Chapter 6

Hidden Variables

We have thus far learned several probabilistic models including Naïve Bayes models and log-linear models. When these models are trained, both the inputs and the outputs are **observed variables** since they are labeled in a corpus. MLE can be used to estimate model parameters by counting relative frequencies. In practice, there are also situations where some variables in a model are not observable in training data. For example, manually labeled corpora can be scarce for low-resource languages and domains. As a second example, for machine translation modeling, it is relatively easy to find bilingual sentence pairs that are translations to each other, but labeling the alignment between word pairs in a sentence pair can be extremely costly. In such cases, those unlabeled variables are **hidden variables**, which makes it infeasible to count the relative frequencies of hidden variables directly from data.

Expectation maximisation (EM) is an iterative training algorithm that can deal with hidden variables. Given a randomly initialized model, EM contains two iterative steps, namely the expectation step and the maximisation step. The expectation step uses the current model parameters to derive the probability distribution of the counts of hidden variable over a training set. The maximisation step uses the resulting count distributions as a basis for updating the model parameters. This chapter introduces the EM algorithm, and three applications of EM in NLP, including the unsupervised Naïve Bayes model, IBM model 1 for machine translation and the probabilistic latent semantic analysis (PLSA) model. Towards the end of the chapter, theoretical justifications of EM are given.

6.1 Expectation Maximisation

EM is a general algorithm to train models with hidden variables. Formally, suppose that we have a set of observed variables O, a set of hidden variables H, and a set of model parameters Θ. The model calculates $P(O, H|\Theta)$,

namely the joint probability of observed and hidden variables. Given a specific problem, the techniques that we discussed in Chapter 2, such as the probability chain rule and independence assumptions can be used to parameterise the joint probability distribution.

Due to the existence of H, our notations are different from those of the previous chapters, where the data consists of inputs X and outputs Y, both being observed. There is no fixed mapping between (O, H) and (X, Y). For example, we will see cases where O consists of the inputs and H consists of the output, and also cases where O consists of both the inputs and the outputs, and H consists of only intermediate variables that help to better model O. We will discuss EM using the (O, H) setting for notational convenience.

It is worth noting that this notation reflects the *training setting* with latent variables, but not the *model structure* or parameterisation. In fact, the same model can be trained using both supervised MLE and EM, depending on whether there are unobserved variables in the training data. Such examples include the Naïve Bayes text classifier in this Chapter and the hidden Markov model in the next chapter. In addition to the (O, H) notation, we will include Θ in conditional probabilities in order to differentiate their values when different versions of the same parameters exist in an equation.

Hidden variables and relative frequency counting. The existence of hidden variables H makes counting relative frequencies infeasible. In particular, given a dataset $D = \{o_i\}|_{i=1}^N$ the joint likelihood is

$$L(\Theta) = \sum_{i=1}^{N} \log P(o_i, h_i|\Theta) = \log P(O, H|\Theta), \tag{6.1}$$

which is unspecified and thus cannot be maximised directly, because the counts of H is unobserved in D. Let us take naïve Bayes text classification as an example. Given a document $d = W_1^n = w_1 w_2 ... w_n$ with a class label c, the model calculates $P(d, c) = P(c) \prod_{j=1}^{n} P(w_j|c)$. Given a dataset $D = \{(d_i, c_i)\}|_{i=1}^N$, the data likelihood can be specified as

$$\begin{aligned} P(D) = \prod_{i=1}^{N} P(d_i, c_i) &= \prod_{i=1}^{N} \Big(P(c_i) \prod_{j=1}^{|d_i|} P(w_{i,j}|c_i) \Big) \\ &= \Big(\prod_{c \in C} P(c)^{N_c} \Big) \cdot \Big(\prod_{w \in V} \prod_{c \in C} P(w|c)^{N_{w,c}} \Big), \end{aligned}$$

where $w_{i,j}$ denotes the jth words in d_i, C denotes the set of class labels, V denotes the vocabulary, N_c denotes the number of documents under the class c in D and $N_{w,c}$ denotes the number of occurrences of word w in documents with class c. The parameter set is $\Theta = \{P(c), P(w|c)\}$ for $w \in V$ and $c \in C$.

In the above likelihood, the terms $\prod_{c \in C} P(c)^{N_c}$ and $\prod_{w \in V} \prod_{c \in C} P(w|c)^{N_{w,c}}$ can be seen as two independent multinomial distributions, each representing

the probability of a set of *iid* samples according to a categorial distribution (i.e., $P(c)$ and $P(w|c)$, respectively). As discussed in Chapter 2, by maximising $P(D)$ we can derive the values of $P(c)$ ($c \in C$) and $P(w|c)$ ($w \in V, c \in C$) by counting relative frequencies (i.e. N_c and $N_{w,c}$). Now given a set of unlabeled documents $D = \{d_i\}|_{i=1}^{N}$, we cannot obtain the counts N_c or $N_{w,c}$. As a result, the data likelihood is not specified, and direct MLE is infeasible.

One way to solve the problem is to use an iterative approach. In particular, suppose that we have model, we can calculate *expected counts* of the hidden variables. For example, in the Naïve Bayes case, this can be achieved using at least three methods, including (1) by calculating the most likely values of the hidden variables given the observed variables, (2) by simply calculating a distribution $P(c|d)$, where the probabilities serve as pseudo counts, and (3) by sampling a set of (d, c) pairs according to the generative story of Naïve Bayes. In case (1) above, each hidden variable instance receives a single value. For example, we determine a fixed class label for each document in D. In contrast, in case (2) and (3) above, each hidden variable can have different values. For example, for method (2), each document distributes its class labels count to each possible class c according to $P(c|d)$, and for method (3) we are likely to sample different class labels for each document at different sampling rounds. We will discuss method (1) and (2) in this chapter, and method (3) in Chapter 12 for Bayesian learning.

With the expected counts of hidden variables, we can then re-estimate the model using MLE. Then with this re-estimated model, we can have better estimations of hidden variable counts again. Thus the iterative approach can alternate the estimation of model parameters and hidden variable counts. According to this observation, EM works by randomly initialising Θ, and then iteratively executing expectation steps and optimisation steps. The former finds the expected counts of H according to the current model Θ using method (2), while the latter updates Θ by maximising the joint likelihood of O and H with the expected counts of H.

EM and k-means clustering. We have seen one iterative algorithm in this book, namely k-means clustering. The algorithm groups a given set of feature vectors into k clusters in the vectors space according to their relative distances, by randomly initialising a set of cluster centroids, and then iteratively performing cluster assignment and centroid calculation. It turns out that this algorithm is connected to EM. In particular, if the cluster centroids are taken as model parameters and the cluster assignment taken as hidden variables, k-means can be regarded as belonging to a simplified EM version by replacing method (2) above with method (1), which is conceptually simpler. We therefore introduce EM by first revisiting k-means, discussing the simplified EM version before introducing the full EM algorithm.

6.1.1 K-Means Revisited

Let us describe k-means in light of the aforementioned EM framework by first writing it down as a model with hidden variables, and then specifying its parameterisation in terms of $P(O, H|\Theta)$. Formally, given a set of observed variables (i.e., input vectors) $O = \{\vec{v}_i\}|_{i=1}^{N}$, the learning objective of k-means can be defined as minimising

$$L(\Theta) = \sum_{i=1}^{N}\sum_{k=1}^{K} h_{ik}||\vec{v}_i - \vec{c}_k||^2, \tag{6.2}$$

where $\vec{c}_k$ is the centroid of the cluster k, and h_{ik} is an indicator variable:

$$h_{ik} = \begin{cases} 1 \text{ if } \vec{v}_i \in \text{ cluster } k \\ 0 \text{ otherwise} \end{cases}$$

In this model, $H = \{h_{ik}\}|_{i=1,k=1}^{N,K}$ are hidden variables and $\Theta = \{\vec{c}_k\}|_{k=1}^{K}$ are model parameters. h_{ik} represents the assignment of input vectors to output clusters. For every $\vec{v}_i$, only one h_{ik} can be 1 among all $k \in [1, ..., K]$.

K-means uses an iterative approach to minimise Eq 6.2. It randomly initialises each $\vec{c}_k$ to $\vec{c}_k^0$, and determines the values of hidden variables H^t at the tth iteration by fixing the model parameters $\vec{c}_k = \vec{c}_k^t$ and allocating every input $\vec{v}_i$ to the closest cluster centroid:

$$H^t \leftarrow \arg\min_{H} \sum_{i=1}^{N}\sum_{k=1}^{K} h_{ik}||\vec{v}_i - \vec{c}_k^t||^2,$$

In this way, we obtain

$$h_{ik}^t = \begin{cases} 1 \text{ if } k = \arg\min_{k'} ||\vec{v}_i - \vec{c}_{k'}^t||^2 \\ 0 \text{ otherwise} \end{cases} \tag{6.3}$$

The above step serves as one expectation step, based on which k-means takes a maximisation step to reestimate the model parameters $\vec{c}_k$ ($k \in [1, ..., K]$):

$$\Theta^{t+1} \leftarrow \arg\min_{\vec{c}_1,\vec{c}_2,...,\vec{c}_K} \sum_{i=1}^{N}\sum_{k=1}^{K} h_{ik}^t||\vec{v}_i - \vec{c}_k||^2$$

Taking the derivatives of $\sum_{i=1}^{N}\sum_{k=1}^{K} h_{ik}^t||\vec{v}_i - \vec{c}_k||^2$ with respect to every $\vec{c}_k$, and setting the derivative values to zeros, we obtain the optimal values of the model parameters for the next iteration:

$$\vec{c}_k^{t+1} = \frac{\sum_{i=1}^{N} h_{ik}^t \vec{v}_i}{\sum_{i=1}^{N} h_{ik}^t}, \tag{6.4}$$

which is the average of all vectors in the cluster k.

The iterative process above continues until the algorithm converges.

K-means as a probabilistic algorithm. We can turn the distance measure for k-means into a probability distribution. In particular, rewrite Eq 6.2 as follows:

$$\begin{aligned}
\min L(\Theta) &= \min \sum_{i=1}^{N}\sum_{k=1}^{K} h_{ik}||\vec{v}_i - \vec{c}_k||^2 \qquad \text{(minimising loss)} \\
&= \max \sum_{i=1}^{N}\sum_{k=1}^{K} -h_{ik}||\vec{v}_i - \vec{c}_k||^2 \quad \text{(maximising negative loss)} \\
&= \max \sum_{i=1}^{N} \log e^{-\sum_{k=1}^{K} h_{ik}||\vec{v}_i - \vec{c}_k||^2} \quad (x = \log e^x) \\
&= \max \sum_{i=1}^{N} \log \frac{e^{-\sum_{k=1}^{K} h_{ik}||\vec{v}_i - \vec{c}_k||^2}}{Z} \quad (Z \text{ is a constant}) \\
&= \max \sum_{i=1}^{N} \log P(\vec{v}_i, h_i|\Theta),
\end{aligned} \tag{6.5}$$

where $P(\vec{v}_i, h_i|\Theta)$ is a probability distribution using h_i to denote $h_{i1}, h_{i2}, ..., h_{ik}$, and Z is a normalising constant, where $Z = \sum_{\hbar'} \exp(-\sum_{k=1}^{K} \hbar'_{ik}||\vec{v}_i - \vec{c}_k||^2)$. Here $\hbar'$ is a valid cluster assignment, with only one h'_{ik} ($k \in [1, ..., K]$) being 1, and the others being 0. Therefore,

$$\begin{aligned}
P(\vec{v}_i, h_i|\Theta) &= P(\vec{v}_i, h_{i1}, h_{i2}, \ldots, h_{iK}|\Theta) \\
&= \frac{e^{-\sum_{k=1}^{K} h_{ik}||\vec{v}_i - \vec{c}_k||^2}}{Z} \\
&= \mathcal{N}(\sum_{k=1}^{K} h_{ik}\vec{v}_i, I)
\end{aligned} \tag{6.6}$$

is a multi-variant Gaussian distribution, where I is an identity matrix.

Similarly, the finding of cluster centroids can be written as

$$\arg\min_{H} \sum_{i=1}^{N}\sum_{k=1}^{K} h_{ik}||\vec{v}_i - \vec{c}_k||^2 = \arg\min_{H} \sum_{i=1}^{N} P(\vec{v}_i, h_i|\Theta)$$

K-means and EM. Given the probabilistic interpretation of each step in k-means, we can draw correlation between k-means and Eq 6.1. Formally, k-means optimises in alternation

$$\begin{aligned}
\mathcal{H} &= \arg\max_{\mathcal{H}'} P(O, H = \mathcal{H}'|\Theta) = \arg\max_{\mathcal{H}'} \sum_{i=1}^{N} P(\vec{v}_i, h_i = \hbar'_i|\Theta), \\
\Theta &= \arg\max_{\Theta} P(O, H = \mathcal{H}|\Theta) = \arg\max_{\Theta} \sum_{i=1}^{N} P(\vec{v}_i, h_i = \hbar_i|\Theta)
\end{aligned} \tag{6.7}$$

Algorithm 6.1: K-means as a "hard" EM algorithm.

Inputs: observed data $O = \{\vec{v}_i\}|_{i=1}^N$;
Hidden Variables: $H = \{h_i\}|_{i=1}^N$;
Initialisation: model $\Theta^0 \leftarrow$ RandomModel(), $t \leftarrow 0$;
repeat
  Expectation step:
    $H^t \leftarrow \arg\max_H \log P(O, H|\Theta^t)$; ▷ `Eq 6.3`;
  Maximisation step:
    $\Theta^{t+1} \leftarrow \arg\max_\Theta \log P(O, H^t|\Theta)$; ▷ `Eq 6.4`;
  $t \leftarrow t + 1$;
until Converge(H, Θ);

Algorithm 6.1 shows the algorithm. The model parameters Θ are randomly initialised to Θ^0. Θ is then optimised using an iterative process. At the tth iteration, Θ^t is fixed first and H^t is predicted by finding the most likely values for the hidden variables. Then, H^t is kept unchanged and acts as training labels at the maximisation step. MLE can be used to find Θ^{t+1} for the next iteration. This algorithm can be viewed as a "hard" EM algorithm, differing from the EM algorithm in the calculation of expectation. The next section discusses EM by making a change to this algorithm.

6.1.2 Expectation Maximisation

Different from k-means (or hard EM), which uses the most likely values of hidden variables as the expectations of the variables, the expectation maximisation (EM) algorithm calculates the hidden variable distributions by considering all possible values of hidden variables. This applies to both the expectation (E) step and the maximisation (M) step.

E-step. Rather than predicting only one expected value of H, the expectation step of EM produces a distribution of H. Denote this distribution as P_C, where $P_C(H = \mathcal{H})$ (or $P_C(\mathcal{H})$ in brief) is the probability of a specific value $\mathcal{H}$ from the set of all possible values for the hidden variables H. In particular, the distribution $P_C(H)$ is defined as $P(H|O, \Theta)$, which is the posterior distribution of H given observations O and the model parameters Θ (we will see the reason in Section 6.3). For example, under the k-means task setting, the E-step would calculate $P(h_i|\vec{v}_i, \Theta)$ for all the K possible cluster assignments h_i given each vector $\vec{v}_i$, which can be $\frac{||\vec{v}_i - \vec{c}_k||^2}{2}$ for $h_i =$ OneHot(k) ($k \in [1, ..., K]$). Ex 6.4 discusses this in further detail. We will see more examples in addition to k-means in Section 6.2.

Algorithm 6.2: Expectation maximisation.

Inputs: data $O = \{o_i\}|_{i=1}^N$;
Hidden Variables: $H = \{h_i\}|_{i=1}^N$;
Initialisation: model $\Theta^0 \leftarrow$ RANDOMMODEL(), $t \leftarrow 0$;
repeat
  Expectation step:
    Compute $P(H = \mathcal{H}|O, \Theta^t)$ for each possible $\mathcal{H}$ (i.e., $P(h_i = \hbar|o_i, \Theta^t)$ for each value $\hbar$ of h_i ($i \in [1, ..., N]$));
  Maximisation step:
    $Q(\Theta, \Theta^t) \leftarrow \sum_{\mathcal{H}} P(H = \mathcal{H}|O, \Theta^t) \log P(O, H = \mathcal{H}|\Theta)$ (i.e., $\sum_{i=1}^N \sum_{\hbar} P(\hbar|o_i, \Theta^t) \log P(o_i, \hbar|\Theta)$);
    $\Theta^{t+1} \leftarrow \arg\max_{\Theta} Q(\Theta, \Theta^t)$;
  $t \leftarrow t + 1$;
until CONVERGE (H, Θ);

M-step. The maximisation step optimises Θ by maximising the expected log-likelihood $\log P(O, H|\Theta)$ given the distribution $P(H|O, \Theta)$,

$$\begin{aligned} \hat{\Theta} &= \arg\max_{\Theta} E_{H \sim P(H|O,\Theta)} \log P(O, H|\Theta) \\ &= \arg\max_{\Theta} \sum_{\mathcal{H}} P(H = \mathcal{H}|O, \Theta) \log P(O, \mathcal{H}|\Theta) \end{aligned} \tag{6.8}$$

In the above equation, $P(H = \mathcal{H}|O, \Theta)$ are the values of $P_C(\mathcal{H})$ for each possible hidden variable $\mathcal{H}$, as calculated in the E-step. As a result, they are fixed in finding $\arg\max_{\Theta}$. The parameter Θ to adjust exists only in $P(O, \mathcal{H}|\Theta)$. The M-step is typically a constrained optimisation process, for which Lagrange techniques can be used, as we have discussed in Chapter 5. We will see detailed examples in Section 6.2.

Algorithm 6.2 shows pseudocode of the EM algorithm over a dataset $O = \{o_i\}|_{i=1}^N$, where each o_i is associated with a set of hidden variables h_i. The algorithm starts from a randomly initialised model, repeatedly performing expectation and maximisation steps until convergence. Here CONVERGE can be defined as returning true when the Θ values between two iterations become similar by a certain measure, and false otherwise. The iteration number is denoted as a variable t. Variable values determined in a certain iteration t are denoted with t as a superscript and taken as constants. Free variables are denoted without t. In particular, in the maximisation step to find Θ^{t+1}, Θ^t in the objective function $Q(\Theta, \Theta^t)$ is regarded a constant while Θ a variable to optimise.

Given a random start, it has been shown that EM can always reach a local optimum of the model training objective. However, local optima can be highly different from the global optimum. As a result, the initial value Θ^0 has a large influence on the final model.

Q function and EM. In Algorithm 6.2, the objective function $\mathrm{Q}(\Theta, \Theta^t)$ for the M-step is:

$$Q(\Theta, \Theta^t) = \sum_{\mathcal{H}} P(H = \mathcal{H}|O, \Theta^t) \log P(O, H = \mathcal{H}|\Theta), \tag{6.9}$$

which is called a **Q-function**. This function is central to EM. We can view EM as iteratively optimising this function. As mentioned earlier, this training objective is the expectation of the joint likelihood to optimise, and therefore can also be called the *expectation function*.

The Q-function can be regarded as a weighted version of Eq 6.1

$$\hat{\Theta} = \arg\max_{\Theta} \sum_{\mathcal{H}} w_{\mathcal{H}} \log P(O, H = \mathcal{H}|\Theta),$$

where $w_{\mathcal{H}}$ is the corresponding weight for the specific hidden variable assignment $\mathcal{H}$. In particular, $w_{\mathcal{H}} = P(H = \mathcal{H}|O, \Theta)$ is a constant value found in the E-step. Now if in the training data, each o_i is manually given a gold-standard label y_i, we can define

$$P(h_i|o_i, \Theta^t) = \begin{cases} 1 \text{ if } h_i = y_i \\ 0 \text{ otherwise,} \end{cases} \tag{6.10}$$

in which case the Q-function in Eq 6.9 becomes:

$$\begin{aligned} Q(\Theta, \Theta^t) &= \sum_{i=1}^{N} \sum_{\hbar} P(\hbar|o_i, \Theta^t) \log P(o_i, \hbar|\Theta) \\ &= \sum_{i=1}^{N} \log P(o_i, y_i|\Theta), \end{aligned}$$

which is exactly the maximum log-likelihood training objective.

This also shows the importance of $P(h|o, \Theta)$ to the success of EM training. With complete knowledge of $P(H|O, \Theta)$ such as Eq 6.10, EM becomes equivalent to fully supervised learning. The parameterisation of $P(H|O, \Theta)$ can convey prior knowledge about the problem to solve, which guides the learning of H through the optimisation of O. For example, in the k-means algorithm, we draw connections between O and H by knowing the correlation between cluster centroids and cluster assignments. In this light, the initial values of Θ also plays a crucial role in the success of EM.

Using the Q-function as a training objective, every possible value of the hidden variables can make a contribution to the parameter update. In Section 6.3, we will give theoretical justifications of Algorithm 6.2, showing that this algorithm is guaranteed to converge.

6.2 Using EM for Training Models with Hidden Variables

We discuss three examples of EM training, namely the unsupervised Naïve Bayes model, IBM model 1 for machine translation and probabilistic latent semantic analysis (PLSA). For each task, we follow Algorithm 6.2 to: (1) parameterise the complete data likelihood $P(O, H|\Theta)$, (2) compute $P(H|O, \Theta)$, (3) maximise $Q(\Theta, \Theta^t)$, where (2) and (3) are executed iteratively.

6.2.1 Unsupervised Naïve Bayes Model

Let us return to Naïve Bayes models for text classifications. In Chapter 2, we discussed supervised settings. Given a set of documents and their corresponding labels $D = \{(d_i, c_i)\}|_{i=1}^N$, where $c_i \in C$, $d_i = \{w_1^i, w_2^i, \ldots, w_{|d_i|}^i\}$, and $w_j^i \in V$ denotes the jth word in document d_i, MLE is used to estimate the model parameter by counting relative frequencies. In particular, we have:

$$\begin{aligned} P(c) &= \frac{\sum_{i=1}^N \delta(c_i, c)}{N} \\ P(w|c) &= \frac{\sum_{i=1}^N \left(\delta(c_i, c) \cdot \sum_{j=1}^{|d_i|} \delta(w_j^i, w)\right)}{\sum_{i=1}^N \delta(c_i, c)|d_i|} \end{aligned} \tag{6.11}$$

for each $w \in V$ and $c \in C$. $\delta(c_i, c)$ tests whether c_i equals c.

Now let us consider unsupervised settings, where the output classes are not available. In this case, the inputs are still documents d but the outputs are hidden class labels h. Similar to k-means, let us suppose that there are K document classes. Since we do not have control over the meaning of each class, we can denote the set of class labels as $C = \{1, 2, ..., K\}$. Now given a set of documents $D = \{d_i\}|_{i=1}^N$, for each document d_i, the data likelihood with each possible class $\hbar \in C$ is:

$$P(d_i, \hbar|\Theta) = P(\hbar|\Theta)P(d_i|\hbar, \Theta) = P(\hbar|\Theta) \prod_{j=1}^{|d_i|} P(w_j^i|\hbar, \Theta) \tag{6.12}$$

In the above equation, $P(d_i, h|\Theta)$ is the main model, where the parameters Θ consist of $P(\hbar)$ and $P(w|\hbar)$ for all $w \in V$ and $\hbar \in C$. Following Naïve Bayes, we assume that each word in conditionally independent given $\hbar$ and Θ. As a result, the model is a bag-of-words model and hence the name unsupervised Naïve Bayes model.

We use EM as shown in Algorithm 6.2 to train this model, which iteratively calculates $P(\hbar|d)$ for all $\hbar \in C$ and maximises $\sum_{i=1}^N \sum_{\hbar} P(\hbar|d_i) \log P(d_i, \hbar)$. Similar to Section 6.1.2 and Section 6.3, Θ is included in the conditional probability in order to denote parameter and hidden variable values at a

certain iteration number. In particular, $P(\hbar|d_i)$ becomes $P(\hbar|d_i,\Theta)$ and $P(d_i,\hbar)$ becomes $P(d_i,\hbar|\Theta)$. Θ^0 are randomly initialised, and at iteration t, one E-step is first taken, where $P(\hbar|d_i,\Theta^t)$ is calculated by

$$P(\hbar|d_i,\Theta^t)=\frac{P(d_i,\hbar|\Theta^t)}{\sum_{\hbar\in C}P(d_i,\hbar|\Theta^t)}=\frac{P(\hbar|\Theta^t)\prod_{i=1}^{|d_i|}P(w_i|\hbar,\Theta^t)}{\sum_{\hbar\in C}P(\hbar|\Theta^t)\prod_{i=1}^{|d_i|}P(w_i|\hbar,\Theta^t)} \quad (6.13)$$

We then take one M-step to maximise $Q(\Theta,\Theta^t)$, namely:

$$Q(\Theta,\Theta^t)=\sum_{i=1}^{N}\sum_{\hbar\in C}P(\hbar|d_i,\Theta^t)\log P(d_i,\hbar|\Theta)$$

As discussed in Section 6.1.2, to find $\arg\max_\Theta Q(\Theta,\Theta^t)$ s.t. $\sum_{\hbar\in C}P(\hbar|\Theta)=1$ and $\sum_{w\in V}P(w|\hbar,\Theta)=1$, we use Lagrangian optimisation, where the Lagrangian function with the constraints is

$$\begin{aligned}\Lambda(\Theta,\lambda)&=Q(\Theta,\Theta^t)-\lambda_1\Big(\sum_{\hbar\in C}P(\hbar|\Theta)-1\Big)-\sum_{\hbar\in C}\lambda_\hbar\Big(\sum_{w\in V}P(w|\hbar,\Theta)-1\Big)\\&=\sum_{i=1}^{N}\sum_{\hbar\in C}P(\hbar|d_i,\Theta^t)\log P(d_i,\hbar|\Theta)-\lambda_1\Big(\sum_{\hbar\in C}P(\hbar|\Theta)-1\Big)\\&\quad-\sum_{\hbar\in C}\lambda_\hbar\Big(\sum_{w\in V}P(w|\hbar,\Theta)-1\Big)\\&=\sum_{i=1}^{N}\sum_{\hbar\in C}P(\hbar|d_i,\Theta^t)\Big(\log P(\hbar|\Theta)+\sum_{j=1}^{|d_i|}\log P(w_j|\hbar,\Theta)\Big)\\&\quad-\lambda_1\Big(\sum_{\hbar\in C}P(\hbar|\Theta)-1\Big)-\sum_{\hbar\in C}\lambda_\hbar\Big(\sum_{w\in V}P(w|\hbar,\Theta)-1\Big)\end{aligned}$$

Taking partial derivatives of $\Lambda(\Theta,\lambda)$ with respect to $P(\hbar|\Theta)$ gives

$$\frac{\partial\Lambda(\Theta,\lambda)}{\partial P(\mathcal{H}|\Theta)}=\frac{\sum_{i=1}^{N}P(\hbar|d_i,\Theta^t)}{P(\hbar|\Theta)}-\lambda_1$$

Letting $\frac{\partial\Lambda(\Theta,\lambda)}{\partial P(\hbar|\Theta)}=0$, we have

$$P(\hbar|\Theta)=\frac{\sum_{i=1}^{N}P(\hbar|d_i,\Theta^t)}{\lambda_1}$$

Under the constraint that $\sum_{\hbar\in C}P(\hbar|\Theta)=1$, we have

$$\sum_{\hbar\in C}P(\hbar|\Theta)=\sum_{\hbar\in C}\frac{\sum_{i=1}^{N}P(\hbar|d_i,\Theta^t)}{\lambda_1}$$

and thus

$$\lambda_1=\sum_{\hbar\in C}\sum_{i=1}^{N}P(\hbar|d_i,\Theta^t)=\sum_{i=1}^{N}\sum_{\hbar\in C}P(\hbar|d_i,\Theta^t)=N$$

Therefore we have

$$P(\hbar|\Theta) = \frac{\sum_{i=1}^{N} P(\hbar|d_i, \Theta^t)}{N}, \tag{6.14}$$

which is taken as the value of $P(\hbar|\Theta^{t+1})$ for the next iteration. Similarly,

$$\frac{\partial \Lambda(\Theta, \lambda)}{\partial P(w|\hbar, \Theta)} = \frac{\sum_{i=1}^{N} P(\hbar|d_i, \Theta^t) \sum_{j=1}^{|d_i|} \delta(w_j, w)}{P(w|\hbar, \Theta)} - \lambda_\hbar,$$

where the delta function $\delta(w_j, w)$ connects $w_j \in d_i$ with $w \in V$.

Given $\frac{\partial \Lambda(\Theta,\lambda)}{\partial P(w|\hbar,\Theta)} = 0$ and $\sum_{w \in V} P(w|\hbar, \Theta) = 1$, we have

$$\lambda_\hbar = \sum_{w \in V} \sum_{i=1}^{N} P(\hbar|d_i, \Theta^t) \sum_{j=1}^{|d_i|} \delta(w_j, w) = \sum_{i=1}^{N} P(\hbar|d_i, \Theta^t)|d_i|$$

$$P(w|\hbar, \Theta) = \frac{\sum_{i=1}^{N} P(\hbar|d_i, \Theta^t) \sum_{j=1}^{|d_i|} \delta(w_j, w)}{\lambda_\hbar} = \frac{\sum_{i=1}^{N} P(\hbar|d_i, \Theta^t) \sum_{j=1}^{|d_i|} \delta(w_j, w)}{\sum_{i=1}^{N} P(\hbar|d_i, \Theta^t)|d_i|}, \tag{6.15}$$

which is taken as the value of $P(w|\hbar, \Theta^{t+1})$ for the next iteration.

The above process is executed iteratively, taking the expectation step using Eq 6.13 and the maximisation step using Eq 6.14 and Eq 6.15 until the differences between $P(\hbar|\Theta^{t+1})$ and $P(\hbar|\Theta^t)$ and between $P(\hbar|\Theta^{t_0})$ and $P(\hbar|\Theta^t)$ are below a threshold. After convergence, $P(\hbar|\Theta)$ and $P(w|\hbar, \Theta)$ are taken as the final model.

Unsupervised Naïve Bayes v.s. Naïve Bayes. Comparing Eq 6.14 and Eq 6.15 with Eq 6.11, it is easy to find that the terms $P(c)$ v.s. $P(\hbar|\Theta)$ and $P(w|c)$ v.s. $P(w|\hbar, \Theta)$ are quite similar. In particular, for the ith example, setting $P(c|d_i, \Theta^t) = \delta(c_i, c)$ makes Eq 6.14 and Eq 6.15 identical to Eq 6.11. This is very intuitive. In supervised settings, $\sum_{i=1}^{N} \delta(c_i, c)$ in Eq 6.11 is the total actual count for the label c, while $\sum_{i=1}^{N} P(\hbar|d_i, \Theta^t)$ denotes the expected count for label $\hbar$ in unsupervised settings. Similar in Eq 6.14, explanations can be made between $\sum_{i=1}^{N} \left(\delta(c_i, c) \cdot \sum_{j=1}^{|d_i|} \delta(w_j^i, w)\right)$ in Eq 6.11 and $\sum_{i=1}^{N} P(\hbar|d_i, \Theta^t) \sum_{j=1}^{|d_i|} \delta(w_j, w)$ in Eq 6.15.

Unsupervised Naïve Bayes v.s. K-means. Unsupervised Naïve Bayes is a clustering model for documents. Compared with the k-means clustering algorithm in Chapter 2, there are two main differences. First, k-means is based on vector space geometry, finding a partition of vector space based on vector points and Euclidean distances. In contrast, Naïve Bayes is a direct probability model of documents and words. Second, an unsupervised Naïve Bayes model is optimised with EM, while k-means is a hard variant of EM. Note that similar to k-means clustering, there is no direct interpretation of each class label by unsupervised classification. The automatically-induced

classes do not necessarily correspond to a specific set of class labels such as {Word, Leisure} or {Travel, Non-Travel}. Manual inspection of documents in each class is typically necessary for understanding each class label.

6.2.2 IBM Model 1

Given a *source* sentence X, the task of machine translation (MT) is to find a corresponding *target* language translation Y. Here $X = x_1 x_2 \dots x_{|X|}$, where x_i ($i \in [1, \dots, |X|]$) is a source word, and $Y = y_1 y_2 \dots y_{|Y|}$, where y_j ($j \in [1, \dots, |Y|]$) is a target word. A simple probabilistic model for machine translation calculates $P(Y|X)$, namely the probability of a candidate target translation Y given a source sentence X. Since both X an Y are sentences, which are highly sparse, we parameterise the model by defining a generative story using the techniques in Chapter 2.

A probabilistic model for MT. According to the Bayes rule, we have

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \propto P(X|Y)P(Y) \tag{6.16}$$

In Eq 6.16, $P(Y)$ is the probability of the candidate target sentence, and therefore serves as a *language model* to ensure fluency. $P(X|Y)$ is the probability of X given Y, which serves as a *translation model* to ensure adequacy. The above generative story yields a sentence pair (X, Y) by first yielding Y, and then yielding X given Y. This design offers modularity by introducing a fluency component $P(Y)$, compared to a model for $P(Y|X)$ directly, without using the Bayes rule.

Next we simplify $P(X|Y)$, deriving model parameters by using the probability chain rule and making independent assumptions. According to the chain rule, we have

$$\begin{aligned} P(X|Y) =& P(x_1 x_2 \dots x_{|X|} | y_1 y_2 \dots y_{|Y|}) \\ =& P(x_1 | y_1 y_2 \dots y_{|Y|}) P(x_2 | x_1 y_1 \dots y_{|Y|}) \dots \\ & P(x_{|x|} | x_1 \dots x_{|X|-1}, y_1 \dots y_{|Y|}) \end{aligned}$$

Further assuming that each source word x_i is conditionally dependent to only one target word y_{a_i}, we have

$$\begin{aligned} P(X|Y) =& P(x_1 | y_1 y_2 \dots y_{|Y|}) P(x_2 | x_1 y_1 \dots y_{|Y|}) \dots \\ & P(x_{|X|} | x_1 \dots x_{|X|-1}, y_1 \dots y_{|Y|}) \\ =& P(x_1 | y_{a_1}) P(x_2 | y_{a_2}) \dots P(x_{|X|} | y_{a_{|X|}}) \end{aligned}$$

Here a_i denotes the index of the target word that the ith source word translates to. Given two sentences X and Y, the set $A = \{a_i\}|_{i=1}^{|X|}$ is referred to as their **word alignment**. There are several types of word alignments

ID	Source	Target	Alignment
1 (*French*)	J'$_1$(I) aime$_2$(like) lire$_3$(reading)	I$_1$ like$_2$ reading$_3$	{1→1, 2→2, 3→3}
2 (*German*)	Ich$_1$(I) lese$_2$(read) hier$_3$(here) ein$_4$(a) Buch$_5$(book)	I$_1$ read$_2$ a$_3$ book$_4$ here$_5$	{1→1, 2→2, 3→5, 4→3, 5→4 }
3 (*Chinese*)	我$_1$ (I) 在$_2$ (at) 这里$_3$ (here) 读$_4$ (read) 一$_5$ (a) 本$_6$ (this) 书$_7$ (book)	I$_1$ read$_2$ a$_3$ book$_4$ here$_5$	{1→1, 2→5, 3→5, 4→ 2, 5→3, 6→NULL, 7→4}
4 (*Japanese*)	私は$_1$(I) 家で$_2$(at home) 本を$_3$(a book) 読む$_4$(read)	I$_1$ read$_2$ a$_3$ book$_4$ at$_5$ home$_6$	{1→1, 2→{5, 6}, 3→{3, 4}, 4→2}

Table 6.1: Word alignment examples.

between sentence translation pairs, as shown in Table 6.1. In particular, the alignment in example 1 is *monotonic*, with $a_i = i$. The alignment in example 2, however, is *non-monotonic*, with $a_1 = 1$, $a_2 = 2$, $a_3 = 5$, $a_4 = 3$ and $a_5 = 4$. Example 3 contains *many-to-one* alignments (i.e., $a_2 = a_3 = 5$) and *null* alignments (i.e., $a_6 = \text{NULL}$). Example 4 contains *one-to-many* alignments, with $a_2 = \{5, 6\}$ and $a_3 = \{3, 4\}$. To simplify our model, we treat one-to-many alignments as illegal, considering a as a function mapping from a source index to a unique target index or NULL.

The final model $P(Y|X)$ therefore consists of two main parameter types, namely word translation probabilities $P(x|y)$ for $P(X|Y)$ and language model parameters (e.g., a trigram LM) for $P(Y)$. $P(x|y)$ denotes the probability of a source *vocabulary* word $x \in V_x$ given a target *vocabulary* word $y \in V_y$, which is similar to a dictionary with probability values. Below we focus on $P(X|Y)$ since language models have been discussed in Chapter 2.

EM training. Datasets that consist of sentence translation pairs $D = \{(X_i, Y_i)\}|_{i=1}^N$ are relatively easy to obtain. On the other hand, gold-standard word alignments are extremely costly to obtain. As a result, given a sentence translation pair (X_i, Y_i), the word alignment A_i between them has to be treated as a hidden variable. In this case, the observed variables are $O = (X_i, Y_i)$ and the hidden variables are $H = \{A_i\}$.

If A_i are known, the training of our translation model can be achieved using standard MLE by counting relative frequencies, where

$$P(x|y) = \frac{\#(x \text{ aligned to } y \text{ in } D)}{\#(y \text{ in } D)}$$

In the reverse direction, if the model $P(x|y)$ is given, one can calculate

the expected values of A_i. Consequently, EM can be used to derive our model, with Θ being $P(x|y)$, H being A_i, the E-step being the process of deriving a model distribution of A_i using $P(x|y)$, and the M-step being the estimation of $P(x|y)$ using the distribution of A_i.

For notational convenience, let us use one specific training instance to illustrate the EM process, denoting the sentence pair as $O = (X, Y)$ and the alignment as $H = A$. According to Algorithm 6.2, at each iteration t, the expectation step calculates $P(H|O, \Theta^t)$ and the maximisation step maximises $\sum_H P(H|O, \Theta^t) \log P(O, H|\Theta) = \sum_A P(A|X, Y, \Theta^t) \log P(X, A|Y, \Theta)$. We now need to denote $P(A|X, Y)$ and $P(X, A|Y)$ in terms of $P(x|y)$, our translation model parameters. In particular,

$$P(A|X, Y) = \frac{P(A, X|Y)}{P(X|Y)} \text{ (Eq 2.9; conditioned on } Y)$$

$P(A, X|Y)$ (i.e., $P(X, A|Y)$) can be further decomposed into

$$P(A, X|Y) = P(A|Y)P(X|A, Y) \tag{6.17}$$

Thus for calculating the joint distribution $P(A, X|Y)$, we can calculate $P(A|Y)$ first, which are the probabilities of generating A given Y, and then $P(X|A, Y)$, which are the probabilities of generating X given Y and A.

For a simplest model, assume that given a target sentence, the alignment of a source word to each target word in the sentence is equally probable. Further since we assume that each x_i is aligned to exactly one y_j or NULL,

$$P(A|Y) = (\frac{1}{|Y| + 1})^{|X|} = \frac{1}{(|Y| + 1)^{|X|}}, \tag{6.18}$$

where 1 accounts for NULL in $(|Y| + 1)$.

$P(X|A, Y)$ is straightforward to calculate when the alignment A and the target Y are both known. Assuming that each x_i is generated independently of all the other x'_i $(i' \neq i)$, x_i depends only on the word y_{a_i}, we have

$$P(X|A, Y) = \prod_{i=1}^{|X|} P(x_i|y_{a_i}), \tag{6.19}$$

and therefore substituting Eq 6.18 and Eq 6.19 into Eq 6.17 we have

$$\begin{aligned} P(A, X|Y) &= P(A|Y)P(X|A, Y) \\ &= \frac{\prod_{i=1}^{|X|} P(x_i|y_{a_i})}{(|Y| + 1)^{|X|}} \end{aligned}$$

We further calculate $P(X|Y)$ by marginalising out A

$$\begin{aligned}
P(X|Y) &= \sum_A P(A, X|Y) \\
&= \sum_A \frac{\prod_{i=1}^{|X|} P(x_i|y_{a_i})}{(|Y|+1)^{|X|}} \\
&= \sum_{a_1=0}^{|Y|} \sum_{a_2=0}^{|Y|} \cdots \sum_{a_{|X|}=0}^{|Y|} \frac{\prod_{i=1}^{|X|} P(x_i|y_{a_i})}{(|Y|+1)^{|X|}} \\
&= \frac{1}{(|Y|+1)^{|X|}} \sum_{a_1=0}^{|Y|} \sum_{a_2=0}^{|Y|} \cdots \sum_{a_{|X|}=0}^{|Y|} \prod_{i=1}^{|X|} P(x_i|y_{a_i}) \\
&= \frac{1}{(|Y|+1)^{|X|}} \prod_{i=1}^{|X|} \sum_{j=0}^{|Y|} P(x_i|y_j), \quad \text{(distributivity)}
\end{aligned} \tag{6.20}$$

where $a_i = 0$ denotes that the ith source word is aligned to NULL (i.e., we have a special target word y_0=NULL). The last step above uses the law of distributivity to transform a sum of products to a product of sums, the proof of which is left for Ex 6.6. With this change, an exponential number of products are reduced to a linear number of sums, which is computationally tractable.

The alignment probability $P(A|X, Y)$ is given by

$$\begin{aligned}
P(A|X, Y) &= \frac{P(A, X|Y)}{P(X|Y)} \\
&= \frac{\prod_{i=1}^{|X|} P(x_i|y_{a_i})}{\prod_{i=1}^{|X|} \sum_{j=0}^{|Y|} P(x_i|y_j)} \\
&= \prod_{i=1}^{|X|} \frac{P(x_i|y_{a_i})}{\sum_{j=0}^{|Y|} P(x_i|y_j)}
\end{aligned}$$

We have now obtained the form of $P(A|X, Y)$ and $P(A, X|Y)$ in terms of our model parameter $P(x|y)$, which can serve as a basis for the E-step. Further, for a time step t, the M-step optimises Θ by maximising:

$$\begin{aligned}
Q(\Theta, \Theta^t) &= \sum_A P(A|X, Y, \Theta^t) \log P(A, X|Y, \Theta) \\
&= \sum_A P(A|X, Y, \Theta^t) \log \frac{\prod_{i=1}^{|X|} P(x_i|y_{a_i}, \Theta)}{(|Y|+1)^{|X|}}
\end{aligned} \tag{6.21}$$

Here we introduce Θ to the probabilities in Eq 6.17, so that we can explicitly use Θ and Θ^t to denote adjusted and fixed model parameters,

respectively. Given the probability constraint $\sum_x P(x|y) = 1$ for every y, we can define a Lagrangian function

$$\begin{aligned}\Lambda(\Theta, \lambda) &= Q(\Theta, \Theta^t) - \sum_y \lambda_y \Big(\sum_x P(x|y, \Theta) - 1 \Big) \\ &= \sum_A P(A|X, Y, \Theta^t) \log \frac{\prod_{i=1}^{|X|} P(x_i|y_{a_i}, \Theta)}{(|Y|+1)^{|X|}} - \sum_y \lambda_y \Big(\sum_x P(x|y, \Theta) - 1 \Big) \\ &= \sum_A P(A|X, Y, \Theta^t) \Big(\sum_{i=1}^{|X|} \log P(x_i|y_{a_i}, \Theta) - |X| \log(|Y|+1) \Big) \\ &\quad - \sum_y \lambda_y \Big(\sum_x P(x|y, \Theta) - 1 \Big)\end{aligned}$$

Taking derivatives of $\Lambda(\Theta, \lambda)$ with respect to $P(x|y, \Theta)$, we have

$$\frac{\partial \Lambda(\Theta, \lambda)}{\partial P(x|y, \Theta)} = \frac{\sum_A P(A|X, Y, \Theta^t) \sum_{k=1}^{|X|} \delta(x, x_k) \delta(y, y_{a_k})}{P(x|y, \Theta)} - \lambda_y$$

Setting $\frac{\partial \Lambda(\Theta, \lambda)}{\partial P(x|y, \Theta)} = 0$ gives

$$\begin{aligned}P(x|y, \Theta) &= \frac{\sum_A P(A|X, Y, \Theta^t) \sum_{k=1}^{|X|} \delta(x, x_k) \delta(y, y_{a_k})}{\lambda_y} \\ &\propto \sum_A P(A|X, Y, \Theta^t) \sum_{k=1}^{|X|} \delta(x, x_k) \delta(y, y_{a_k})\end{aligned}$$

which can serve as $P(x|y, \Theta^{t+1})$ in the next iteration for EM.

So far we only consider a single sentence pair. Moving back to the whole corpus, the summation should be calculated over all sentence pairs:

$$\sum_{(X_i, Y_i) \in D} \sum_{A_i} P(A_i|X_i, Y_i, \Theta^t) \sum_{k=1}^{|X|} \delta(x, x_k^i) \delta(y, y_{a_k}^i)$$

To derive pseudocode for the above process, let us define

$$\begin{aligned}\textsc{ExpectedAlign}(x, y, X, Y) &= \sum_A P(A|X, Y) \cdot \sum_{k=1}^{|X|} \delta(x, x_k) \delta(y, y_{a_k}) \\ &= E_{A \sim P(A|X,Y)}[\sum_{k=1}^{|X|} \delta(x, x_k) \delta(y, y_{a_k})],\end{aligned}$$

which is the *expected* alignment between a word pair of source and target vocabulary words x and y in a sentence translation pair $(X,\ Y)$ given a

model $P(x|y)$. To calculate its value, we have

$$\begin{aligned}\textsc{ExpectedAlign}(x, y, X, Y) &= \sum_A P(A|X, Y) \cdot \sum_{k=1}^{|X|} \delta(x, x_k)\delta(y, y_{a_k}) \\ &= \sum_A \prod_{i=1}^{|X|} \frac{P(x_i|y_{a_i})}{\sum_{j=0}^{|Y|} P(x_i|y_j)} \cdot \sum_{k=1}^{|X|} \delta(x, x_k)\delta(y, y_{a_k}) \\ &= \frac{P(x|y)}{\sum_{j=0}^{|Y|} P(x|y_j)} \sum_{i=1}^{|X|} \delta(x, x_i) \sum_{j=0}^{|Y|} \delta(y, y_j)\end{aligned} \tag{6.22}$$

The last step of Eq 6.22 can be derived using similar tricks as Eq 6.20, which we leave for Ex 6.7. Intuitively, $\sum_{i=1}^{|X|} \delta(x, x_i) \sum_{j=0}^{|Y|} \delta(y, y_j)$ is the total alignment count of the word x and the word y in the sentence X and the sentence Y, and $\frac{P(x|y)}{\sum_{j=0}^{|Y|} P(x|y_j)}$ is a weight probability score. In this respect, $\textsc{ExpectedAlign}(x, y, X, Y)$ represents a soft count.

For the maximisation step, we take this expected count as a real count and perform MLE to obtain the Θ^{t+1} value that maximises $Q(\Theta, \Theta^t)$. In particular, for a source vocabulary word x and a target vocabulary word y,

$$P(x|y) = \frac{\sum_{(X_i,Y_i)\in D} \textsc{ExpectedAlign}(x, y; X_i, Y_i)}{\sum_{(X_i,Y_i)\in D} \left(\sum_{x'} \textsc{ExpectedAlign}(x', y; X_i, Y_i)\right)}$$

The equation above can be implemented using Algorithm 6.3, where $\textsc{Converge}(P(x|y))$ can be calculated based on the perplexity (mentioned in Chapter 5) of the model over the corpus D. In particular, the perplexity of a translation model can be defined as

$$\Upsilon(P) = 2^{-\sum_{i=1}^{N} \log_2 P(X_i|Y_i)},$$

where (X_i, Y_i) is a sentence pair in the bilingual corpus D. During training $\upsilon(P)$ decreases. When $\Upsilon(P)$ becomes stable, the algorithm can stop.

The model above was named *IBM model 1*, which is the simplest among 5 probabilistic models for statistical machine translation developed at IBM in the early 1990s. IBM models are *word-based* machine translation models, performing translation word by word. It has been discovered in the 2000s that translating a source sentence phrase-by-phrase gives significantly better translation quality. As a result, *phrase-based* translation systems replaced word-based translation systems as the state-of-the-art at that time. The dominant approach was further overtaken by neural machine translation (NMT) in the 2010s, which will be discussed in Chapter 16.

Algorithm 6.3: Word alignment.

Input: D $= \{(X_i, Y_i)\}|_{i=1}^{N}$;
Variables: *count*$(x|y)$; *count*(y); *sent-total*(x);
Initillisation $p(x|y) \leftarrow$ UniformDistribution();
repeat
  count$(x|y) \leftarrow 0$;
  count$(y) \leftarrow 0$;
  for $(X, Y) \in D$ **do**
    for $x_i \in X$ **do**
      sent-total$(x_i) \leftarrow 0$;
      for $y_j \in Y_i$ **do**
        sent-total$(x_i) \leftarrow$*sent-total*$(x_i)+p(x_i|y_j)$;
    for $x_i \in X$ **do**
      for $y_j \in Y$ **do**
        count$(x_i|y_j) \leftarrow$*count*$(x_i|y_j) + \frac{p(x_i|y_j)}{\textit{sent-total}(x_i)}$;
        count$(y_j) \leftarrow$*count*$(y_j) + \frac{p(x_i|y_j)}{\textit{sent-total}(x_i)}$;
  for $x \in$ SourceVocab$(D), y \in$ TargetVocab$(D), y \in D$ **do**
    $p(x|y) = \frac{\textit{count}(x|y)}{\textit{count}(y)}$;
until Converge$(p(x|y))$;

6.2.3 Probabilistic Latent Semantic Analysis

We have seen several methods for representing a document into the vector space, such as count-based vectors and TF-IDF vectors. They are high-dimensional vectors. **Latent semantic allocation** is a different method for representing documents, using lower-dimensional vectors, each element of which represents a semantic attribute of the document. To this end, **probabilistic latent semantic analysis** (PLSA) is a generative model, which represents a document by its topic distribution. Given a set of documents $D = \{d_i\}|_{i=1}^{N}$, where d_i consists of $w_1^i, w_2^i, ..., w_{|d_i|}^i$, PLSA assumes that each document d_i contains a mixture of **topics**, where each topic refers to a semantic class such as "politics" or "sports". Following our discussion of k-means and unsupervised Naïve Bayes, let us define the set of topics as $T = [1, ..., K]$. The goal of PLSA is to calculate a multinomial document-topic distribution $P(\hbar|d_i)$ as a dense vector in R^K to represent every document d_i, where each element is the probability of a specific topic value $\hbar \in T$.

In addition to a document-topic correlation, there is a topic-word distribution $P(w|\hbar)$ for each topic $\hbar$. $P(w|\hbar)$ decides the probabilities of vocabulary words under $\hbar$, which reflects the meaning of the topic $\hbar$. For example, if $\hbar$ is highly correlated with words such as "*policy*", "*election*", "*president*",

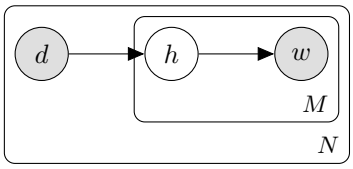

Figure 6.1: PLSA in plate notation.

"*tax*", "*economic*" and "*healthcare*", then $\hbar$ is likely a politics-related topic. In contrast, $\hbar$ tends to be a finance-related topic if $P(w|\hbar)$ is dominated by words such as "*stock*", "*IPO*", "*share*", "*trade*", "*market*" and "*investment*".

Given D, topics are hidden variables. They serve as a link in a generative story that helps to better represent a document. In particular, PLSA treats each document as being generated by generating a bag of words, where each word is generated according to a topic. A plate notation for this generative process is shown in Figure 6.1. Given a document d_i, the generative story for $w_1^i, w_2^i, ..., w_{|d_i|}^i$ can be considered as: for every word location index j, (1) generate a topic h_j according to $P(h|d)$; (2) generate the word w_j according to $P(w|h)$. One thing to note here is that d_i in this generative process is not directly regarded as a sequence of words. Instead, it is rather symbolic serving as a more abstract representation of each $d_i \in D$, as a start for generating the word sequence, as shown in the plate notation. The purpose is to learn the distribution of topics for each d_i, namely $P(h|d_i)$. As a result, both the abstract symbolic d_i ($i \in [1, ..., N]$) and the words in V are observed variables, and the topics in T are hidden variables.

According to the above generation process, the complete data likelihood of a word-topic pair $\langle w, h\rangle$ under a certain document d is

$$P(w, h|d) = P(h|d)P(w|h)$$

$P(w, h|d)$ is thus the target joint probability of the PLSA model, with $P(h|d)$ and $P(w|h)$ being its parameter types. We train this model using EM over D. At iteration t, the E-step calculates $P(H|O, \Theta^t)$ for every combination of H and O, namely $P(\hbar|d_i, w, \Theta^t)$ for every document $d_i \in D$ and word $w \in V$, and then defines the Q-function. In particular,

$$\begin{aligned} P(\hbar|d_i, w, \Theta^t) &= \frac{P(\hbar, w|d_i, \Theta^t)}{P(w|d_i, \Theta^t)} \\ &= \frac{P(\hbar|d_i, \Theta^t)P(w|\hbar, \Theta^t)}{\sum_{\hbar'} P(\hbar', w|d_i, \Theta^t)} \\ &= \frac{P(\hbar|d_i, \Theta^t)P(w|\hbar, \Theta^t)}{\sum_{\hbar'} P(\hbar'|d_i, \Theta^t)P(w|\hbar', \Theta^t)} \end{aligned} \tag{6.23}$$

Correspondingly, $Q(\Theta, \Theta^t)$ is

$$\begin{aligned} Q(\Theta, \Theta^t) &= \sum_{i=1}^{N} \sum_{w_j^i \in d_i} \sum_{\hbar} P(\hbar | d_i, w_j^i, \Theta^t) \log P(\hbar, d_i, w_j^i | \Theta) \\ &= \sum_{i=1}^{N} \sum_{w_j^i \in d_i} \sum_{\hbar} P(\hbar | d_i, w_j^i, \Theta^t) [\log P(\hbar | d_i, \Theta) + \log P(w_j^i | \hbar, \Theta)] \\ &= \sum_{i=1}^{N} \sum_{w \in V} C(w, d_i) \sum_{\hbar} P(\hbar | d_i, w, \Theta^t) [\log P(\hbar | d_i, \Theta) + \log P(w | \hbar, \Theta)], \end{aligned} \tag{6.24}$$

where $C(w, d_i)$ denotes the count of w in document d_i.

The M-step optimises $Q(\Theta, \Theta^t)$. Given that $\sum_{\hbar} P(\hbar | d_i, \Theta) = 1$ and $\sum_{w} P(w | \hbar, \Theta) = 1$, we can define a Lagrangian function

$$\Lambda(\Theta, \lambda) = Q(\Theta, \Theta^t) - \sum_i \lambda_{d_i} \Big(\sum_{\hbar} P(\hbar | d_i, \Theta) - 1 \Big) - \sum_{\hbar} \lambda_{\hbar} \Big(\sum_{w} P(w | \hbar, \Theta) - 1 \Big) \tag{6.25}$$

Taking derivatives of $\Lambda(\Theta, \lambda)$ with respect to $P(\hbar | d_i, \Theta)$, we have

$$\frac{\partial \Lambda(\Theta, \lambda)}{\partial P(\hbar | d_i, \Theta)} = \frac{\sum_{w \in V} C(w, d_i) P(\hbar | d_i, w, \Theta^t)}{P(\hbar | d_i, \Theta)} - \lambda_{d_i} \tag{6.26}$$

Considering $\frac{\partial \Lambda(\Theta, \lambda)}{\partial P(\hbar | d_i, \Theta)} = 0$ and $\sum_{\hbar} P(\hbar | d_i, \Theta) - 1 = 0$ gives

$$\begin{aligned} \lambda_{d_i} &= \sum_{\hbar \in T} \sum_{w \in V} C(w, d_i) P(\hbar | d_i, w, \Theta^t) = \sum_{w \in V} C(w, d_i) \\ P(\hbar | d_i, \Theta) &= \frac{\sum_{w \in V} C(w, d_i) P(\hbar | d_i, w, \Theta^t)}{\lambda_{d_i}} = \frac{\sum_{w \in V} C(w, d_i) P(\hbar | d_i, w, \Theta^t)}{\sum_{w \in V} C(w, d_i)}, \end{aligned} \tag{6.27}$$

which serves as $P(\hbar | d_i, \Theta^{t+1})$ for the next EM training iteration.

Intuitively, $\sum_{w \in V} C(w, d_i) P(\hbar | d_i, w, \Theta^t)$ is the expected counts of a latent topic $\hbar$ in document d_i, and $\sum_{w \in V} C(w, d_i)$ is the document length. Using similar methods, we can obtain

$$P(w | \hbar, \Theta) = \frac{\sum_{i=1}^{N} C(w, d_i) P(\hbar | d_i, w, \Theta^t)}{\sum_{i=1}^{N} \sum_{w \in V} C(w, d_i) P(\hbar | d_i, w, \Theta^t)}, \tag{6.28}$$

which serves as $P(w | \hbar, \Theta^{t+1})$ for the next EM training iteration.

PLSA Applications. PLSA is useful for NLP applications by providing a semantic vector representation of texts via topic distributions, which

can be less sparse compared to word count vectors. For information retrieval, for example, PLSA can be used to better evaluate the similarities between queries and documents compared to token matching. When a new query q comes in, the topic of q can be inferred according to a topic-word distribution $P(w|\hbar, \Theta)$ shared between q and d. Once we know the latent topic distribution $P(\hbar|q, \Theta)$ of the query q, we can calculate the distance between the query q and a document d using cosine similarity between the two topic distributions $P(\hbar|q, \Theta)$ and $P(\hbar|d, \Theta)$, and the documents with higher similarity scores can be returned as retrieval targets. It is intuitively better to measure similarity in the latent semantic space instead of the original word-based feature space since the words appearing in the query and documents might be different, but in the latent space, their topics might be the same. For example, "*bank*" and "*ATM*" are different, but their topics might be all about "money". One the other hand, the topic of a same word in different documents might be different. For example, the topics of "*bank*" in "*the bank of Ganges River*" and "*the commercial bank manager*" are different.

6.2.4 The Relative Advantages of Generative Models

The models we discussed in this section are generative models, which make generative stories in parameterisation, rather than directly scoring output candidates using rich overlapping features. They model a joint probability distribution $P(X, Y)$ of the input X and output Y rather than $P(Y|X)$. Compared to discriminative models such as SVMs and log-linear models, the relative advantages of generative models is their interpretability. They can be used to explain the production process of X with the help of hidden variables. As exemplified by Figure 6.1, parameters in a generative model explain why a set of data is observed, by explaining the origin of them from a set of inter-related factors, such as topics. In addition, generative models can be used to synthesise data, by sampling random variables according to model probabilities, as shown in the example of using language models to generate sentences in Chapter 2.

6.3 Theory behind EM

In this section, we show why EM works by demonstrating that it optimises a maximum likelihood objective. Formally, given a dataset $O = \{o_i\}|_{i=1}^{N}$, the model that we train calculates $P(O, H|\Theta)$. Since we do not know the full data likelihood, the model can be trained indirectly by maximising only

the log-likelihood of the observed data O given Θ:

$$\begin{aligned} L(\Theta) &= \log P(O|\Theta) \\ &= \log \sum_H P(O, H|\Theta) \end{aligned} \tag{6.29}$$

In the equation above, $\sum_H$ enumerates all possible values of H. Thus we effectively learn Θ in the $P(O, H|\Theta)$ parameter setting, by marginalising out H to obtain $P(O|\Theta)$.

Unfortunately, optimising log of sums is intractable. Therefore, it is difficult to directly optimise Eq 6.29. EM optimises a lower bound of Eq 6.29 using *Jensen inequality*. In particular, regarding the concave function as $\log(\cdot)$, Jensen inequality states that if X is a random variable under distribution $P(X)$, then $\log\Big(E_{X\sim P(X)} f(X)\Big) \geq E_{X\sim P(X)}\Big(\log(f(X))\Big)$ for a real-valued function f. Let $P_C(H)$ be a certain probability distribution of H, where $\sum_H P_C(H) = 1$. Using Jensen inequality, we have

$$\begin{aligned} L(\Theta) &= \log \sum_H P(O, H|\Theta) \\ &= \log \sum_H P_C(H) \frac{P(O, H|\Theta)}{P_C(H)} \\ &= \log E_{H\sim P_C(H)} \frac{P(O, H|\Theta)}{P_C(H)} \\ &\geq E_{H\sim P_C(H)} \log \frac{P(O, H|\Theta)}{P_C(H)} \text{ (Jensen inequality)} \\ &= \sum_H P_C(H) \log \frac{P(O, H|\Theta)}{P_C(H)} \end{aligned} \tag{6.30}$$

Denote $F(\Theta, P_C) = \sum_H P_C(H) \log \frac{P(O,H|\Theta)}{P_C(H)}$. We have $L(\Theta) \geq F(\Theta, P_C)$ in Eq 6.30, which means that $F(\Theta, P_C)$ is a lower bound of $L(\Theta)$. Below we derive two different methods for optimising $F(\Theta, P_C)$, both of which lead to Algorithm 6.2.

6.3.1 EM and KL-Divergence

$F(\Theta, P_C)$ can be rewritten as

$$\begin{aligned} F(\Theta, P_C) &= \sum_H P_C(H) \log \frac{P(O, H|\Theta)}{P_C(H)} \\ &= \sum_H P_C(H) \log \frac{P(O|\Theta)P(H|O, \Theta)}{P_C(H)} \\ &= \sum_H P_C(H) \log P(O|\Theta) + \sum_H P_C(H) \log \frac{P(H|O, \Theta)}{P_C(H)} \\ &= \log P(O|\Theta) - \Big(- \sum_H P_C(H) \log \frac{P(H|O, \Theta)}{P_C(H)} \Big) \\ &= L(\Theta) - KL\Big(P_C(H), P(H|O, \Theta)\Big) \end{aligned} \tag{6.31}$$

In Chapter 5 we have learned that KL-divergence is always non-negative. In addition, $KL(P, Q)$ is zero if and only if $P = Q$. According to Eq 6.31, the difference between $F(\Theta, P_C)$ and $L(\Theta)$ is $KL\Big(P_C(H), P(H|O, \Theta)\Big)$. In order to make the bound as tight as possible, $KL\Big(P_C(H), P(H|O, \Theta)\Big)$ should be as small as possible. Since $KL\Big(P_C(H), P(H|O, \Theta)\Big) \geq 0$, letting $KL\Big(P_C(H), P(H|O, \Theta)\Big) = 0$ gives the best estimate of $P_C(H)$. Therefore, $P_C(H) = P(H|O, \Theta)$, which is the reason behind the choice in Section 6.1.2.

If the model parameters Θ are already known, $P_C(H) = P(H|O, \Theta)$ is the distribution of the hidden variable H given the observed data O according to the model. $P_C(H)$ can thus be regarded as the soft count of each hidden variable value H. In this scenario, finding the distribution $P_C(H)$ corresponds to the E-step in Algorithm 6.2.

We can now take the M-step to optimise $F(\Theta, P_C)$ by using the $P(H|O, \Theta)$ values we have obtained. To distinguish the fixed parameters (i.e., those in the value in $P(H|O, \Theta)$) and variables being adjusted, we again explicitly include the iteration number as superscripts, where $P_C^{t+1}(H) = P(H|O, \Theta^t)$ is fixed when finding Θ^{t+1}.

Substituting $P(H|O, \Theta^t)$ back to Eq 6.31, we have

$$F(\Theta, P_C^{t+1}) = \sum_H P(H|O, \Theta^t) \log \frac{P(O, H|\Theta)}{P(H|O, \Theta^t)} \tag{6.32}$$

Θ^{t+1} is given by

$$\begin{aligned}\Theta^{t+1} &= \arg\max_{\Theta} F(\Theta, P_C^{t+1}) \\ &= \arg\max_{\Theta} \sum_H P(H|O,\Theta^t) \log \frac{P(O,H|\Theta)}{P(H|O,\Theta^t)} \\ &= \arg\max_{\Theta} \sum_H P(H|O,\Theta^t) \log P(O,H|\Theta) \\ &= \arg\max_{\Theta} Q(\Theta,\Theta^t),\end{aligned} \tag{6.33}$$

where $Q(\Theta, \Theta^t)$ is the same as Algorithm 6.2 and Eq 6.9.

6.3.2 EM Derivation Using Numerical Optimisation

As mentioned earlier, $F(\Theta, P_C)$ is a lower bound of $L(\Theta)$ to optimise. $F(\Theta, P_C)$ contains two variables, and as a result can be optimised via **coordinate ascent**. Different from gradient ascent, at each iteration, coordinate ascent chooses one coordinate (or a variable) to optimise, while keeping the others fixed. The gradient direction of coordinate ascent is along the current coordinate. For our problem, first, assuming that Θ^t is already known, we find $P_C^{t+1} = \arg\max_{P_C} F(\Theta^t, P_C)$. Then using P_C^{t+1}, we find $\Theta^{t+1} = \arg\max_{\Theta} F(\Theta, P_C^{t+1})$. The first step is the E-step and the second step is the M-step.

Expectation step. The E-step finds an optimum distribution $P_C(H)$ that maximises $F(\Theta^t, P_C)$:

$$\begin{aligned}P_C^{t+1} &= \arg\max_{P_C} F(\Theta^t, P_C) \\ &= \arg\max_{P_C} \sum_H P_C(H) \log \frac{P(O,H|\Theta^t)}{P_C(H)} \\ &= \arg\max_{P_C} \sum_H \Big(P_C(H) \log P(O,H|\Theta^t) - P_C(H) \log P_C(H) \Big)\end{aligned} \tag{6.34}$$

This is a constrained optimisation problem. We can use Lagrange multipliers to incorporate the constraint $\sum_H P_C(H) = 1$. The corresponding Lagrange function is

$$F_\lambda(\Theta^t, P_C) = \sum_H \Big(P_C(H) \log P(O,H|\Theta^t) - P_C(H) \log P_C(H) \Big) - \lambda(\sum_H P_C(H) - 1),$$

where $\lambda \in \mathbb{R}$. Taking partial derivatives of $F_\lambda(\Theta^t, P_C)$ with respect to $P_C(H)$, we have for all H

$$\frac{\partial F_\lambda(\Theta^t, P_C)}{\partial P_C(H)} = \log P(O,H|\Theta^t) - \log P_C(H) - 1 - \lambda$$

Let $\frac{\partial F_\lambda(\Theta^t, C)}{\partial P_C(H)} = 0$, we obtain

$$P_C(H) = \frac{P(O, H|\Theta^t)}{e^{1+\lambda}} \text{ (for all possible } H)$$

Further, given the fact that $\sum_H P_C(H) = 1$. Therefore, we have

$$\begin{aligned} e^{1+\lambda} &= \sum_H P(O, H|\Theta^t) = P(O|\Theta^t) \\ P_C^{t+1}(H) &= \frac{P(O, H|\Theta^t)}{P(O|\Theta^t)} \\ &= P(H|O, \Theta^t) \end{aligned} \tag{6.35}$$

Maximisation step. The M-step finds the optimal Θ^{t+1} for $F(\Theta, P_C^{t+1})$ using P_C^{t+1}. This step is the same as Section 6.3.1.

After setting the distribution $P_C^{t+1}(H) = P(H|O, \Theta^t)$, according to Eq 6.31, we know that

$$L(\Theta^t) = F(\Theta^t, P_C^{t+1}) \tag{6.36}$$

Therefore, any Θ that can increase $F(\Theta, P_C^{t+1})$ can improve $L(\Theta)$.

Convergence. After one iteration of E-step and M-step, we can show that $L(\Theta^{t+1}) - L(\Theta^t) \geq 0$. In particular, we have

$$\begin{aligned} L(\Theta^t) &= F(\Theta^t, P_C^{t+1}) \text{ (Eq 6.36)} \\ &\leq F(\Theta^{t+1}, P_C^{t+1}) \ (\Theta^{t+1} = \arg\max_\Theta F(\Theta, P_C^{t+1})) \\ &\leq F(\Theta^{t+1}, P_C^{t+2}) \ (P_C^{t+2} = \arg\max_{P_C} F(\Theta^{t+1}, P_C)) \\ &= L(\Theta^{t+1}) \text{ (Eq 6.36)} \end{aligned}$$

Therefore, $L(\Theta^t)$ is a monotonically increasing function with respect to $|t|$, where $L(\Theta^0) \leq L(\Theta^1) \leq L(\Theta^2) \cdots \leq L(\Theta^n)$. It can be proved that EM is guaranteed to converge to local optima. The initial values Θ^0 significantly affect the resulting models. In practice, it is useful to try multiple random starting points, selecting a best model on a set of development data.

6.4 Summary

In this chapter we have introduced:

- The concept of hidden variables;
- The expectation maximisation (EM) algorithm;
- The correlation between EM and MLE for training probabilistic models;

Document	Document
Apple released iPod .	Tom bought one iPod .
Apple released iPhone .	Tom bought one iPhone .
Apple released iPad .	Tom bought one iPad .

Table 6.2: A collection of documents for clustering.

- EM for unsupervised text classification;
- IBM model 1 for statistical machine translation;
- Probabilistic latent semantic allocation.

Chapter Notes

Hartley (1958) first proposed the expectation maximisation (EM) algorithm. Dempster et al. (1977) formalised EM and provided a proof of convergence. Lauritzen (1995) further extended the EM algorithm to graphical association models. A recent book devoted entirely to EM and applications is (McLachlan and Krishnan, 1997), whereas (Schafer and Joseph, 1997) is another popular and very useful reference.

Neal and Hinton (1998) and Minka (1998) proposed an insightful explanations of EM in terms of lowerbound maximisation. Dempster et al. (1977) used the expectation maximisation (EM) algorithm to learn a Bayesian network (Chapter 12) with hidden variables. Brown et al. (1993) proposed IBM Models for machine translation, in which word alignments obtained via EM algorithm. Hofmann (1999) proposed probabilistic latent semantic analysis that contains hidden variable.

Exercises

6.1 In Eq 6.9, which variables are fixed and which can be adjusted?

6.2 Corpus-level hidden variable assignment.

1) Section 6.1.2 mentions a hidden variable assignment $\mathcal{H}$. Given a specific example what it is in the case of k-means clustering.

2) Eq 6.8 gives a corpus-level training objective for the M-step of soft EM. Change the equation to the instance-level supposing that $O = \{\vec{v}_i\}|_{i=1}^N$ and $H = \{h_i\}|_{i=1}^N$.

6.3 Given a collection of six documents as shown in Table 6.2, use the unsupervised Naïve Bayes model to cluster the documents into: (1) 2 classes

ID	Source	Target
1	เขา(he) อาศัย(live) อยู่ใน(in) กรุงเทพ(Bangkok)	He is living in Bangkok
2	เขา(he) ชอบ(like) กรุงเทพ(Bangkok)	He likes Bangkok
3	เขา(he) ชอบ(like) อาศัย(live) อยู่ใน(in) กรุงเทพ(Bangkok)	He likes living in Bangkok

Table 6.3: A parallel corpus.

(2) 3 classes. Initialise the model parameter $P(\hbar|\Theta)$ with $\frac{1}{K}$ for every class $\hbar$, where K is the total number of classes. For every class $\hbar$, initialise the model parameter $P(w|\hbar,\Theta)$ with $\frac{1}{|V|}$, where $|V|$ is the vocabulary size. Estimate the model parameters $P(\hbar|\Theta)$ and $P(w|\hbar,\Theta)$ according to Eq 6.14 and Eq 6.15, respectively. Compare the the 2-class clustering results with the 3-class ones. Can you design a real EM variant of k-means?

6.4 Consider a semi-supervised settings for the Naïve Bayes model, where a set of labeled documents $D = \{(d_i, c_i)\}|_{i=1}^{N}$ and a set of unlabeled documents $U = \{d_i\}|_{N+1}^{N+M}$ are available. The training objective is to maximise

$$L(\Theta) = \sum_{i=1}^{N} \log P(d_i, c_i|\Theta) + \sum_{j=N+1}^{N+M} \log P(d_j|\Theta) \tag{6.37}$$

1) Describe how to train the model parameters Θ using an algorithm similar to Algorithm 6.2.

2) What is the role of the unlabeled data in this training objective? If we add a hyper-parameter λ to indicate that how much attention we should pay to the unlabeled data, the training objective becomes,

$$L(\Theta) = \sum_{i=1}^{N} \log P(d_i, c_i|\Theta) + \lambda \sum_{j=N+1}^{N+M} \log P(d_j|\Theta)$$

Compare the second term $\lambda \sum_{j=N+1}^{N+M} \log P(d_j|\Theta)$ with the L2-regulariser introduced in Chapter 3.

6.5 Given a parallel corpus as shown in Table 6.3, (1) execute IBM model 1 for one iteration and show the model parameters; (2) suppose that we already have a location dictionary, indicating that "กรุงเท" in Thai and "*Bangkok*" in English should always be connected, which means $P(\text{กรุงเท}|\textit{Bangkok}) = 1$, run IBM model 1 from scratch again and show the model parameters.

6.6 Prove the last step of Eq 6.20.

ID	Document	ID	Document
1	World Cup, Russia, host	2	World Cup, boost, Russia, economy
3	Russia, bid, World Cup	4	Russia, economy, growing, oil
5	Russia, economy, recover, continue	6	Russia, oil, dependence

Table 6.4: A collection of documents for latent topic analysis.

(Hint: Calculate $\sum_{a_{|X|}=0}^{|Y|} \prod_{i=1}^{|X|} P(x_i|y_{a_i}) = \left(\prod_{i=1}^{|X|-1} P(x_i|y_{a_i=0}) \right) \sum_{j=0}^{|Y|} P(x_{|X|}|y_j)$ first, and then $\sum_{a_{|X|-1}=0}^{|Y|} \sum_{a_{|X|}=0}^{|Y|} \prod_{i=1}^{|X|} P(x_i|y_{a_i}) = \prod_{i=1}^{|X|-2} P(x_i|y_{a_i=0}) \sum_{j=0}^{|Y|} P(x_{|X|-1}|y_j) \sum_{j=0}^{|Y|} P(x_{|X|}|y_j)$ before deriving $\sum_{a_1=0}^{|Y|} \sum_{a_2=0}^{|Y|} \cdots \sum_{a_{|X|}=0}^{|Y|} \prod_{i=1}^{|X|} P(x_i|y_{a_i}) \prod_{i=1}^{|X|} \sum_{j=0}^{|Y|} P(x_i|y_j)$.)

6.7 Prove the last step of Eq 6.22.

6.8 Given a document collection as shown in Table 6.4, suppose that there are two latent topic, one is about "World Cup" and the other is about "Russia's economy", (1) use PLSA to estimate the document-topic and topic-word probabilities; (2) compare the similarities of document pairs $\langle d_1, d_3 \rangle$, $\langle d_4, d_5 \rangle$ and $\langle d_2, d_5 \rangle$ using the document-topic distribution.

6.9 Self-training in Chapter 4 and hard EM are to some extent similar. They both predict labels for unlabeled instances and make use of automatically generated labels for iterative training. Show similarities and differences between self-training and hard EM.