6

Generative Probabilistic Models

Thijs Westerveld[1], Arjen de Vries[1], and Franciska de Jong[2]

[1] Centrum voor Wiskunde en Informatica
[2] University of Twente

6.1 Introduction

Many content-based multimedia retrieval tasks can be seen as decision theory problems. Clearly, this is the case for classification tasks, like face detection, face recognition, or indoor/outdoor classification. In all these cases a system has to decide whether an image (or video) belongs to one class or another (respectively face or no face; face A, B, or C; and indoor or outdoor). Even the *ad hoc* retrieval tasks, where the goal is to find *relevant* documents given a description of an information need, can be seen as a decision theory problem: documents can be classified into relevant and non-relevant classes, or we can treat each of the documents in the collection as a separate class, and classify a query as belonging to one of these. In all these settings, a probabilistic approach seems natural: an image is assigned to the class with the highest probability.[3]

The generative probabilistic approach to image retrieval described in this chapter is one such approach. To get a feeling for the approach, the following analogy to solving jigsaws is useful. Suppose we have been solving a number of jigsaw puzzles all weekend and put all puzzles in their respective boxes again on Sunday evening. Now it is Monday morning and while cleaning the room, we find a forgotten piece of one of the jigsaws. Of course, in practice, we would keep the piece separate until we solve one of the puzzles again and discover that a piece is missing. But suppose that we have to make an immediate decision and put the piece in one of the boxes. To put it in the proper box, we have to guess to which puzzle this piece belongs. The only clues we have are the appearance of the piece at hand and our memory of the puzzles we solved. A good solution would be to put the piece in the box to which it most likely belongs given these clues. If for example, the piece at hand is mainly

[3] If some misclassifications are more severe than others, a decision theoretic approach should be taken, and images should be assigned to the class with lowest risk.

blue with a watery texture, it is most likely to come from a jigsaw with a lot of water.

In the retrieval framework presented here, instead of boxes with jigsaws we have a collection of documents, instead of a forgotten jigsaw piece, we have a query, and instead of our memories of the puzzles we have models of the documents. The goal now, is to find the document that is most likely given the query, similar to choosing the most likely box to put the jigsaw piece in. This generative approach to information retrieval – find the generating source of a piece of information – has proved successful in media specific tasks, like language modeling for text retrieval [2, 8, 16] and Gaussian mixture modeling for image retrieval [7, 14, 21, 23].

6.1.1 Relation to Other Chapters

Many chapters in this book discuss techniques for extracting features or knowledge about multimedia content or for generating metadata. This chapter introduces a method for building abstract models from such features. The models described here are independent of the type of features that are used. While the examples in this chapter use different features, the models can easily be adapted to use many of the features discussed in Chapter 5. In addition, the language modeling technique discussed in Chapter 4 is a special case of using generative probabilistic models for information retrieval. Finally, generative models play an important role in speech recognition (cf. Chapter 7).

6.1.2 Outline

As generative models can be nicely described without going into the details of parameter estimation, those two aspects are treated separately here. This chapter starts with a basic example of a generative model, followed by detailed descriptions of generative models for visual and textual information in Section 6.2. Such models are concise descriptions of the characteristics of the document, which is useful in a retrieval setting. Section 6.3 explains how the models can be used in a retrieval setting. Section 6.4 continues with a detailed description of the parameter estimation process. Here we explain how the models can be learnt from training data. Finally, Section 6.5 discusses how the two modalities can easily be combined for a truly multimodal search.

6.2 Generative Probabilistic Models

Since the goal in information retrieval is to find the best document given a query, one could decide to model the probability of a document given a query directly. In the jigsaw example, this would mean that a direct mapping from the appearance of a piece to a jigsaw box is needed, i.e., we need to calculate the likelihood of the box given the piece, i.e., P(box|piece). This way of modeling the problem is known in the classification literature as *discriminative classification*. In some cases, for example when there are many different

boxes, it is hard to learn this direct mapping. In such cases, it is useful to apply Bayesian inversion and estimate for each box the probability that this box produced the piece at hand, i.e., P(piece|box). This approach is known as *generative classification*. In this approach, each box has a model of the type of pieces it generates. The probability of generating the jigsaw piece at hand is computed for each model and that probability is used to find the most likely box.

First, the basic explanation of the generative models is continued. In information retrieval, many possible sources for a query exist; each document in a collection can be a source. Therefore, learning a discriminative classifier is hard and a generative approach is a natural way of modeling the problem. It is important to realism that in such an approach, a separate distribution is estimated for *each* of the documents in the collection. One of the nice things about generative probabilistic models is that they can easily be understood without digging into the details of estimating the models' parameters. Therefore, in the remainder of this section parameter estimation is put aside and only the basics of the models are explained. This section starts with simple examples of generative models (Section 6.2.1). Sections 6.2.2 and 6.2.3 specialize to generative image models and generative text models respectively.

6.2.1 Examples

Generative probabilistic models are random sources that can generate (infinite) sequences of samples according to some probability distribution. In the simplest case, the model generates samples independently, thus the probability of a particular sample does not depend on the samples generated previously. These simple models, often called memoryless models, will be the primary focus in this chapter. A good example of a generative source with a memoryless model is an ordinary die. The model describes the process of throwing the die and and observing the outcome. If the die is fair, throwing it generates positive integers between 1 and 6 according to a uniform distribution:[4]

$$\mathrm{P}(i) = \frac{1}{6}, \quad \text{for } i \in \{1, 2, 3, 4, 5, 6\}. \tag{6.1}$$

In a memoryless model, the observations or samples are assumed to be independent, so the probability of observing a particular sequence is calculated as the product of the probabilities of the individual observations:

$$\mathrm{P}(\{i_1, i_2, \ldots, i_n\}) = \prod_{j=1}^{n} \mathrm{P}(i_j). \tag{6.2}$$

[4] Throughout this chapter, random variables are omitted from the notation of probability functions, unless this causes confusion. Thus, $\mathrm{P}(i)$ means the probability that the random variable describing the observed outcome from throwing the die takes value i.

Section 6.3 returns to calculating the probabilities of observations. Here, the focus is on the probabilistic models themselves. A probabilistic model is an abstraction from the physical process that generates the data. Instead of specifying that the sequence of positive integers is produced by throwing an ordinary fair die, it suffices to state that there is some source that generates integers between 1 and 6 according to a uniform distribution, (6.1). The underlying physical process can remain unknown. Still, to understand the models it is often useful to think of simple processes like throwing a die, drawing colored balls from an urn, or drawing jigsaw pieces from a box.

Generative models can be more complex and have a hierarchical structure like in the following example. Suppose we have two dice (where we represent a die as a list of faces): Die A, with the usual faces 1 through 6, i.e., $A = (1, 2, 3, 4, 5, 6)$, and die B, which has ones on all faces, $B = (1, 1, 1, 1, 1, 1)$. Now we can imagine the following random process:

1. pick a die according to a uniform distribution;
2. sample a number by throwing the chosen die.

For this generative process, the probability of observing a single sample i is:

$$\mathrm{P}(i) = \mathrm{P}(A) \cdot \mathrm{P}(i|A) + \mathrm{P}(B) \cdot \mathrm{P}(i|B) = \begin{cases} \frac{1}{2} \cdot \frac{1}{6} + \frac{1}{2} \cdot 0, & \text{for } i \in \{2, 3, 4, 5, 6\} \\ \frac{1}{2} \cdot \frac{1}{6} + \frac{1}{2} \cdot 1, & \text{for } i = 1. \end{cases} \tag{6.3}$$

A generative process with a model like this is called a *mixture model*. It is a weighted sum of a number of different probability distributions. As will become clear in Section 6.2.2, mixture models are useful for describing the mixture of aspects that can be present in images.

It is often insightful to represent generative models in a graphical manner. For graphical representations, we follow the standards described in [11], where random variables are represented as nodes and dependencies between them as edges. Observed variables are represented as solid nodes and hidden, or unobserved, variables as open nodes. A box or plate around a part of the graph indicates repetition, i.e., the repeated sampling of variables. As an example, Figure 6.1 represents two variants of drawing a sequence of N numbers from the hierarchical dice. The variant on the left represents the process as described above: for each of the N numbers, we pick a new die. The variant on the right represents the case where we select a die once for the whole process and then repeatedly sample numbers by throwing that die.

The remainder of this section introduces generative models for images and text.

6.2.2 Generative Image Models

As stated in the introduction to this chapter, generative image models are like the boxes of jigsaw puzzles, from which one can randomly draw pieces. An important difference though, is the following. Jigsaw boxes contain a finite

Sampling with repeated choice of dice Sampling from a single die

Fig. 6.1. Graphical representations for dice example variants.

number (say 1000) of discrete pieces; a piece is either in there or not. By sampling from the box *with replacement*, we can draw infinitely many pieces, but each piece has to be one of the fixed set of 1000 pieces. The generative image models described below, however, are probability distributions over a (high dimensional) *continuous* feature space. The number of different samples that can be drawn is infinite. The models describe the location in the feature space where we are most likely to observe samples and what kind of variance can be expected. The nature of the feature space, i.e., the set of features used for describing a sample is not discussed in this chapter. The models are independent of the features. Here we simply assume an image is represented as a set of samples ($\mathcal{V} = \{\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_S}\}$), each described by a n-dimensional feature vector: $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$, as illustrated in Figure 6.2. The nature of the samples is independent of the models. The examples used in this chapter are based on DCT coefficients[5] and position information, but in principle other features like for example the ones introduced in Chapters 5 and 9 can be used.

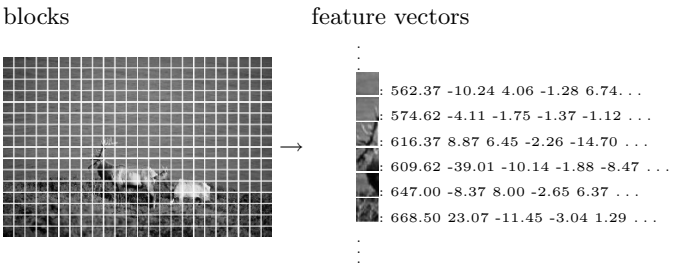

Fig. 6.2. Illustration of visual document representation.

Gaussian Mixture Models

The generative image models discussed in this chapter are based on Normal distributions, or Gaussian distributions as they are often called. These distributions are appropriate models for the situation in which an ideal point in a

[5] The Discrete Cosine Transform (DCT) captures both intensity and texture information and is also used in JPEG compression.

feature space exists and where all observations are assumed to be versions of this ideal feature vector that are randomly corrupted by many independent small influences [4]. For simple images this is the case, one can easily imagine a single ideal point in feature space describing for example the perfect water texture. All observations from the water class can be seen as versions of the ideal water texture that have been corrupted by many independent causes (lightning condition, camera angle, etc.). However, most real-life images show more than a single texture or object. Therefore, instead of using a single Gaussian distribution, it makes sense to use a mixture of Gaussian distributions for modeling images with multiple colors and textures [21].

In general, a finite mixture density is a weighted sum of a finite number (C) of density functions [20, 4]:

$$\mathrm{p}(x) = \sum_{i=1}^{C} \mathrm{P}(c_i)\mathrm{p}(x|c_i). \tag{6.4}$$

(Notation: in this chapter we consistently use capital P for a probability mass function and lowercase p for a density.)

The mixing weights $\mathrm{P}(c_i)$ are the prior probabilities of the components c_i in the mixture. The density functions $\mathrm{p}(x|c_i)$ each describe a bit of the total density. These densities are Gaussian distributions in the case of a Gaussian mixture model, but of course other densities can be used in other situations.

Titterington et al. [20] divide the usage of mixture models in two broad classes: *direct application* and *indirect application*. Direct application is used to refer to situations in which it is believed that there exists a number (C) of underlying categories or sources such that the observed samples all belong to one of these. Indirect application refers to a situation in which the link between probability distributions and categories is less clear and where a mixture model is merely used as a mathematical way of obtaining a tractable form of analyzing data. Modeling images using finite mixture models is somewhere halfway on the continuum from direct to indirect application. On the one hand, the idea is that an image can contain only a finite number of things; each sample is assumed to be generated by one of the mixture components. For example, one component might describe the grass, another the water and a third the sky in an image. This is the direct application view. On the other hand, we do not explicitly model grass, water and sky. We merely believe that to model the many different facets of an image, a mixture of distributions is needed. This mixture model describes image samples without explicitly separating the components. In that sense, mixture modeling is just a mathematical tool to describe images (indirect application view). Still, the direct application view with separate components for modeling grass, water and sky, is a useful way of thinking about finite mixture models for images, especially for understanding the parameter estimation discussed in Section 6.4.

Gaussian Mixture Models for Representing Images

A Gaussian mixture model can describe an image. The idea is that the model captures the main characteristics of the image. The samples in an image are assumed to be generated by a mixture of Gaussian sources, where the number of Gaussian components C is fixed for all images in the collection. A Gaussian mixture model is described by a set of parameters $\boldsymbol{\theta} = (\boldsymbol{\theta_1}, \ldots, \boldsymbol{\theta_C})$ each defining a single component. Each component c_i is described by its prior probability $\mathrm{P}(c_i|\boldsymbol{\theta})$, the mean $\boldsymbol{\mu_i}$ and the variance $\boldsymbol{\Sigma_i}$, thus $\boldsymbol{\theta_i} = (\mathrm{P}(c_i|\boldsymbol{\theta}), \boldsymbol{\mu_i}, \boldsymbol{\Sigma_i})$. Details about estimating these parameters are deferred to Section 6.4.1. The process of generating an image is assumed to be the following (see Figure 6.3):

1. Take the Gaussian mixture model $\boldsymbol{\theta}$ for the image.
2. For each sample $\boldsymbol{v}$ in the document:
   (a) Pick a random component c_i from Gaussian mixture model $\boldsymbol{\theta}$ according to the prior distribution over components $\mathrm{P}(c_i|\boldsymbol{\theta})$.
   (b) Draw a random sample from c_i according to the Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$.

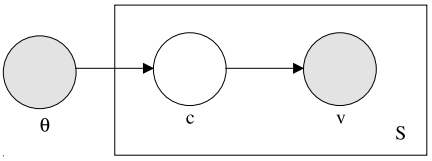

Fig. 6.3. Graphical representation of Gaussian mixture model.

Here, $\boldsymbol{\theta}$ is an observed variable, i.e., the mixture model from which the samples for a given image are drawn, is known. For a given sample however, it is unknown which component generated it, thus components are unobserved variables. The probability of drawing a single sample $\boldsymbol{v}$ from a Gaussian mixture model with parameters $\boldsymbol{\theta}$ is thus defined as the marginalization over all possible components:

$$\mathrm{p}(\boldsymbol{v}|\boldsymbol{\theta}) = \sum_{i=1}^{C} \mathrm{P}(c_i|\boldsymbol{\theta})\mathrm{p}(\boldsymbol{v}|c_i, \boldsymbol{\theta}) \tag{6.5}$$

$$= \sum_{i=1}^{C} \mathrm{P}(c_i|\boldsymbol{\theta}) \; \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma_i}|}} e^{-\frac{1}{2}(\boldsymbol{v}-\boldsymbol{\mu_i})^T \boldsymbol{\Sigma_i}^{-1}(\boldsymbol{v}-\boldsymbol{\mu_i})}. \tag{6.6}$$

A visualization of the model built from the image in Figure 6.2 is shown in Figure 6.4. For this example, a Gaussian mixture with three components is estimated from the set of feature vectors extracted from the image (cf. Figure 6.2).[6] The resulting model is described by the mean vectors and covariance

[6] The process of building a model is described in Section 6.4.1

matrices of the three components in the high-dimensional feature space and by the prior probabilities of the components. The figure shows a projection of the components onto the two-dimensional subspace defined by the position in the image plane (i.e., the space spanned by the x and y coordinates of the feature vectors). The ellipsoids in the image plane show the mean position of the three components along with their variance. The filled areas, are the areas in the image plane, where the standard deviation from the mean position for a given component is below 2. The coloring of the area is a representation of the component's other dimensions: it shows the mean color and mean texture. Variance in color and texture information are not visualized. The bars to the right of each component indicate the component's prior probability.

Fig. 6.4. Visualization of a model of the image in Figure 6.2.

In the example, three Gaussian components are used, but that is not necessarily enough to capture all information in an image. Any distribution can be approximated arbitrarily closely by a mixture of Gaussians. The higher the number of components in the mixture, the better the approximation can be. However, keeping in mind that the models will be used for retrieval, a perfect description of an image is not the ultimate goal. The goal is to find images that are similar to a query image. A perfect model would only be able to find exact matches and those are not the most interesting ones. Therefore, it is important to avoid over-fitting. Experiments have shown that eight components are typically enough to capture the most important aspects of an image [22, 24].

6.2.3 Generative Language Models

Language models are discussed in Chapter 7, where they are used for speech recognition, and Chapter 4 demonstrates their use for information retrieval. To highlight the generative nature of these models as well as the similarity to the image models discussed above, we look at them again in this chapter.

To repeat what was said before, a language model is a probability distribution over strings of text in a given language. It simply states how likely it is to observe a given string in a given language. For example, a language model for English should capture the fact that the term *the* is more likely to occur than the term *restaurant*. When context is taken into account this might change. For example, after seeing the phrase:

"*They went to an Italian*",

restaurant is a more likely completion than *the*. As discussed in Chapter 7, for speech recognition (but also for example for spelling error correction), this contextual information is important; a limited amount of context is typically taken into account and so called *n-gram models* are used [12]. In n-gram models, the probability of observing a given term only depends on the previous $n-1$ terms. If bigram models ($n = 2$) are used, the probability of the next term in the example given above would only depend on *Italian*.

For information retrieval context is of minor importance (cf. Chapter 4). Although language models are generative models, in retrieval they are not used to generate *new* pieces of text. As long as the models capture most of the topicality of a text, they are useful. Therefore, context is typically ignored in information retrieval and terms are assumed to be generated independently. The models are thus memoryless. In language modeling memoryless language models are known as unigram language models. Song and Croft experimented with higher order (n-gram) language models for information retrieval and found no significant improvement over unigram models [18].

Unigram Language Model

In the unigram language modeling approach to information retrieval, documents are assumed to be multinomial sources generating terms. This multinomial basis is not always mentioned explicitly (Chapter 4 ignored it), but, in this chapter, it is useful to take this view because it clearly shows the generative probabilistic nature and it nicely separates the model from the estimation of the model parameters, which is discussed in Section 6.4.1.

Multinomial sources are often introduced using urns with colored balls, but the boxes with jigsaw pieces we used before are equally suitable. Suppose we have a jigsaw puzzle box that contains pieces with grass, pieces with water and pieces with sky. Now, if we draw ten pieces from this box with replacement, what is the probability of observing exactly five grass pieces, two water pieces and three sky pieces? This can be modeled using a multinomial distribution. For unigram language modeling, instead of jigsaw pieces of a particular type (grass, water, sky), we have terms in a given language. A question could now be: If we draw six terms from the English language, what is the probability of observing each of the terms *an*, *Italian*, *restaurant*, *they*, *to* and *went* exactly once? In the language modeling approach to information retrieval, instead of having a single model for a whole language, each document in a collection is modeled as a separate multinomial source. Each of these models is described by a vector of term probabilities $\boldsymbol{\phi} = (\phi_1, \phi_2, \ldots, \phi_T)$, where T is the size of the vocabulary and ϕ_i is the probability of seeing term_i under model $\boldsymbol{\phi}$.

The generative process for textual documents, as visualized in Figure 6.5 is very simple:

1. Pick the language model $\boldsymbol{\phi}$ for the document.
2. For each term:

draw a random term from $\boldsymbol{\phi}$ according to the multinomial distribution mult($\boldsymbol{\phi}$).

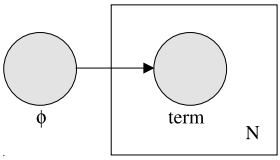

Fig. 6.5. Graphical representation of language model.

Like with the Gaussian mixture models for images, the model that generates the samples (terms) is an observed variable; each document has its own, known generative model $\boldsymbol{\phi}$. The probability of observing a particular document $\boldsymbol{t} = (t_1, t_2, \ldots, t_T)$, from this model is defined as:

$$\mathrm{P}(\boldsymbol{t}|\boldsymbol{\phi}) = \frac{\left(\sum_{i=1}^{T} t_i\right)!}{\prod_{i=1}^{T} t_i!} \prod_{i=1}^{T} \phi_i^{t_i}. \tag{6.7}$$

The second factor in this equation, $\prod_{i=1}^{T} \phi_i^{t_i}$, is the joint probability of observing the term counts for individual terms, $\mathrm{P}(\mathrm{term}_i|\boldsymbol{\phi}) = \phi_i$. The unigram assumption states that all observations are independent, thus the joint probability is simply the product of the probabilities of the individual terms. The normalization factor $\left(\sum_{i=1}^{T} t_i\right)!/(\prod_{i=1}^{T} t_i!)$ implements the *bag-of-words* model, it states that an observation (a query or a document) is a bag, the ordering of the terms is unimportant. A simple example will clarify this. Suppose we have a vocabulary with only four terms: A,B,C and D and observation $ABAC$, then $\boldsymbol{t} = (2, 1, 1, 0)$. Note that in the representation of the observation, the order of the terms is already ignored, it simply says there are two A's, one B, one C, and no D's. Thus, the probability of observing this $\boldsymbol{t}$ from a given model $\boldsymbol{\phi}$, is in fact the probability of drawing any permutation of the original string $ABAC$: $\mathrm{P}(\boldsymbol{t}) = \mathrm{P}(ABAC) + \mathrm{P}(AABC) + \mathrm{P}(ABCA) + \mathrm{P}(ACAB) + \ldots$. In total $(2+1+1+0)!\,/\,(2!{\cdot}1!{\cdot}1!{\cdot}0!) = \frac{24}{2} = 12$ different possible permutations exist. Thus $\mathrm{P}(\boldsymbol{t}) = 12\,\phi_1^2\,\phi_2^1\,\phi_3^1\,\phi_4^0$.

6.3 Retrieval Using Generative Models

By drawing enough observations from a single model (or pieces from a box, to take the jigsaw analogy), a random document or a random image can be generated. An example of a random image from the model visualized in Figure 6.4 is shown in Figure 6.6. Different models will produce different random images, just like different boxes can contain different jigsaws. This idea can be used to rank documents.

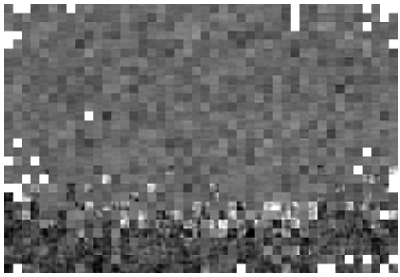

Fig. 6.6. Random sample from image model presented in Figure 6.4.

6.3.1 Sample Likelihood

The idea of ranking models based on observations is illustrated by a simple dice example. Suppose we have two dice:

$$\begin{aligned} D_1 &= (1, 2, 3, 4, 5, 6) \\ D_2 &= (1, 1, 3, 4, 5, 6). \end{aligned} \tag{6.8}$$

Say someone tells us that a sequence of five throws with one of them resulted in the observation: $O = (4, 3, 4, 3, 1)$. We can then easily calculate the likelihood of observing this sequence given each of the models:

$$\begin{aligned} \mathrm{P}(O|D_1) &= (\tfrac{1}{6})^5 \\ \mathrm{P}(O|D_2) &= (\tfrac{1}{6})^4 \cdot \tfrac{2}{6}. \end{aligned} \tag{6.9}$$

Since $\mathrm{P}(O|D_2) > \mathrm{P}(O|D_1)$, the observation is more likely under D_2. We call this probability of an observation O given a model D the *Sample Likelihood*: it is the likelihood of observing this sample.

The same principle can be used to rank documents given a query. The assumption is that the query is an observation from one of the generative document models in the collection and the goal is to find the document model under which this query is most likely. For a visual query $\mathcal{V} = \{\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_S}\}$, assuming memoryless models, we can compute the joint likelihood of observing all samples by taking the product of the likelihoods for the individual samples $\boldsymbol{v_j}$:

$$\mathrm{p}(\mathcal{V}) = \prod_{\boldsymbol{v} \in \mathcal{V}} \mathrm{p}(\boldsymbol{v}|\boldsymbol{\theta}). \tag{6.10}$$

For textual queries $\boldsymbol{q} = (q_1, q_2, \ldots, q_T)$, we can simply use (6.7).

6.4 Estimating Model Parameters

6.4.1 Maximum Likelihood Estimates

In the previous sections, the assumption has been that the model parameters ($\boldsymbol{\theta}$) are known. Given the parameters, it is straightforward to use the

models for ranking documents (as we have seen in Section 6.3). In general however, the parameters of a specific document model are unknown. Usually, the only available information is the representation of the documents, i.e., the feature vectors. A common way to use this data is to assume that they are observations from the models and use them as training samples to estimate the unknown model parameters. As a first step to estimating these parameters, we will use the *maximum likelihood estimate*. This estimate is defined as the parameter setting which maximizes the likelihood of the observed samples. Thus, for a set of training samples $\mathcal{S} = \{s_1, s_2, \ldots, s_K\}$ and model parameter $\boldsymbol{\psi}$, the maximum likelihood estimate $\boldsymbol{\psi}_{\mathrm{ML}}$ is defined as:

$$\boldsymbol{\psi}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\psi}} \prod_{s \in \mathcal{S}} \mathrm{P}(s|\boldsymbol{\psi}). \tag{6.11}$$

Below this approach is applied to Gaussian mixture models and language models. Techniques for handling unobserved data and for improving generalization capabilities are discussed in Section 6.4.2.

Estimating Gaussian Mixture Model Parameters

The maximum likelihood estimate for a Gaussian mixture model from a set of samples $\mathcal{V}$ (an image) is defined as follows:

$$\begin{aligned} \boldsymbol{\theta}_{\mathrm{ML}} &= \arg\max_{\boldsymbol{\theta}} \prod_{\boldsymbol{v} \in \mathcal{V}} \mathrm{P}(v|\boldsymbol{\theta}) \\ &= \prod_{\boldsymbol{v} \in \mathcal{V}} \sum_{i=1}^{C} \mathrm{P}(c_i|\boldsymbol{\theta}) \; \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma_i}|}} e^{-\frac{1}{2}(\boldsymbol{v}-\boldsymbol{\mu_i})^T \boldsymbol{\Sigma_i}^{-1} (\boldsymbol{v}-\boldsymbol{\mu_i})}. \end{aligned} \tag{6.12}$$

This equation is hard to solve analytically, but we can use the Expectation Maximization (EM) algorithm [3] to find parameters for the model. To understand this iterative procedure, it is useful to assume that an image shows a limited number of different things (such as grass, sky, water), each of which is modeled by a separate Gaussian distribution. Each sample in a document can then be assumed to be generated from one of these Gaussian components.

To accurately describe the different components of a Gaussian mixture model for a given document, it is necessary to decide which of the document's samples are generated by which component. The assignments of samples $\boldsymbol{v_j}$ to components C_i are unknown, but they can be viewed as hidden variables and the EM algorithm can be applied. This algorithm iterates between estimating the a posteriori class probabilities for each sample given the current model settings (the E-step) and re-estimating the components' parameters based on the sample distribution and the current sample assignments (M-step).

The EM algorithm first assigns each sample to a random component. Next, the first M-step computes the parameters ($\boldsymbol{\theta_i}$) for each component, based on the samples assigned to that component. Using maximum likelihood estimates,

this comes down to computing the mean and variance of the feature values over all samples assigned to the component.

This assignment of samples to components is a *soft* clustering, a sample does not belong entirely to one component. In fact, we compute means, covariances and priors on the weighted feature vectors, where the feature vectors are weighted by their proportion of belonging to the class under consideration. In the next E-step, the class assignments are re-estimated, i.e., the posterior probabilities, $\mathrm{P}(c_i|\boldsymbol{v_j})$ are computed. We iterate between estimating class assignments (expectation step) and estimating class parameters (maximization step) until the algorithm converges. Figure 6.7 is a visualization of training a model from the image in Figure 6.2. From top to bottom, it alternates between showing sample assignments (E-step) and visualizations of the intermediate models (M-step). After 10 iterations already, the model accurately distinguishes water, grass and elks.

More formally, to estimate a Gaussian mixture model from a document $\mathcal{V} = \{\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_S}\}$, the following steps are alternated:

E-step

Estimate the hidden assignments h_{ij} of samples to components for each sample x_j and component c_i:

$$h_{ij} = \mathrm{P}(c_i|\boldsymbol{v_j}) = \frac{\mathrm{p}(\boldsymbol{v_j}|c_i)\mathrm{P}(c_i)}{\sum_{c=1}^{C} \mathrm{p}(\boldsymbol{v_j}|c_c)\mathrm{P}(c_c)}. \tag{6.13}$$

M-step

Update the component's parameters to maximize the joint distribution of component assignments and samples: $\boldsymbol{\theta}^{\mathrm{new}} = \arg\max_{\boldsymbol{\theta}} \mathrm{p}(\mathcal{V}, \boldsymbol{H}|\boldsymbol{\theta})$, where $\boldsymbol{H}$ is the matrix with all sample assignments h_{ij}. More specifically, this means:

$$\boldsymbol{\mu_i}^{\mathrm{new}} = \frac{\sum_j h_{ij}\boldsymbol{v_j}}{\sum_j h_{ij}}, \tag{6.14}$$

$$\boldsymbol{\Sigma_i}^{\mathrm{new}} = \frac{\sum_j h_{ij}(\boldsymbol{v_j} - \boldsymbol{\mu_i}^{\mathrm{new}})(\boldsymbol{v_j} - \boldsymbol{\mu_i}^{\mathrm{new}})^T}{\sum_j h_{ij}}, \tag{6.15}$$

$$\mathrm{P}(c_i)^{\mathrm{new}} = \frac{1}{N}\sum_j h_{ij}. \tag{6.16}$$

The algorithm is guaranteed to converge [3]. The error after each iteration is the negative log likelihood of the training data:

$$E = -\log \mathrm{p}(\mathcal{V}) = -\sum_{\boldsymbol{v}\in\mathcal{V}} \log \mathrm{p}(\boldsymbol{v}|\boldsymbol{\theta}). \tag{6.17}$$

This error will decrease with each iteration of the algorithm, until a minimum is reached.[7]

[7] The found minima are local ones. The effects of this on retrieval quality need thorough investigation.

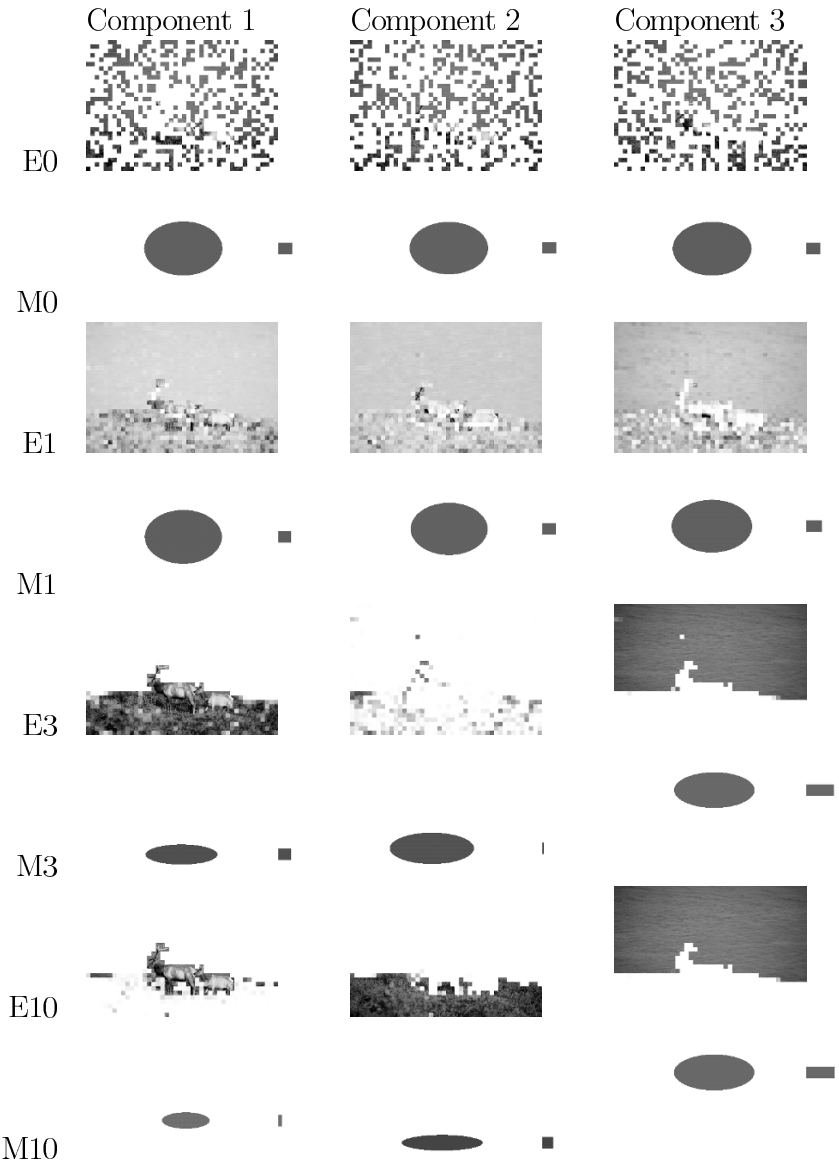

Fig. 6.7. Visualization of the estimation of parameters for a Gaussian mixture model built from the image shown in Figure 6.2. E and M steps are shown after initialization and after 1, 3 and 10 iterations. The E-steps show to what degree each sample is assigned to each component (higher transparency indicates a lower degree of assignment). The M-steps show visualizations of the models (cf. Figure 6.4).

Estimating Language Model Parameters

The maximum likelihood estimates for the parameters of the multinomial distribution for a given document are straightforward. They are simply the relative frequency of the terms in the document. If a document is represented as a vector of term counts, $\boldsymbol{t} = (t_1, t_2, \ldots, t_T)$, then ϕ_i, the probability of term i in this document, is estimated by:

$$\phi_{i_{\mathrm{ML}}} = \frac{t_i}{\sum_{j=1}^{T} t_j}. \tag{6.18}$$

6.4.2 Smoothing

If maximum likelihood estimates, (6.18), are used to find the language model parameters, we run into the so-called *zero-frequency problem*, a sparse data problem. Terms that did not occur in the training data for a document are assigned zero probability ($\phi_i = 0$ for these terms). This means that a query containing such a term will get zero probability for this document model, no matter how likely the other query terms are.

Consider for example the dice example of Section 6.3, where we introduced the following two dice:

$$\begin{aligned} D_1 &= (1, 2, 3, 4, 5, 6) \\ D_2 &= (1, 1, 3, 4, 5, 6). \end{aligned} \tag{6.19}$$

Now, if we observe the sequence $O = (1, 2, 1, 4, 3)$, we would conclude the observation comes from D_1, since $\mathrm{P}(2|D_2) = 0$ and thus $\mathrm{P}(O|D_2) = 0$. If D_2 indeed does not have a 2 on one of its faces, this is correct, but if the distribution is estimated from data (as it is in the generative document models) it may not be. Suppose we buy a die in a shop, we roll it six times and we observe the sequence $(1, 1, 3, 4, 5, 6)$, concluding that these six observations correspond to the six faces and that there is no 2 on this die does not seem wise.

Interpolation

Smoothing solves the zero-frequency problem by transferring some of the probability mass from the observed samples to the unseen samples. The specific smoothing technique used commonly in the language modeling approach to information retrieval is *interpolation*, also known as Jelinek–Mercer smoothing [10]. For multimedia material, and especially for video data, interpolation is useful, since it allows for easy extension of the language models for describing different levels of a document, like shots, scenes and videos (discussed later in this section). For other smoothing techniques, the interested reader is referred to [12] and [25].

In Jelinek–Mercer smoothing, the maximum likelihood estimates are interpolated with a more general distribution, often called *background model*, or *collection model*; the maximum likelihood estimates are often referred to

as *foreground models* or *document models*. The smoothed estimates are calculated as follows:

$$\phi_i = \lambda\phi_{i_{\mathrm{ML}}} + (1-\lambda)\phi_{i_{\mathrm{BG}}}, \tag{6.20}$$

where $\phi_{i_{\mathrm{BG}}} = \mathrm{P}(\mathrm{term}_i)$ is the background probability of observing term_i and λ is a mixing parameter indicating the relative importance of maximum likelihood estimates.[8] The background probability is usually estimated using either collection frequency, the relative frequency of the term in the collection ($\phi_{i_{\mathrm{BG}}} = \sum_d t_{d,i} / \sum_d \sum_j t_{d,j}$), or document frequency the relative fraction of documents that the term occurs in ($\phi_{i_{\mathrm{BG}}} = df(t_i) / \sum_j df(t_j)$). The mixing parameter λ can be estimated on a training set with known relevant query-document pairs.

The *idf* Role of Smoothing

Besides avoiding the zero-frequency problem, smoothing also serves another purpose, namely that of explaining common query terms and reducing their influence [25]. Because common terms have high background probability, the influence of their foreground probability on the ranking will be relatively small. This becomes apparent when we substitute the ϕs in the retrieval function, (6.7), for the smoothed estimates, (6.20), and do some formula manipulation:

$$\mathrm{P}(\boldsymbol{q}|\boldsymbol{\phi}) = \frac{\left(\sum_{j=1}^{T} q_j\right)!}{\prod_{j=1}^{T} q_j!} \prod_{i=1}^{T} [\lambda\phi_{i_{\mathrm{ML}}} + (1-\lambda)\phi_{i_{\mathrm{BG}}}]^{q_i} \tag{6.21}$$

$$= \frac{\left(\sum_{j=1}^{T} q_j\right)!}{\prod_{j=1}^{T} q_j!} \prod_{i=1}^{T} \left[\frac{\lambda\phi_{i_{\mathrm{ML}}}}{(1-\lambda)\phi_{i_{\mathrm{BG}}}} + 1\right]^{q_i} \prod_{i=1}^{T} [(1-\lambda)\phi_{i_{\mathrm{BG}}}]^{q_i}. \tag{6.22}$$

For terms that are not present in the document $\lambda\phi_{i_{\mathrm{ML}}} = 0$ and the corresponding factor reduces to 1. Thus the first product needs only to be considered for query terms that are matched in the document; The latter is document independent and can be ignored for ranking. Also the normalization factor does not affect the ranking. The reduced formula is:

$$\mathrm{P}(\boldsymbol{q}|\boldsymbol{\phi}) \propto \prod_{i \in \{1,\ldots,T\}: t_i > 0 \wedge q_i > 0} \left[\frac{\lambda\phi_{i_{\mathrm{ML}}}}{(1-\lambda)\phi_{i_{\mathrm{BG}}}} + 1\right]^{q_i}. \tag{6.23}$$

In this last equation, it is clear that the background probability ($\phi_{i_{\mathrm{BG}}}$) plays a normalization role, similar to *idf* in traditional *tf.idf* weighting [17]. Common terms, i.e., terms with high $\phi_{i_{\mathrm{BG}}}$, contribute less to the final ranking; for these terms, the influence of $\phi_{i_{\mathrm{ML}}}$ is reduced.

[8] Note that the smoothed distribution is a mixture model (cf. Section 6.2.2), with $\phi_{i_{\mathrm{ML}}}$ and $\phi_{i_{\mathrm{BG}}}$ describing the class densities and where λ and $1-\lambda$ are the class priors.

Interpolated Language Models for Video

Besides smoothing, interpolation can serve other purposes. For example when a document collection contains video material, we would like to exploit the hierarchical data model of video, in which a video is subdivided into scenes, which are subdivided into shots, which are in turn subdivided into frames. Interpolation based smoothing is particularly well-suited for modeling such representations of the data. To include the different levels of the hierarchy, we can simply extend estimation of the mixture of foreground and background model, (6.20), with models for shots and scenes:

$$\phi_i = \lambda_{\text{Shot}} \text{P}(\text{term}_i|\text{Shot}) + \lambda_{\text{Scene}} \text{P}(\text{term}_i|\text{Scene}) + \lambda_{\text{Coll}} \text{P}(\text{term}_i), \\ \text{where } \lambda_{\text{Coll}} = 1 - \lambda_{\text{Shot}} - \lambda_{\text{Scene}}. \quad (6.24)$$

The main idea behind this approach is that a good shot contains the query terms and is part of a scene having more occurrences of the query terms. Also, by including scenes in the model, misalignment between audio and video can be handled. Depending on the information need of the user, a similar strategy might be used to rank scenes or complete videos instead of shots, that is, the best scene might be a scene that contains a shot in which the query terms (co-)occur. Finally, interpolated language models are not only suitable for video retrieval, they can be used in any situation where language has a hierarchical structure. For example, it can be used for passage retrieval from (xml) documents, where a document can be a hierarchical structure of chapters, sections and paragraphs [15].

Interpolated Gaussian Mixture Models

The zero-frequency problem does not exist for images, since they are modeled using Gaussian mixture models and Gaussians have infinite support. However, the *idf* role of smoothing is also useful in image retrieval, since it distinguishes between common and typical features of a document. Suppose we have a query image depicting a clear blue sky over a snowy mountain. Now, if all images in our collection have clear blue skies, than the retrieval results should mainly depend on the snowy and mountainy bits. This means we may want to down-weight the influence of the sky bits. This can, like in the text case, be achieved by interpolating with a more general, background distribution. The new version of the likelihood for a single image sample $\boldsymbol{v}$, cf. (6.5), becomes:

$$\text{p}(\mathcal{V}) = \prod_{\boldsymbol{v} \in \mathcal{V}} \kappa \text{p}(\boldsymbol{v}|\boldsymbol{\theta}) + (1 - \kappa) \text{p}(\boldsymbol{v}), \quad (6.25)$$

where κ is used as the mixing parameter. The background density $\text{p}(\boldsymbol{v})$ can be estimated by marginalization over all document models in a reference collection $\boldsymbol{\Theta}$:

$$\text{p}(\boldsymbol{v}) = \sum_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \text{p}(\boldsymbol{v}|\boldsymbol{\theta}) \text{P}(\boldsymbol{\theta}). \quad (6.26)$$

The reference collection $\boldsymbol{\Theta}$ can be either the current collection, a representative sample, or a separate, comparable collection.

This interpolation of foreground and background probabilities has the same effect as in the text case; it will decrease the influence of samples $\boldsymbol{v}$ with a high background probability $\mathrm{p}(\boldsymbol{v})$ on the final ranking. Experiments have shown that, this interpolation with a background collection is crucial for retrieval performance [22, 24].

6.5 Combining Visual and Textual Information

Above, we have described models for visual and textural information in isolation, but it makes sense to combine the two. One could imagine textual information setting the context (this shot is about Yasser Arafat), whereas visual information could filter the shots in the video where the person (or his patterned scarf) is actually visible. Vice versa, visual information could set a context (there is an object against a clear blue sky visible), and textual information could help in deciding whether it is a helicopter, an aircraft or a balloon. If both textual and visual information are modeled in a generative framework like discussed in this chapter, combining the modalities is a viable option.

6.5.1 Joint Probability

When both visual and textual information are described using generative probabilistic models, we can simply compute the joint probability of observing textual and visual part of a multimedia query, $Q_{MM} = \{\mathcal{V}, \boldsymbol{q}\}$:

$$\mathrm{p}(Q_{MM}|D) = \mathrm{p}(\mathcal{V}|\boldsymbol{\theta_D})\mathrm{P}(\boldsymbol{q}|\boldsymbol{\phi_D}). \tag{6.27}$$

This requires two independence assumptions:

1. Textual terms and visual samples are generated independently: $\mathrm{p}(\mathcal{V}, \boldsymbol{q}|\cdot) = \mathrm{p}(\mathcal{V}|\cdot)\mathrm{P}(\boldsymbol{q}|\cdot)$.
2. The generation of documents in one modality is independent of the model in the other modality. The generation of textual terms only depends on the language model and the generation of visual terms only on the visual model.

Treating textual and visual information independently, contradicts the assumption that textual information is useful for visual multimedia retrieval. If textual information can actually help in retrieving relevant visual images or shots, then documents that have a high likelihood based on textual information should be more likely to be visually relevant than documents with a low textual score. Clearly, textual and visual information are dependent. As soon as a document is likely to be relevant based on the textual information, then the likelihood of observing something visually similar to the query examples should increase. For example, if the name Yasser Arafat is mentioned, the

likelihood of observing him increases. Theoretically, this might lead to overly high scores for documents that match on both textual and visual information. In practice this simple multimodal model gives reasonable results, although in many settings textual information is most useful for finding the relevant information [24, 22].

6.6 Summary

In this chapter the main principles of generative probabilistic models are introduced. These models provide concise descriptions of the characteristics of a document. The particular instance of generative models described in this chapter, Gaussian mixture models, is well-suited for describing documents with a variety of different characteristics, and therefore useful for modeling heterogeneous images.

In a retrieval setting, the generative properties are used to decide which documents to show to the user. Documents are ranked by decreasing probability of generating the various parts of the query. These query parts can be small descriptions of visual information in a visual setting, query terms in a textual setting, or a combination of both in a multimodal setting.

To train the generative models from data, we start from a maximum likelihood approach. The parameter setting of a model describing a document are those that maximize the likelihood of observing that document. The maximum likelihood estimate however does not distinguish between characteristics that are common for many documents in a collection and characteristics that are typical of a particular document. For retrieval, this distinction is very important. Therefore, we interpolate with a background model, a model describing the main characteristics shared by many documents in the collection. This interpolation decreases the influence of common characteristics and thus improves retrieval results.

Finally, we show how the modeling of textual information in the same generative probabilistic framework can be adapted to describe video documents. The two modalities can then be combined using a simple joint probability. Even though the independence assumption needed for this joint probability is somewhat counter intuitive, in practice this combination of modalities is useful.

6.7 Further Reading

A very thorough introduction to generative models can be found in the book by Duda et al. [4]. The book covers many aspects related to this chapter such as maximum likelihood estimation, mixture models and the expectation maximization (EM) algorithm as well as many other pattern classification techniques.

Few examples exist of the application of generative models for multimedia retrieval. The work most closely related to the models presented in this chapter

is that of Vasconcelos et al. [21] and that of Greenspan et al. [7]. Both model each of the images in the collection using a mixture of Gaussians like discussed above. Instead of then using the maximum likelihood for ranking, they also estimate a Gaussian mixture model for the query image and directly compare the models. Gaussian mixture models that model not only color and texture, but also the dynamic aspects of those are discussed in [6, 9]. Generative models are also in computer vision sometimes to classify objects [5] or medical video clips [14].

A collection of high-quality papers on the application of language modeling techniques is available in the book *Language Modeling for Information Retrieval*, edited by Croft and Lafferty [2]. A number of papers in this collection are of particular interest. First of all, the paper by Sparck-Jones and others [19] started some controversy around the idea of using language models for information retrieval, since the notion of relevance is absent from the framework and the goal is to find *the* document that generated the query terms, implying there can only be one relevant document. Lavrenko and Croft [13] solve the problem by estimating *relevance models* rather than *document models*. Finally, Lafferty and Zhai argue that the language modeling framework and the traditional probabilistic framework are probabilistically equivalent [2].

We briefly discussed the combination of visual and textual information. Of course they can be more tightly coupled than by their joined probability. Blei et al. [1] give a fine discussion of a generative approach for representing images and captions simultaneously.

Finally, the work that lead to this chapter has been published previously in many places. Elsewhere [22, 23] we give more extensive discussions on the techniques presented here.

References

1. David M. Blei and Michael I. Jordan. Modeling annotated data. In Jamie Callan, Gordon Cormack, Charles Clarke, David Hawking, and Alan F. Smeaton, editors, *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Toronto Canada, 2003. ACM Press.
2. W. Bruce Croft and John Lafferty, editors. volume 13 of *The Kluwer International Series on Information Retrieval*. Kluwer Academic Publishers, 2003.
3. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, series B*, 39(1):1–38, 1977.
4. Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley-Interscience, 2nd edition, 2000.
5. R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2003.
6. Hayit Greenspan, Jacob Goldberger, and Arnaldo Mayer. Probabilistic space-time video modeling via piecewise GMM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):384–396, 2004.

7. Hayit Greenspan, Jacob Goldberger, and Lenny Ridel. A continuous probabilistic framework for image matching. *Computer Vision and Image Understanding*, 84(3):384–406, 2001.
8. Djoerd Hiemstra. A linguistically motivated probabilistic model of information retrieval. In Christos Nicolaou and Constantine Stephanidis, editors, *Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, volume 513 of *Lecture Notes in Computer Science*, pages 569–584. Springer-Verlag, 1998.
9. Tzvetanka Ianeva, Arjen P. de Vries, and Thijs Westerveld. A dynamic probabilistic retrieval model. In *IEEE International Conference on Multimedia and Expo (ICME)*, pages 1607–1610, 2004.
10. F. Jelinek and R. L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In E. Gelsema and L. Kanal, editors, *Proceedings of the Workshop on Pattern Recognition in Practice*, pages 381–397, 1980.
11. Michael I. Jordan. Graphical models. *Statistical Science*, 19(1):140–155, 2003. Special Issue on Bayesian Statistics.
12. Daniel Jurafsky and James H. Martin. *Speech and Language Processing – An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, 2000.
13. Victor Lavrenko and W. Bruce Croft. Relevance-based language models: Estimation and analysis. In Croft and Lafferty [2].
14. Hangzai Luo, Jianping Fan, Jing Xiao, and Xingquan Zhu. Semantic principal video shot classification via mixture gaussian. In *IEEE International Conference on Multimedia and Expo (ICME)*, 2003.
15. Paul Ogilvie and Jamie Callan. Using language models for flat text queries in XML retrieval. In *Proceedings of the Initiative for the Evaluation of XML Retrieval Workshop (INEX 2003)*, 2003.
16. J. M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In W. Bruce Croft, Alistair Moffat, C. J. van Rijsbergen, Ross Wilkinson, and Justin Zobel, editors, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281. ACM Press, 1998.
17. Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.
18. Fei Song and W. Bruce Croft. A general language model for information retrieval. In *Proceedings of the Eighth International Conference on Information and Knowledge Management*, pages 316–321. ACM Press, 1999.
19. Karen Sparck Jones, Stephen Robertson, Djoerd Hiemstra, and Hugo Zaragoza. Language modeling and relevance. In Croft and Lafferty [2].
20. D. M. Titterington, A. F. M. Smith, and U. E. Makov. *Statistical Analysis of Finite Mixture Distributions*. Wiley Series in Probability and Mathematical Statistics. John Wiley and Sons, 1985.
21. Nuno Vasconcelos. *Bayesian Models for Visual Information Retrieval*. PhD thesis, Massachusetts Institut of Technology, 2000.
22. Thijs Westerveld. *Using generative probabilistic models for multimedia retrieval*. PhD thesis, University of Twente, November 2004.
23. Thijs Westerveld and Arjen P. de Vries. Generative probabilistic models for multimedia retrieval: query generation against document generation. *IEE Proceedings – Vision, Image, and Signal Processing*, 152(6):852–858, December 2005.

24. Thijs Westerveld, Arjen P. de Vries, Alex van Ballegooij, Fransiska M. G. de Jong, and Djoerd Hiemstra. A probabilistic multimedia retrieval model and its evaluation. *EURASIP Journal on Applied Signal Processing*, 2003(2):186–198, 2003. Special issue on Unstructured Information Management from Multimedia Data Sources.
25. Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 334–342. ACM Press, 2001.