CS 181 Spring 2022 Section 9
GMMs, LDA, pLSA

1 Mixture Models

1.1 Motivation

A *mixture model* is a type of probabilistic model for unsupervised learning. Suppose you have some observed data $\{\mathbf{x}_n\}_{n=1}^N$. Mixture models are used when you have reason to believe that each individual observation has a discrete *latent variable* $\mathbf{z}_n$ that determines the data generating process. A latent variable is some piece of data that is unknown, but influences the observed data.

Say there are K possible values for each $\mathbf{z}_n$, denoted $\{C_k\}_{k=1}^K$ where each C_k is a one-hot encoded vector of length K.

Consider the following data-generating process for each data point $\mathbf{x}_n$:

- Sample latent class $\mathbf{z}_n$ from $\boldsymbol{\theta}$, the categorical distribution over $\{C_k\}_{k=1}^K$ s.t. $p(\mathbf{z} = C_k; \boldsymbol{\theta}) = \theta_k$. Call this sampled latent class C_S.
- Given that $\mathbf{z}_n = C_S$, sample $\mathbf{x}_n$ from the distribution

  $$p(\mathbf{x}|\mathbf{z} = C_S; \mathbf{w})$$

  This conditional distribution is a modeling assumption (which means we will give it to you in this class), and is specified using unknown parameters $\mathbf{w}$.

  For example, we may assume that $\mathbf{x} \sim p(\mathbf{x}|\mathbf{z} = C_k) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, where $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ are the unknown mean and covariance of the k-th cluster. (See Section 2.4 for more about Gaussian mixture models!)

Example: Say you have a dataset containing weights from a random sample of animals in a pet store. Each x_n is the animal's weight. The latent variables z_n represent what kind of animal is being weighed, so the possible values $\{C_1, C_2, \ldots, C_K\}$ may represent the categories cat, dog, bird, etc. In your model, you also use the assumption that $p(x|z = C_k; \mathbf{w}) \sim N(\mu_k, \sigma_k)$.

Exercise 1. *In this example, can you give an intuitive explanation of what vector $\boldsymbol{\theta}$ represents? What does it mean that $p(x|z = C_k; \mathbf{w}) \sim N(\mu_k, \sigma_k)$?*

Solution. $\boldsymbol{\theta}$ represents the proportion of each type of animal in the pet store. For example, if $\theta_k = 0.2$ and C_k represents dogs, then 20% of all animals in the pet stores are dogs (or, if an animal is chosen at random, there is a 20% chance it will be a dog).

This class-conditional distribution means that for each type of animal, the weights of the animal are distributed normally with a fixed mean and variance for that animal. □

2 Expectation Maximization Recap

Expectation maximization is a general technique for maximum-likelihood estimation used primarily for models with latent variables. Here we will show how to use EM to train a mixture model, but EM is also used for a variety of other models!

Consider a generative mixture model consisting of a latent variable $\mathbf{z}$ from a distribution $p(\mathbf{z}; \boldsymbol{\theta})$ and an observed variable $\mathbf{x}$, such that we draw $\mathbf{x}$ from a distribution $p(\mathbf{x}|\mathbf{z}; \mathbf{w})$.

We have 2 goals:

1. To compute the MLE for $\mathbf{w}$ and $\boldsymbol{\theta}$, i.e. the values of $\mathbf{w}$, $\boldsymbol{\theta}$ that maximize $p(\mathbf{x}; \mathbf{w}, \boldsymbol{\theta})$.
2. To estimate the latent variable $\mathbf{z}_n$ corresponding to a particular $\mathbf{x}_n$, which in this case means maximize the distribution $p(\mathbf{z}_n|\mathbf{x}_n; \mathbf{w}, \boldsymbol{\theta})$.

Goal 2 is easy once we have an estimate of the MLE for $\mathbf{w}, \boldsymbol{\theta}$, because we can apply Bayes' rule:

$$p(\mathbf{z}|\mathbf{x}; \mathbf{w}, \boldsymbol{\theta}) \propto p(\mathbf{x}|\mathbf{z}; \mathbf{w}, \boldsymbol{\theta}) p(\mathbf{z}; \mathbf{w}, \boldsymbol{\theta})$$

$$p(\mathbf{z}|\mathbf{x}; \mathbf{w}, \boldsymbol{\theta}) \propto p(\mathbf{x}|\mathbf{z}; \mathbf{w}) p(\mathbf{z}; \boldsymbol{\theta}) \tag{1}$$

2.1 Why EM?

The likelihood of the data can be written as

$$p(\mathbf{x}; \mathbf{w}, \boldsymbol{\theta}) = \sum_{\mathbf{z} \in Z} p(\mathbf{x}, \mathbf{z}; \mathbf{w}, \boldsymbol{\theta})$$

Unfortunately calculating the MLE is often computationally intractable, because the log-likelihood is:

$$\log p(\mathbf{x}; \mathbf{w}, \boldsymbol{\theta}) = \log \sum_{\mathbf{z} \in Z} p(\mathbf{x}, \mathbf{z}; \mathbf{w}, \boldsymbol{\theta}) \tag{2}$$

There is no closed form for the MLE of the log-likelihood because it is the log of a sum of expressions. We know the form of the model $p(\mathbf{x}, \mathbf{z}; \mathbf{w}, \boldsymbol{\theta})$, but in general we cannot solve for the $(\mathbf{w}, \boldsymbol{\theta})$ which maximize the likelihood $p(\mathbf{x}; \mathbf{w}, \boldsymbol{\theta})$ in closed form.

2.2 The EM Algorithm

Since finding the MLE directly is difficult, we will use expectation maximization: an approximate iterative approach. The steps of the algorithm are:

1. Initialize $\mathbf{w}^{(0)}, \boldsymbol{\theta}^{(0)}$ randomly.
2. *(E-step)* Use the parameters to predict the distribution $\mathbf{q}$ for each example. The vector $\mathbf{q}_n$ represents how likely it is that the latent variable $\mathbf{z}_n$ comes from each class, given our current setting for the model parameters:

$$q_{n,k} := p(\mathbf{z}_n = C_k | \mathbf{x}_n; \mathbf{w}^{(t)}, \boldsymbol{\theta}^{(t)}) \propto p(\mathbf{x}_n | \mathbf{z}_n = C_k; \mathbf{w}^{(t)}) p(\mathbf{z}_n = C_k; \boldsymbol{\theta}^{(t)}) \tag{3}$$

3. *(M-step)* Update parameters: Choose the value of $\mathbf{w}^{(t+1)}, \boldsymbol{\theta}^{(t+1)}$ that maximizes the expected complete data log likelihood (where the expectation is over the distribution calculated above):

$$\mathbf{w}^{(t+1)}, \boldsymbol{\theta}^{(t+1)} = \arg\max_{\mathbf{w}, \boldsymbol{\theta}} \mathbb{E}_{\mathbf{Z}|\mathbf{X}} \left[\sum_{n=1}^{N} \log p(\mathbf{x}, \mathbf{z}; \mathbf{w}, \boldsymbol{\theta}) \right] \tag{4}$$

4. Go back to step 2 until the log-likelihood estimate in step 3 converges.

3 Dice Example: Mixture of Multinomials

Consider the following example scenario: we have two biased dice (with 6 faces) and one biased coin (with 2 sides). Data is generated as follows: first, the biased coin is flipped. Suppose it lands heads. Then dice 1 is rolled $c = 10$ times. This gives the first example, i.e., $\mathbf{x}_1$ would correspond to the number of times of rolling each of a $1, 2, \ldots, 6$. Then we repeat, flipping the biased coin. Suppose it lands tails. Then Dice 2 is rolled 10 times. We record the result of the dice rolls as $\mathbf{x}_2$. We keep repeating, obtaining additional examples.

For example, our observations for the first 10 rolls may look like: $1, 5, 3, 4, 2, 2, 3, 1, 6, 2$ and we'd record as our first example

$$\mathbf{x}_1 = \begin{bmatrix} 2 \\ 3 \\ 2 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

We're going to try to infer the parameters of each of the dice based on these observations. Let's consider how this scenario fits into our idea of a mixture model. First, the latent variable $\mathbf{z}_n$ has a natural interpretation as being which dice was rolled for the n^{th} observed data point

$\mathbf{x}_n$. We can represent $\mathbf{z}_n$ using a one-hot vector, so that if the n^{th} data point came from Dice 1, we'd denote that:

$$\mathbf{z}_n = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

We denote the probability vector associated with the biased coin as $\boldsymbol{\theta} \in [0,1]^2$, summing to 1, with θ_1 being the probability of the biased coin landing heads and θ_2 being the probability of the biased coin landing tails. Furthermore, we need parameters to describe the behavior of the biased dice. We use $\boldsymbol{\pi_1}, \boldsymbol{\pi}_2 \in [0,1]^6$, summing to 1, where each 6-dimensional vector describes the probability that the respective dice lands on each face.

For a given dice, this defines a multinomial distribution. For c trials, and counts $x_1, \ldots, x_6$ for each of 6 faces on a 6-sided dice, and probabilities $\boldsymbol{\pi}$, this is

$$p(\mathbf{x}; \boldsymbol{\pi}) = \frac{c!}{x_1! \cdot \ldots \cdot x_6!} \pi_1^{x_1} \cdot \ldots \cdot \pi_6^{x_6} \tag{5}$$

For our purposes, let $p(\mathbf{x}_n | \mathbf{z}_n = C_k; \boldsymbol{\pi}_1, \boldsymbol{\pi}_2)$ denote the multinomial distribution on observation $\mathbf{x}_{nj}$ when latent vector $\mathbf{z}_n = C_k$.

The model parameters are $\mathbf{w} = \{\boldsymbol{\theta}, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2\}$. We can optimize the model parameters using EM. We start by initializing the parameters $\boldsymbol{\theta}^{(0)}, \boldsymbol{\pi}^{(0)}$.

In the E-step, we compute the soft assignment values, $\mathbf{q}_n$. For dice k, this given by

$$\begin{aligned} q_{nk} =& p(\mathbf{z}_n = C_k | \mathbf{x}_n; \boldsymbol{\theta}^{(i)}, \boldsymbol{\pi}^{(i)}) && (6) \\ =& \frac{p(\mathbf{x}_n | \mathbf{z}_n = C_k; \boldsymbol{\pi}^{(i)}) p(\mathbf{z}_n = C_k; \boldsymbol{\theta}^{(i)})}{\sum_{k=1}^K p(\mathbf{x}_n | \mathbf{z}_n = C_k; \boldsymbol{\pi}^{(i)}) p(\mathbf{z}_n = C_k; \boldsymbol{\theta}^{(i)})} && (7) \\ =& \frac{p(\mathbf{x}_n | \mathbf{z}_n = C_k; \boldsymbol{\pi}^{(i)}) \theta_k^{(i)}}{\sum_{k=1}^K p(\mathbf{x}_n | \mathbf{z}_n = C_k; \boldsymbol{\pi}^{(i)}) \theta_k^{(i)}} && (8) \end{aligned}$$

We could also use the "product trick" to write a single expression

$$\begin{aligned} p(\mathbf{z}_n | \mathbf{x}_n; \boldsymbol{\theta}^{(i)}, \boldsymbol{\pi}^{(i)}) &= \frac{\prod_{k=1}^K \left(p(\mathbf{x}_n | \mathbf{z}_n = C_k; \boldsymbol{\pi}^{(i)}) \theta_k^{(i)} \right)^{z_{nk}}}{\sum_{k=1}^K p(\mathbf{x}_n | \mathbf{z}_n = C_k; \boldsymbol{\pi}^{(i)}) \theta_k^{(i)}} \\ &= \frac{\prod_{k=1}^K p(\mathbf{x}_n | \mathbf{z}_n = C_k; \boldsymbol{\pi}^{(i)})^{z_{nk}} \prod_{k=1}^K (\theta_k^{(i)})^{z_{nk}}}{\sum_{k=1}^K p(\mathbf{x}_n | \mathbf{z}_n = C_k; \boldsymbol{\pi}^{(i)}) \theta_k^{(i)}} \end{aligned}$$

The vector $\mathbf{q}_n$ is defined as:

$$\mathbf{q}_n = \begin{bmatrix} p(\mathbf{z}_n = C_1 | \mathbf{x}_n; \boldsymbol{\theta}^{(i)}, \boldsymbol{\pi}^{(i)}) \\ p(\mathbf{z}_n = C_2 | \mathbf{x}_n; \boldsymbol{\theta}^{(i)}, \boldsymbol{\pi}^{(i)}) \end{bmatrix} \tag{9}$$

After computing the values of $\mathbf{q}_n$, we are ready to perform the M-step. Recall that we are maximizing the expected complete-data log likelihood, which takes the form:

$$\mathbb{E}_{\mathbf{Z}|\mathbf{X}}[\log p(\mathbf{X}, \mathbf{Z})] = \mathbb{E}_{\mathbf{q}_n}\left[\sum_{n=1}^{N} \log p(\mathbf{z}_n; \boldsymbol{\theta}^{(i+1)}, \boldsymbol{\pi}^{(i+1)}) + \log p(\mathbf{x}_n|\mathbf{z}_n; \boldsymbol{\theta}^{(i+1)}, \boldsymbol{\pi}^{(i+1)})\right] \tag{10}$$

$$= \sum_{n=1}^{N} \mathbb{E}_{\mathbf{z}_n|\mathbf{x}_n}\left[\log p(\mathbf{z}_n; \boldsymbol{\theta}^{(i+1)}, \boldsymbol{\pi}^{(i+1)}) + \log p(\mathbf{x}_n|\mathbf{z}_n; \boldsymbol{\theta}^{(i+1)}, \boldsymbol{\pi}^{(i+1)})\right] \tag{11}$$

We can then substitute in for the multinomial expression and simplify, and dropping constants we have that we're looking for parameters that solve

$$\arg \max_{\boldsymbol{\theta}^{(i+1)}, \boldsymbol{\pi}^{(i+1)}} \left\{ \sum_{n=1}^{N} \sum_{k=1}^{2} q_{n,k} \log \theta_k^{(i+1)} + \sum_{n=1}^{N} \sum_{k=1}^{2} q_{n,k} \log \left(\pi_{k,1}^{x_n,1} \cdot \ldots \cdot \pi_{k,6}^{x_n,6} \right) \right\}$$

$$= \arg \max_{\boldsymbol{\theta}^{(i+1)}, \boldsymbol{\pi}^{(i+1)}} \left\{ \sum_{n=1}^{N} \sum_{k=1}^{2} q_{n,k} \log \theta_k^{(i+1)} + \sum_{n=1}^{N} \sum_{k=1}^{2} \sum_{j=1}^{6} q_{n,k} x_{n,j} \log(\pi_{k,j}) \right\} \tag{12}$$

To maximize the expected complete-data log likelihood, it's necessary to introduce Lagrange multipliers to enforce the constraints $\sum_k \theta_k^{(i+1)} = 1$ and $\sum_j \pi_{k,j}^{(i+1)} = 1$, for each k. After doing this, and solving, we recover the following update equations for the model parameters:

$$\theta_k^{(i+1)} \leftarrow \frac{\sum_{n=1}^{N} q_{n,k}}{N}$$

$$\boldsymbol{\pi}_k^{(i+1)} \leftarrow \frac{\sum_{n=1}^{N} q_{n,k} \mathbf{x}_n}{c \sum_{n=1}^{N} q_{n,k}},$$

where $c = 10$ in out example.

We now have everything we need to perform EM for this setup. After initializing our parameters $\mathbf{w}^{(0)}$, we perform the E-step by evaluating 9. After calculating our values of $\mathbf{q}_n$ in the E-step, we update our parameters $\mathbf{w} = \{\boldsymbol{\theta}, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2\}$ in the M-step by maximizing 12 with respect to $\boldsymbol{\theta}, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2$. We perform these two steps iteratively, until convergence of our parameters.

4 Gaussian Mixture Modeling

We now examine a commonly used mixture model called a Gaussian Mixture Model (GMM). As you might expect, a GMM consists of a combination of multiple Gaussian distributions. Among other things, it is useful for modeling scenarios where the observed data is continuous.

Let's go over a more rigorous formulation of the GMM setup. First, we have observed continous data $\mathbf{x}_n \in \mathbb{R}^m$ and latent variables $\mathbf{z}_n$ which indicate which Gaussian 'cluster' our observed data point was drawn from. In other words:

$$p(\mathbf{x}_n|\mathbf{z}_n = C_k) = \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

where $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ are the mean and covariance parameters respectively for the k^{th} cluster center.

The data generation process works as follows: we first sample a cluster center from a Categorical distribution parameterized by $\boldsymbol{\theta} \in \mathrm{R}^K$. Then, based on the sampled cluster center, we sample a data point $\mathbf{x}_n \in \mathrm{R}^m$, which is the only piece of data that we actually observe. As usual for a mixture model, it is our goal to use the observed data to determine the cluster means and covariances, as well as the parameters of the Categorical distribution that selects the cluster centers.

Fortunately, this problem setup is perfectly suited for EM. We can apply the same machinery we've discussed throughout the chapter and used in the previous example.

1. First, we randomly initialize our parameters $\boldsymbol{\theta}, \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$.

2. [E-Step] Calculate the posterior distribution over $\mathbf{z}_n$ given by $\mathbf{q}_n$:

$$\begin{aligned}\mathbf{q}_n = \mathrm{E}[\mathbf{z}_n|\mathbf{x}_n] &= \begin{bmatrix} p(\mathbf{z}_n = C_1|\mathbf{x}_n; \theta_1, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \\ \vdots \\ p(\mathbf{z}_n = C_K|\mathbf{x}_n; \theta_K, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_K) \end{bmatrix} \\ &\propto \begin{bmatrix} \theta_1 \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \\ \vdots \\ \theta_K \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_K) \end{bmatrix}\end{aligned}$$

   This is the current expectation for our latent variables $\mathbf{z}_n$ given our data $\mathbf{x}_n$ and the current setting of our model parameters $\boldsymbol{\theta}, \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$.

3. [M-Step] Using our values of $\mathbf{q}_n$, calculate the expected complete-data log likelihood, and then use that term to optimize our model parameters:

$$\begin{aligned}\mathbb{E}_{\mathbf{q}_n}[\log p(\mathbf{X}, \mathbf{Z})] &= \mathbb{E}_{\mathbf{q}_n}\left[\sum_{n=1}^N \ln(p(\mathbf{x}_n, \mathbf{z}_n; \boldsymbol{\theta}, \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K))\right] \\ &= \sum_{n=1}^N \sum_{k=1}^K q_{n,k} \ln \theta_k + q_{n,k} \ln \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\end{aligned}$$

   We can then use this expected complete-data log likelihood to optimize our model parameters $\boldsymbol{\theta}, \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ by computing the MLE as usual. Using a Lagrange multiplier

to enforce $\sum_{k=1}^{K} \theta_k = 1$, we recover the update equations:

$$\theta_k^{(i+1)} \leftarrow \frac{\sum_{n=1}^{N} q_{n,k}}{N}$$
$$\boldsymbol{\mu}_k^{(i+1)} \leftarrow \frac{\sum_{n=1}^{N} q_{n,k}\mathbf{x}_n}{\sum_{n=1}^{N} q_{n,k}}$$
$$\boldsymbol{\Sigma}_k^{(i+1)} \leftarrow \frac{\sum_{n=1}^{N} q_{n,k}(\mathbf{x}_n - \boldsymbol{\mu}_k^{(i+1)})(\mathbf{x}_n - \boldsymbol{\mu}_k^{(i+1)})^T}{\sum_{n=1}^{N} q_{n,k}}$$

4. Return to step 2. Repeat until convergence.

5 Topic Models

Topic modeling is used for discovering latent topics or themes in large collections of documents. The goal is the underlying structure in a corpus of text documents and categorizing them into different groups based on their content. Topic modeling is a type of unsupervised learning, where the algorithm tries to discover patterns and structures in the data without prior knowledge of the labels or categories. Some of the popular applications of topic modeling include document clustering, text classification, and information retrieval.

Topic modeling is similar to other latent variable models, such as Gaussian mixture models (GMMs). Like the mixture models that we study in this course, we will describe topic models generatively: one in which we assume our corpus is generated by some process. Then, we will seek to use an optimization method, such as EM, to train the parameters of our model.

5.1 Probabilistic Latent Semantic Analysis (pLSA)

Consider a collection of documents, where each document is a mixture of various topics. pLSA is a generative model that assumes each word in a document is generated by sampling from a topic, and the topic is sampled from a per-document distribution. As a generative model, we want to model the joint probability of words and documents, $p(w, d)$. The generative process for a document in pLSA is as follows:

1. Initialize conditional probabilities $p(z \mid d)$ and $p(w \mid z)$ that represent the per-document distribution for topics and per-topic distribution for words.

2. For each document d, choose a topic z with probability $p(z \mid d)$.

3. For the chosen topic z, pick a word w with probability $p(w \mid z)$.

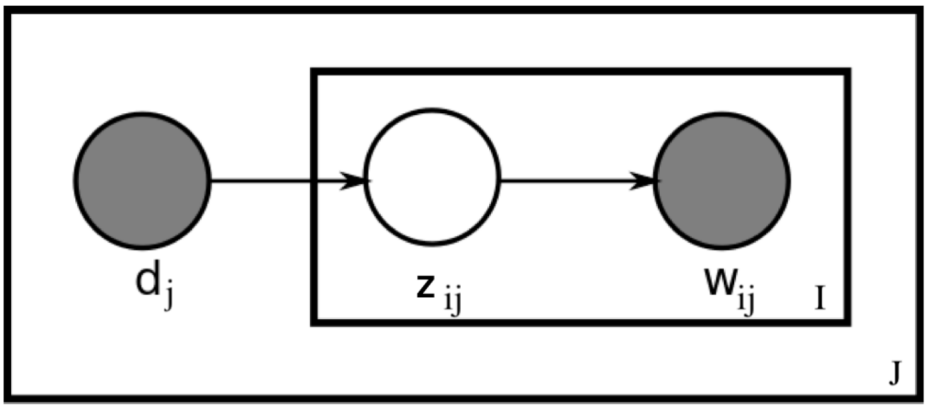
The goal of pLSA is to learn the conditional probabilities $p(w \mid z)$ and $p(z \mid d)$. In pLSA, we do not impose any priors on these probabilities beforehand. Given the observed data $p(w \mid d)$, we can train a latent model to estimate the conditional probabilities $p(w \mid z)$ and $p(z \mid d)$ based on the training data. To do this, we use the Expectation-Maximization algorithm again to maximize the likelihood of the observed data with respect to the latent variables, or the topics. The EM algorithm consists of two steps:

- Expectation (E) step: Compute the posterior probabilities of the topic assignments $p(z \mid w, d)$ using the current estimates of $p(w \mid z)$ and $p(z \mid d)$. To do so, we can use the observed data likelihood function:

$$p(w, d) = \sum_z p(w \mid z)p(z \mid d)p(d)$$

- Maximization (M) step: Update the estimates of $p(w \mid z)$ and $p(z \mid d)$ based on the posterior probabilities computed in the E step.

Note that we only know d_j and w_{ij} but define z_{ij} in order to train the conditional probabilities in such a way that makes intuitive sense in our generative model. The plate diagram for pLSA is below:



In this relatively limited model, the most notable weakness is overfitting. As the number of parameters in the model grows linearly with the number of documents, pLSA is prone to overfitting. The model does not have a prior imposed on the per-document or per-topic distributions, so all of the parameter learning is derived from the training data. In document text, this is particularly bad because the true complexity of possible document features is far more complex than just the documents in the sample corpus.

5.2 Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) is an extension of pLSA that addresses some of its limitations, mainly overfitting and lack of a generative model for new documents. Like pLSA,

LDA is a generative probabilistic model for topic modeling that assumes each document is a mixture of topics, and each topic is a distribution over words. However, LDA introduces a Dirichlet prior on the per-document topic distributions and per-topic word distributions, leading to better generalization and the ability to infer topic distributions for new documents. Specifically, we use fixed parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ as an extra "layer" of sampling.

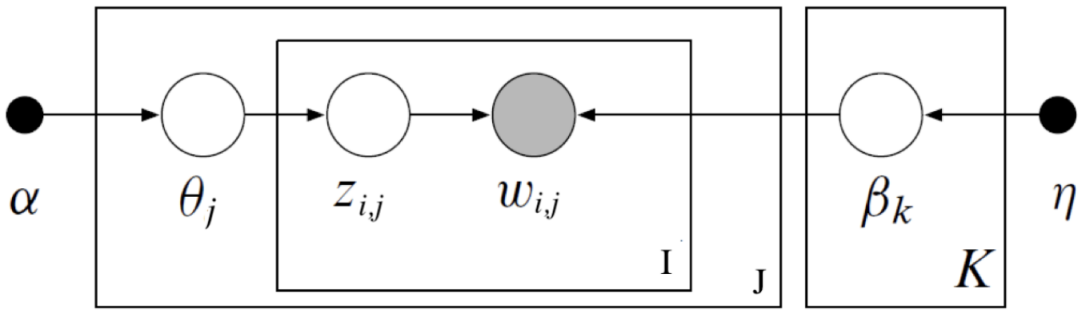
We begin by describing the generative process by which a document i is generated. For topic modeling, similar to K-means, we have to begin by picking the number of topics, K, to look for. We define a topic $\boldsymbol{\phi}_k$ to be a distribution over the words, so $\boldsymbol{\phi}_k \in [0,1]^{|\mathcal{W}|}$, where $\mathcal{W}$ is the set of words. For each document, we have a document-topic distribution $\boldsymbol{\theta}_m \in [0,1]^K$. These are the parameters to estimate in LDA.

We now describe the data generation process:

1. Let $\boldsymbol{\alpha} \in \mathbb{R}_+^K$ and $\boldsymbol{\beta} \in \mathbb{R}_+^{|\mathcal{W}|}$.
2. For each document $m = 1, \ldots, M$, sample a mixture over topics: $\boldsymbol{\theta}_n \sim Dir(\boldsymbol{\alpha})$.
3. For each topic $k = 1, \ldots, K$, sample a mixture over words in that topic: $\boldsymbol{\phi}_k \sim Dir(\boldsymbol{\beta})$.
4. For each word $w_{m,n}$ (for $m = 1, \ldots, M$ and $n = 1, \ldots, N$, the length of the document), first sample the topic $z_{m,n} \sim Cat(\boldsymbol{\theta}_m)$, then sample the word $w_{m,n} \sim Cat(\boldsymbol{\phi}_{z_{m,n}})$.

For some intuition, the Dirichlet distribution takes in a k-sized vector of values and outputs a probability distribution across k categories. Roughly speaking, the Dirichlet parameter is a vector of positive real numbers; the larger the value, the more likely that corresponding component of the sampled vector will have a higher value. At a high level, then, a topic model is a mixture over mixtures: within a single document, $\boldsymbol{\theta}_m$ specifies a distribution over topics in that document, and for each topic, k, in that document, $\boldsymbol{\phi}_k$ specifies a distribution over words.

This process is again summarized in the following plate diagram (where η in the diagram represents the Dirichlet parameter for words per topic):



Comparing this plate diagram to that of pLSA, we can observe that pLSA is just if we consider the plates across I and J, without the fixed inputs of $\boldsymbol{\alpha}$ and η. Like pLSA, we can use a version of EM to optimize the model parameters, but requires an approximation of the posterior distributions (variational inference).

The main takeaway from LDA is that the introduction of Dirichlet priors for the per-document topic distributions $\boldsymbol{\theta}$ and per-topic word distributions $\boldsymbol{\phi}$ acts as a form of regularization. By imposing these priors, the model incorporates some prior knowledge or assumptions about the distributions, which helps guide the learning process. This regularization effect prevents the model from relying too heavily on the training data, leading to better generalization and robustness against overfitting. As a result, LDA can more effectively estimate the underlying topic structure in the data and produce topic distributions for new, unseen documents. This makes LDA a more robust and widely applicable topic modeling method compared to pLSA, which lacks the regularization provided by the Dirichlet priors.

6 Naive Bayes

Naive Bayes is a type of generative model for classification tasks. It imposes the simplifying rule that for a given class C_k, we assume that each feature of the data points $\mathbf{x}$ generated within that class are independent (hence the descriptor 'naive'). This means that the conditional distribution $p(\mathbf{x}|y = C_k)$ can be written as:

$$p(\mathbf{x}|y = C_k) = \prod_{i=1}^{D} p(x_i|y = C_k)$$

where D is the number of features in our data point $\mathbf{x}$ and C_k is the class. Note that Naive Bayes does not specify the form of the model $p(x_i|y = C_k)$, this decision is left up to us.

This is obviously not a realistic simplification for all scenarios, but it can make our calculations easier and may actually hold true in certain cases. We can build more intuition for how Naive Bayes works through an example.

6.1 Naive Bayes Example

We take the same dice setup we had in section 3 but this time we know the bias of the count. The coin has probabilities as follows:

Heads : 30%
Tails : 70%

We also know the probabilities of each of the two dices. Die 1 has probabilities as follows:

1 : 40%
2 : 20%
3 : 10%
4 : 10%
5 : 10%
6 : 10%

Die 2 has probabilities as follows:

$$\begin{aligned}
\mathbf{1} &: 20\% \\
\mathbf{2} &: 20\% \\
\mathbf{3} &: 10\% \\
\mathbf{4} &: 30\% \\
\mathbf{5} &: 10\% \\
\mathbf{6} &: 10\%
\end{aligned}$$

Your friend is tasked with doing the following. First, they flip the coin. If it lands Heads, they select Die 1, otherwise they select Die 2. Then, they roll that die 10 times in a row, recording the results of the die rolls. After they have completed this, you get to observe the aggregated results from the die rolls. Using this information (and assuming you know the biases associated with the coin and dice), you must then classify which die the rolls came from. Assume your friend went through this procedure and produced the following counts:

$$\begin{aligned}
\mathbf{1} &: 3 \\
\mathbf{2} &: 1 \\
\mathbf{3} &: 2 \\
\mathbf{4} &: 2 \\
\mathbf{5} &: 1 \\
\mathbf{6} &: 1
\end{aligned}$$

Determine which die this roll count most likely came from.

We are now faced with a classification problem for a datapoint x which is the 6 roll results shown above. We can view each feature of x as 1 of the 6 rolls. This means the features of x are completely independent. This perfectly situates the classification problem in the Naive Bayes framework. Making a classification in this situation is as simple as computing the probability that the selected die produced the given roll counts. Let's start by computing the probability for Die 1:

$$\begin{aligned}
p(\text{Die 1}) &= p(\text{Coin Flip} = \text{Heads}) * p(\text{Roll Count} = [3,1,2,2,1,1]) \\
&\propto 0.3 * (0.4)^3 * (0.2)^1 * (0.1)^2 * (0.1)^2 * (0.1)^1 * (0.1)^1 \\
&\propto 3.84 * 10^{-9}
\end{aligned}$$

Notice that we don't concern ourselves with the normalization constant for the probability of the roll count - this will not differ between the choice of dice and we can thus ignore it for simplicity. Now the probability for Die 2:

$$\begin{aligned}
p(\text{Die 2}) &= p(\text{Coin Flip} = \text{Tails}) * p(\text{Roll Count} = [3,1,2,2,1,1]) \\
&\propto 0.7 * (0.2)^3 * (0.2)^1 * (0.1)^2 * (0.3)^2 * (0.1)^1 * (0.1)^1 \\
&\propto 1.008 * 10^{-8}
\end{aligned}$$

Therefore, we would classify this roll count as having come from Die 2.

Note that this problem asked us only to make a classification prediction after we already knew the parameters governing the coin flip and dice rolls. However, given a data set, we could have also used a maximum likelihood procedure under the Naive Bayes assumption to estimate the values of the parameters governing the probability of the coin flip and die rolls.