \prod_{j\neq l}^{K} \mathrm{Beta}(t_{jd}; \alpha_{jd}, \beta_{jd}) \mathrm{d}t_{jd} \\
&= r_{il} \frac{{t_{ld}}^{x_{id}} (1 - t_{ld})^{1-x_{id}}}{{\mu_{ld}}^{x_{id}} (1 - \mu_{ld})^{1-x_{id}}} + \sum_{j\neq l}^{K} r_{ij} .
\end{aligned}$$