3 Machine Learning Models

This chapter provides detailed descriptions and derivations of various machine learning models for speaker verification. Instead of discussing these models in chronological order, we will start from the fundamental models, such as Gaussian mixture models and support vector machines and progressively move to the more complicated ones that are built on these fundamental models. We will leave the recent development in DNN-based models to the next two chapters.

3.1 Gaussian Mixture Models

A Gaussian mixture model (GMM) is a weighted sum of Gaussian distributions. In the context of speech and speaker recognition, a GMM can be used to represent the distribution of acoustic vectors $\mathbf{o}$'s:[1]

$$p(\mathbf{o}|\Lambda) = \sum_{c=1}^{C} \pi_c \mathcal{N}(\mathbf{o}|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c), \tag{3.1}$$

where $\Lambda = \{\pi_c, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c\}_{c=1}^{C}$ contains the parameters of the GMM. In Eq. 3.1, π_c, $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$ are the weight, mean vector, and covariance matrix of the cth mixture whose density has the form

$$\mathcal{N}(\mathbf{o}|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) = \frac{1}{(2\pi)^{\frac{D}{2}} |\boldsymbol{\Sigma}_c|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\mathbf{o} - \boldsymbol{\mu}_c)^{\mathsf{T}} \boldsymbol{\Sigma}_c^{-1} (\mathbf{o} - \boldsymbol{\mu}_c)\right\}.$$

The mixture weight π_c's can be considered as the priors of the mixtures and they should sum to 1.0, i.e., $\sum_{c=1}^{C} \pi_c = 1$. Figure 3.1 shows an example of a one-dimensional GMM with three mixtures.

To estimate Λ, we express the log-likelihood of a set of training data in terms of the parameters and find the parameters in Λ that lead to the maximum-likelihood of the training data. Specifically, given a set of T independent and identically distributed (iid) acoustic vectors $\mathcal{O} = \{\mathbf{o}_t; t = 1, \ldots, T\}$, the log-likelihood function (function of Λ) is given by

[1] In this book, we use lower case boldface letters to denote random vectors. Depending on the context, lower-case boldface letters can also represent the realization (i.e., observations) of the corresponding random vectors. Note that this is different from the usual probabilistic convention that random variables are represented by capital letters (e.g., X) and their realization (x), as in $P(X < x)$.

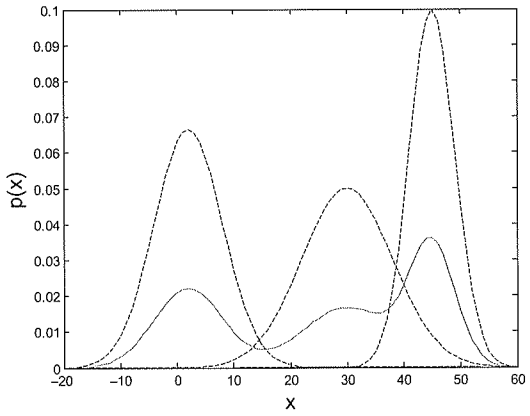

Figure 3.1 A one-D GMM with three mixture components. [Adapted from *Bayesian Speech and Language Processing (Figure 3.2), by S. Watanabe and J.T. Chien, 2015, Cambridge University Press.*]

$$\log p(\mathcal{O}|\Lambda) = \log\left\{\prod_{t=1}^{T}\sum_{c=1}^{C}\pi_c\mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_c,\boldsymbol{\Sigma}_c)\right\} = \sum_{t=1}^{T}\log\left\{\sum_{c=1}^{C}\pi_c\mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_c,\boldsymbol{\Sigma}_c)\right\}.$$

To find Λ that maximizes $\log p(\mathcal{O}|\Lambda)$, we may set $\frac{\partial \log p(\mathcal{O})}{\partial \Lambda} = 0$ and solve for Λ. But this method will not give a closed-form solution for Λ. The trouble is that the summation appears inside the logarithm. The problem, however, can be readily solved by using the EM algorithm to be explained next.

3.1.1 The EM Algorithm

The EM algorithm (see Section 2.2.2) is a popular iterative method for estimating the parameters of statistical models. It is also an elegant method for finding maximum-likelihood solutions for models with latent variables, such as GMMs. This is achieved by maximizing a log-likelihood function of the model parameters through two iterative steps: **E**xpectation and **M**aximization.

Denote the acoustic vectors from a large population as $\mathcal{O} = \{\mathbf{o}_t; t = 1, \ldots, T\}$. In GMM, for each data point $\mathbf{o}_t$, we do not know which Gaussian generates it. Therefore, the latent information is the Gaussian identity for each $\mathbf{o}_t$. Define

$$\mathcal{L} = \{\ell_{tc};\ t = 1, \ldots, T \text{ and } c = 1, \ldots, C\}$$

as the set of latent variables, where $\ell_{tc} = 1$ if $\mathbf{o}_t$ is generated by the cth Gaussian; otherwise $\ell_{tc} = 0$. Figure 3.2 shows a graphical representation of a GMM. This graphical model suggests that $\boldsymbol{o}_t$ depends on $\boldsymbol{\ell}_t = [\ell_{t1} \ \ldots \ \ell_{tC}]^{\mathsf{T}}$ so that the marginal likelihood $p(\mathbf{o}_t)$ can be obtained by marginalizing out the latent variable $\boldsymbol{\ell}_t$:

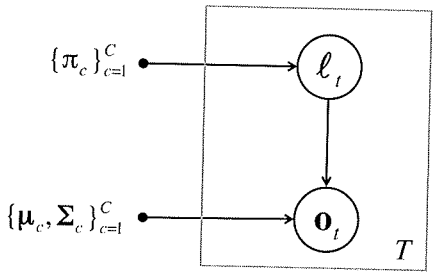

Figure 3.2 Graphical representation of a GMM. $\boldsymbol{\ell}_t = [\ell_{t1} \ \cdots \ \ell_{tC}]^\mathsf{T}$ and $\mathbf{o}_t$, with $t = 1, \ldots, T$, are the latent variables and observed vectors, respectively.

$$\begin{aligned} p(\mathbf{o}_t) &= \sum_{\boldsymbol{\ell}_t} p(\boldsymbol{\ell}_t) p(\mathbf{o}_t|\boldsymbol{\ell}_t) \\ &= \sum_{c=1}^{C} P(\ell_{tc} = 1) p(\mathbf{o}_t|\ell_{tc} = 1) \\ &= \sum_{c=1}^{C} \pi_c \mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c), \end{aligned} \tag{3.2}$$

which has the same form as Eq. 3.1.

In the EM literatures, $\{\mathcal{O}, \mathcal{L}\}$ is called the *complete* data set, and $\mathcal{O}$ is the *incomplete* data set. In EM, we maximize $\log p(\mathcal{O}, \mathcal{L}|\Lambda)$ with respect to Λ instead of maximizing $\log p(\mathcal{O}|\Lambda)$. The important point is that maximizing the former is much more straightforward and will lead to closed-form solutions for each EM iteration, as will be shown below.

However, we actually do not know $\mathcal{L}$. So, we could not compute $\log p(\mathcal{O}, \mathcal{L}|\Lambda)$. Fortunately, we know its posterior distribution, i.e., $P(\mathcal{L}|\mathcal{O}, \Lambda)$, through the Bayes theorem. Specifically, for each $\mathbf{o}_t$, we compute the posterior probability:[2]

$$\begin{aligned} \gamma(\ell_{tc}) &\equiv P(\ell_{tc} = 1|\mathbf{o}_t, \Lambda) \\ &= \frac{P(\ell_{tc} = 1|\Lambda) p(\mathbf{o}_t|\ell_{tc} = 1, \Lambda)}{p(\mathbf{o}_t|\Lambda)} \\ &= \frac{\pi_c \mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)}{\sum_{j=1}^{C} \pi_j \mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \end{aligned} \tag{3.3}$$

In the speech and speaker recognition literatures, computing the posterior probabilities of mixture components is called *alignment*. Its aim is to determine how close a vector $\mathbf{o}_t$ is to the individual Gaussians, accounting for both the priors, means and covariances of the mixture components. Figure 3.3 illustrates the alignment process.

With the posteriors $\gamma(\ell_{tc})$, given the current estimate of the model parameters Λ^{old}, we can find its new estimate Λ by computing the expected value of $\log p(\mathcal{O}, \mathcal{L}|\Lambda)$ under

[2] We denote probabilities and probability mass functions of discrete random variables using capital letter P, and we denote the likelihoods and probability density functions of continuous random variables using lower case letter p.

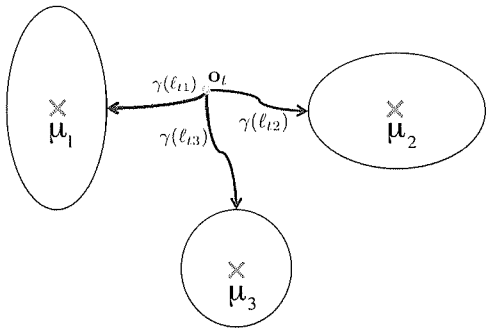

Figure 3.3 Aligning an acoustic vector $\mathbf{o}_t$ to a GMM with three mixture components.

the posterior distribution of $\mathcal{L}$:

$$\begin{aligned} Q(\Lambda|\Lambda^{\text{old}}) &= \mathbb{E}_{\mathcal{L}}\{\log p(\mathcal{O},\mathcal{L}|\Lambda)|\mathcal{O},\Lambda^{\text{old}}\} \\ &= \sum_{t=1}^{T}\sum_{c=1}^{C} P(\ell_{tc}=1|\mathbf{o}_t,\Lambda^{\text{old}})\log p(\mathbf{o}_t,\ell_{tc}=1|\Lambda) \\ &= \sum_{t=1}^{T}\sum_{c=1}^{C} \gamma(\ell_{tc})\log p(\mathbf{o}_t,\ell_{tc}=1|\Lambda) \\ &= \sum_{t=1}^{T}\sum_{c=1}^{C} \gamma(\ell_{tc})\log p(\mathbf{o}_t|\ell_{tc}=1,\Lambda)P(\ell_{tc}=1|\Lambda) \\ &= \sum_{t=1}^{T}\sum_{c=1}^{C} \gamma(\ell_{tc})\log\left[\mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_c,\boldsymbol{\Sigma}_c)\pi_c\right], \end{aligned} \tag{3.4}$$

where $Q(\Lambda|\Lambda^{\text{old}})$ is called the auxiliary function or simply Q-function (see Eq. 2.6). The E-step consists in computing $\gamma(\ell_{tc})$ for all training samples so that $Q(\Lambda|\Lambda^{\text{old}})$ can be expressed as a function of $\boldsymbol{\mu}_c$, $\boldsymbol{\Sigma}_c$, and π_c for $c = 1,\ldots,C$.

Then, in the M-step, we maximize $Q(\Lambda|\Lambda^{\text{old}})$ with respect to Λ by setting $\frac{\partial Q(\Lambda|\Lambda^{\text{old}})}{\partial \Lambda} = 0$ to obtain (see [32, Ch. 3]):

$$\boldsymbol{\mu}_c = \frac{\sum_{t=1}^{T}\gamma(\ell_{tc})\mathbf{o}_t}{\sum_{t=1}^{T}\gamma(\ell_{tc})} \tag{3.5a}$$

$$\boldsymbol{\Sigma}_c = \frac{\sum_{t=1}^{T}\gamma(\ell_{tc})(\mathbf{o}_t-\boldsymbol{\mu}_c)(\mathbf{o}_t-\boldsymbol{\mu}_c)^{\mathsf{T}}}{\sum_{t=1}^{T}\gamma(\ell_{tc})} \tag{3.5b}$$

$$\pi_c = \frac{1}{T}\sum_{t=1}^{T}\gamma(\ell_{tc}), \tag{3.5c}$$

where $c = 1, \ldots, C$. Eq. 3.5a–Eq. 3.5c constitute the M-step of the EM algorithm. In practical implementation of the M-step, we compute the sufficient statistics:

$$n_c = \sum_{t=1}^{T} \gamma(\ell_{tc}) \tag{3.6a}$$

$$\mathbf{f}_c = \sum_{t=1}^{T} \gamma(\ell_{tc}) \mathbf{o}_t \tag{3.6b}$$

$$\mathbf{S}_c = \sum_{t=1}^{T} \gamma(\ell_{tc}) \mathbf{o}_t \mathbf{o}_t^{\mathsf{T}}, \tag{3.6c}$$

where $c = 1, \ldots, C$. Then, Eq. 3.5a–Eq. 3.5c become

$$\mu_c = \frac{1}{n_c} \mathbf{f}_c \tag{3.7a}$$

$$\mathbf{\Sigma}_c = \frac{1}{n_c} \mathbf{S}_c - \mu_c \mu_c^{\mathsf{T}} \tag{3.7b}$$

$$\pi_c = \frac{1}{T} n_c. \tag{3.7c}$$

In summary, the EM algorithm iteratively performs the E- and M-steps until $Q(\Lambda|\Lambda^{\text{old}})$ no longer increases.

- **Initialization**: Randomly select C samples from $\mathcal{O}$ and assign them to $\{\mu_c\}_{c=1}^{C}$; Set $\pi_c = \frac{1}{C}$ and $\mathbf{\Sigma}_c = \mathbf{I}$, where $c = 1, \ldots, C$.
- **E-Step**: Find the distribution of the latent (unobserved) variables, given the observed data and the current estimate of the parameters;
- **M-Step**: Re-estimate the parameters to maximize the likelihood of the observed data, under the assumption that the distribution found in the E-step is correct.

The iterative process guarantees to increases the true likelihood or leaves it unchanged (if a local maximum has already been reached).

3.1.2 Universal Background Models

If we use the speech of a large number of speakers to train a GMM using Eq. 3.3 and Eqs. 3.5a–3.5c, we obtain a universal background model (UBM), Λ^{ubm}, with density function

$$p(\mathbf{o}|\Lambda^{\text{ubm}}) = \sum_{c=1}^{C} \pi_c^{\text{ubm}} \mathcal{N}(\mathbf{o}|\mu_c^{\text{ubm}}, \mathbf{\Sigma}_c^{\text{ubm}}). \tag{3.8}$$

Specifically, the UBM is obtained by iteratively carried out the E- and M-steps as follows.

- E-step: Compute the conditional distribution of mixture components:

$$\gamma(\ell_{tc}) \equiv \Pr(\text{Mixture} = c|\mathbf{o}_t) = \frac{\pi_c^{\text{ubm}}\mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_c^{\text{ubm}}, \boldsymbol{\Sigma}_c^{\text{ubm}})}{\sum_{c=1}^{C}\pi_c^{\text{ubm}}\mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_c^{\text{ubm}}, \boldsymbol{\Sigma}_c^{\text{ubm}})} \tag{3.9}$$

  where $c = 1, \ldots, C$.
- M-step: Update the model parameters:
  - Mixture weights: $\pi_c^{\text{ubm}} = \frac{1}{T}\sum_{t=1}^{T}\gamma(\ell_{tc})$
  - Mean vectors: $\boldsymbol{\mu}_c^{\text{ubm}} = \frac{\sum_{t=1}^{T}\gamma(\ell_{tc})\mathbf{o}_t}{\sum_{t=1}^{T}\gamma(\ell_{tc})}$
  - Covariance matrices: $\boldsymbol{\Sigma}_c^{\text{ubm}} = \frac{\sum_{t=1}^{T}\gamma(\ell_{tc})\mathbf{o}_t\mathbf{o}_t^T}{\sum_{t=1}^{T}\gamma(\ell_{tc})} - \boldsymbol{\mu}_c^{\text{ubm}}(\boldsymbol{\mu}_c^{\text{ubm}})^\top$

  where $c = 1, \ldots, C$.

In a GMM–UBM system, each of the genuine speakers (also called target speakers) has his/her own GMM, and the UBM serves as a reference for comparing likelihood ratios. More precisely, if the likelihood of a test utterance with respect to a genuine-speaker's GMM is larger than its likelihood with respect to the UBM, there is a high chance that the test utterance is spoken by the genuine speaker.

3.1.3 MAP Adaptation

If each target speaker has a large number of utterances for training his/her GMM, the EM algorithm in Section 3.1 can be directly applied. However, in practice, the amount of speech for each speaker is usually small. As a result, directly applying the EM algorithm will easily cause overfitting. A better solution is to apply the maximum *a posteriori* (MAP) adaptation in which target-speaker models are adapted from the UBM.

The MAP algorithm finds the parameters of target-speaker's GMM given UBM parameters $\Lambda^{\text{ubm}} = \{\pi_c^{\text{ubm}}, \boldsymbol{\mu}_c^{\text{ubm}}, \boldsymbol{\Sigma}_c^{\text{ubm}}\}_{c=1}^{C}$ using the EM algorithm. The objective is to estimate the mode of the posterior of the model parameters:

$$\begin{aligned}\Lambda^{\text{map}} &= \underset{\Lambda}{\text{argmax}}\, p(\Lambda|\mathcal{O}) \\ &= \underset{\Lambda}{\text{argmax}}\, p(\mathcal{O}|\Lambda)p(\Lambda) \\ &= \underset{\Lambda}{\text{argmax}} \prod_{t=1}^{T} p(\mathbf{o}_t|\Lambda)p(\Lambda).\end{aligned} \tag{3.10}$$

Direct optimization of Eq. 3.10 is difficult. However, we may iteratively find the optimal solution via the EM algorithm. Similar to the EM algorithm for GMM in Section 3.1, instead of maximizing Eq. 3.10, we maximize the auxiliary function:

$$
\begin{aligned}
Q(\Lambda|\Lambda^{\text{old}}) &= \mathbb{E}_{\mathcal{L}}\left\{\log\left[p(\mathcal{O},\mathcal{L}|\Lambda)p(\Lambda)\right]|\mathcal{O},\Lambda^{\text{old}}\right\} \\
&= \mathbb{E}_{\mathcal{L}}\left\{\log p(\mathcal{O},\mathcal{L}|\Lambda)|\mathcal{O},\Lambda^{\text{old}}\right\} + \mathbb{E}_{\mathcal{L}}\left\{\log p(\Lambda)|\mathcal{O},\Lambda^{\text{old}}\right\} \\
&= \sum_{t=1}^{T}\sum_{c=1}^{C} P(\ell_{tc}=1|\mathbf{o}_t,\Lambda^{\text{old}})\log p(\mathbf{o}_t,\ell_{tc}=1|\Lambda) + \log p(\Lambda) \\
&= \sum_{t=1}^{T}\sum_{c=1}^{C} \gamma(\ell_{tc})\log p(\mathbf{o}_t,\ell_{tc}=1|\Lambda) + \log p(\Lambda) \\
&= \sum_{t=1}^{T}\sum_{c=1}^{C} \gamma(\ell_{tc})\left[\log p(\mathbf{o}_t|\ell_{tc}=1,\Lambda) + \log P(\ell_{tc}=1|\Lambda)\right] + \log p(\Lambda) \\
&= \sum_{t=1}^{T}\sum_{c=1}^{C} \gamma(\ell_{tc})\left[\log \mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_c,\boldsymbol{\Sigma}_c) + \log \pi_c\right] + \log p(\Lambda), \qquad (3.11)
\end{aligned}
$$

If only the mean vectors are adapted, we have $\pi_c = \pi_c^{\text{ubm}}$ and $\boldsymbol{\Sigma}_c = \boldsymbol{\Sigma}_c^{\text{ubm}}$. Also, we only need to assume a prior over $\boldsymbol{\mu}_c$:

$$
p(\boldsymbol{\mu}_c) = \mathcal{N}(\boldsymbol{\mu}_c|\boldsymbol{\mu}_c^{\text{ubm}}, r^{-1}\boldsymbol{\Sigma}_c^{\text{ubm}}), \qquad (3.12)
$$

where r is called the relevant factor [7]. Figure 3.4 shows the graphical model of the Bayesian GMM when only the mean vectors $\{\boldsymbol{\mu}_c\}_{c=1}^{C}$ of the GMM are assumed random with prior distribution given by Eq. 3.12.

Dropping terms independent of $\boldsymbol{\mu}_c$, Eq. 3.11 can be written as:

$$
Q(\boldsymbol{\mu}|\boldsymbol{\mu}^{\text{ubm}}) = \sum_{t=1}^{T}\sum_{c=1}^{C} \gamma(\ell_{tc})\log \mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_c,\boldsymbol{\Sigma}_c^{\text{ubm}}) + \log \mathcal{N}(\boldsymbol{\mu}_c|\boldsymbol{\mu}_c^{\text{ubm}}, r^{-1}\boldsymbol{\Sigma}_c^{\text{ubm}}). \quad (3.13)
$$

where $\boldsymbol{\mu} = [\boldsymbol{\mu}_1^{\mathsf{T}} \ \ldots \ \boldsymbol{\mu}_C^{\mathsf{T}}]^{\mathsf{T}}$.

The E-step is similar to the E-step in training GMMs. Specifically, given T_s acoustic vectors $\mathcal{O}^{(s)} = \{\mathbf{o}_1,\ldots,\mathbf{o}_{T_s}\}$ from speaker s, we compute the sufficient statistics:

$$
n_c = \sum_{t=1}^{T_s} \gamma(\ell_{tc}) \quad \text{and} \quad E_c(\mathcal{O}^{(s)}) = \frac{1}{n_c}\sum_{t=1}^{T_s} \gamma(\ell_{tc})\mathbf{o}_t, \qquad (3.14)
$$

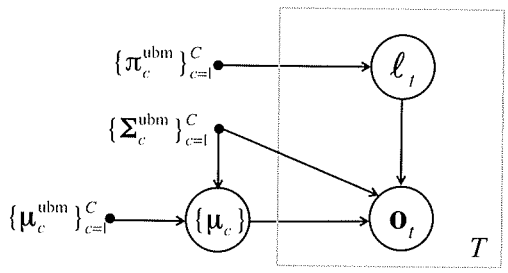

Figure 3.4 Graphical model of Bayesian GMMs with prior over the GMM's mean vectors given by Eq. 3.12.

where $\gamma(\ell_{tc})$ is computed as in Eq. 3.3. In the M-step, we differentiate $Q(\mu|\mu^{\text{old}})$ in Eq. 3.13 with respect to μ_c, which gives

$$\frac{\partial Q(\mu|\mu^{\text{ubm}})}{\partial \mu_c} = \sum_{t=1}^{T_s} \gamma(\ell_{tc})(\Sigma_c^{\text{ubm}})^{-1}(\mathbf{o}_t - \mu_c) - r(\Sigma_c^{\text{ubm}})^{-1}(\mu_c - \mu_c^{\text{ubm}}). \tag{3.15}$$

By setting Eq. 3.15 to $\mathbf{0}$, we obtain the adapted mean

$$\begin{aligned}\mu_c &= \frac{\sum_t \gamma(\ell_{tc})\mathbf{o}_t}{\sum_t \gamma(\ell_{tc}) + r} + \frac{r\mu_c^{\text{ubm}}}{\sum_t \gamma(\ell_{tc}) + r} \\ &= \alpha_c E_c(\mathcal{O}^{(s)}) + (1 - \alpha_c)\mu_c^{\text{ubm}},\end{aligned} \tag{3.16}$$

where

$$\alpha_c = \frac{n_c}{n_c + r}. \tag{3.17}$$

The relevance factor r is typically set to 16. Figure 3.5 shows the MAP adaptation process and Figure 3.6 illustrates a two-dimensional examples of the adaptation process. Note that in practice only the mean vectors will be adapted.

According to Eq. 3.14 and Eq. 3.17, $\alpha_c \to 1$ when $\mathcal{O}^{(s)}$ comprises lots of vectors (long utterances) and $\alpha_c \to 0$ otherwise. This means that $\mu_c^{(s)}$ will be closed to the observed vectors from Speaker s when the utterance is long and will be similar to the cth Gaussian of the UBM when not many frames are aligned to the cth mixture. This property agrees with the Bayesian philosophy.

In Eq. 3.17, r determines the minimum number of frames aligning to mixture c to have the adaptation effect on $\mu_c^{(s)}$. Specifically, when r is very small, say $r = 1$, a very small number of frames aligning to mixture c will be enough to cause the mean vector $\mu_c^{(s)}$ to be adapted to $\mathcal{O}^{(s)}$. On the other hand, when r is large, say $r = 30$, a lot more frames aligning to mixture c are required to have any adaptation effect.

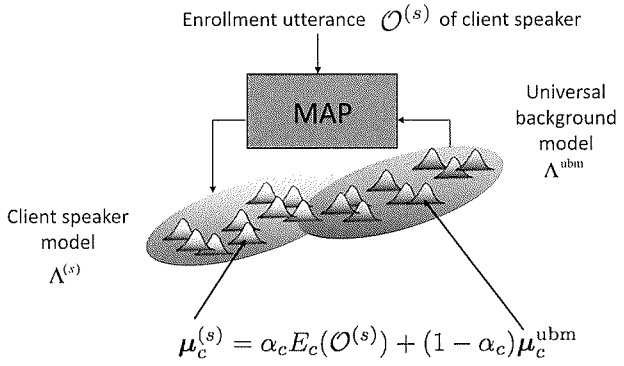

Figure 3.5 The MAP adaptation process.

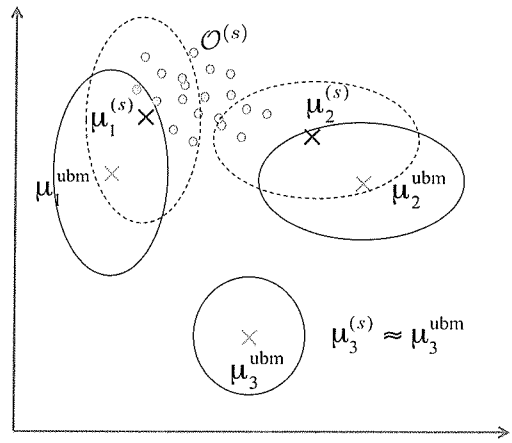

Figure 3.6 A two-dimensional example illustrating the adaptation of mixture components in a UBM. μ_1^{ubm} and μ_2^{ubm} will move toward the speaker-dependent samples $\mathcal{O}^{(s)}$ as the samples are close enough to these two Gaussians, whereas μ_3^{ubm} will remain unchanged because the samples are too far away from it.

3.1.4 GMM–UBM Scoring

Given the acoustic vectors $\mathcal{O}^{(t)}$ from a test speaker and a claimed identity s, speaker verification can be formulated as a two-class hypothesis problem:

- H_0: $\mathcal{O}^{(t)}$ comes from the true speaker s
- H_1: $\mathcal{O}^{(t)}$ comes from an impostor

Verification score is a log-likelihood ratio:

$$S_{\text{GMM–UBM}}(\mathcal{O}^{(t)}|\Lambda^{(s)}, \Lambda^{\text{ubm}}) = \log p(\mathcal{O}^{(t)}|\Lambda^{(s)}) - \log p(\mathcal{O}^{(t)}|\Lambda^{\text{ubm}}), \tag{3.18}$$

where $\log p(\mathcal{O}^{(t)}|\Lambda^{(s)})$ is the log-likelihood of $\mathcal{O}^{(t)}$ given the speaker model $\Lambda^{(s)}$, which is given by

$$\log p(\mathcal{O}^{(t)}|\Lambda^{(s)}) = \sum_{\mathbf{o}\in\mathcal{O}^{(t)}} \log \sum_{c=1}^{C} \pi_c^{\text{ubm}} \mathcal{N}(\mathbf{o}|\mu_c^{(s)}, \Sigma_c^{\text{ubm}}). \tag{3.19}$$

Note that only the mean vectors in the speaker model are speaker-dependent, i.e., $\Lambda^{(s)} = \{\pi_c^{\text{ubm}}, \mu_c^{(s)}, \Sigma_c^{\text{ubm}}\}_{c=1}^{C}$. This is because only the means are adapted in practice. Figure 3.5 shows the scoring process.

A side benefit of MAP adaptation is that it keeps the correspondence between the mixture components of the target-speaker model $\Lambda^{(s)}$ and the UBM. This property allows for fast scoring when the GMM and the UBM cover a large region of the feature space so that only a few Gaussians contribute to the likelihood value for each acoustic vector $\mathbf{o}$. Figure 3.6 illustrates such situation in which the contribute of the third Gaussian (with mean μ_3^{ubm}) can be ignored for any test vectors far away from it. Therefore, we may express the log-likelihood ratio of a test utterance with acoustic vectors $\mathcal{O}^{(t)}$ as

$$
\begin{aligned}
&S_{\text{GMM–UBM}}(\mathcal{O}^{(t)}|\Lambda^{(s)}, \Lambda^{\text{ubm}}) = \log p(\mathcal{O}^{(t)}|\Lambda^{(s)}) - \log p(\mathcal{O}^{(t)}|\Lambda^{\text{ubm}}) \\
&= \sum_{\mathbf{o}\in\mathcal{O}^{(t)}} \left[\log \sum_{c=1}^{C} \pi_c^{\text{ubm}} \mathcal{N}(\mathbf{o}|\boldsymbol{\mu}_c^{(s)}, \boldsymbol{\Sigma}_c^{\text{ubm}}) - \log \sum_{c=1}^{C} \pi_c^{\text{ubm}} \mathcal{N}(\mathbf{o}|\boldsymbol{\mu}_c^{\text{ubm}}, \boldsymbol{\Sigma}_c^{\text{ubm}}) \right] \\
&\approx \sum_{\mathbf{o}\in\mathcal{O}^{(t)}} \left[\log \sum_{c\in\Omega} \pi_c^{\text{ubm}} \mathcal{N}(\mathbf{o}|\boldsymbol{\mu}_c^{(s)}, \boldsymbol{\Sigma}_c^{\text{ubm}}) - \log \sum_{c\in\Omega} \pi_c^{\text{ubm}} \mathcal{N}(\mathbf{o}|\boldsymbol{\mu}_c^{\text{ubm}}, \boldsymbol{\Sigma}_c^{\text{ubm}}) \right], \qquad (3.20)
\end{aligned}
$$

where Ω is a set of indexes corresponding to the C' largest likelihoods in $\{\pi_c^{\text{ubm}} \mathcal{N}(\mathbf{o}|\boldsymbol{\mu}_c^{\text{ubm}}, \boldsymbol{\Sigma}_c^{\text{ubm}})\}_{c=1}^{C}$. Typically, $C' = 5$. The third equation in Eq. 3.20 requires $C + C'$ Gaussian evaluations for each $\mathbf{o}$, whereas the second equation requires $2C$ Gaussian evaluations. When $C = 1024$ and $C' = 5$, substantial computation saving can be achieved.

3.2 Gaussian Mixture Model–Support Vector Machines

A drawback of GMM–UBM systems is that the GMMs and UBM are trained separately, which means that information that discriminates the target speakers from the background speakers cannot be fully utilized. In 2006, Campbell [14] proposed to turn the GMMs into vectors so that target-speakers' GMMs and background-speakers' GMMs can be treated as positive- and negative-class vectors for training discriminative classifiers, one for each target speaker. Because of the high dimensionality of the resulting vectors, linear support vector machines are a natural choice.

3.2.1 Support Vector Machines

To understand the concepts of support vector machines (SVMs), we need to ask ourself "what is a good decision boundary?" Consider a two-class linearly separable classification problem shown in Figure 3.7. There are plenty of methods to find a boundary that can separate the two classes. A naive way is to find a line that is perpendicular to the line joining the two centers and is of equal distance to the centers, as shown in Figure 3.7. Setting the boundary position based on the centers of the two classes means that all samples are considered equally important. But is the boundary in Figure 3.7 good? Obviously it is not.

In 1995, Vapnik [33] advocated finding a subset of vectors that are more relevant to the classification task and using these vectors to define the decision boundary. Moreover, the decision boundary should be as far away from the data of both classes as possible (of course, not infinitely far). These two criteria can be fulfilled by maximizing the margin shown in Figure 3.8. In the figure, relevant vectors highlighted by the big circles are called support vectors, which give rise to the name support vector machines.

Linearly Separable Problems

In Figure 3.8, the decision boundary is given by

$$\mathbf{w}^{\mathsf{T}}\mathbf{x} + b = 0.$$

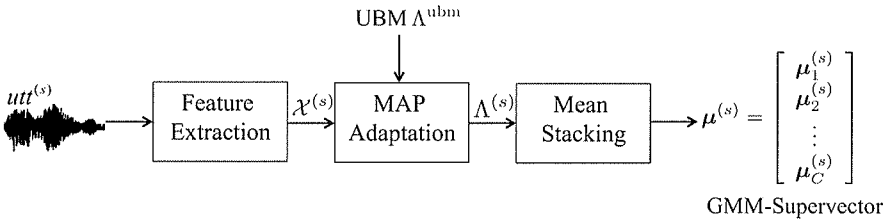

Figure 3.14 Extraction of a GMM-supervector from an utteracne.

3.2.2 GMM Supervectors

To apply SVMs for speaker verification, it is necessary to convert a variable-length utterance into a fixed-length vector, the so-called vectorization process. Campbell et al. [14] proposed to achieve such task by stacking the mean vectors of a MAP-adapted GMM as shown in Figure 3.14. Given the speech of a client speaker, MAP adaptation (Eq. 3.16) is applied to create his/her GMM model. Then, the mean vectors of the speaker model are stacked to form a supervector with dimension CF, where C is the number of Gaussians in the UBM and F is the dimension of the acoustic vectors. Typically, $C = 1024$ and $F = 60$, which result in supervectors with 61,440 dimensions.

Because of this high dimensionality, it is sensible to use linear SVMs to classify the supervectors. To incorporate the covariance matrices and the mixture coefficients of the UBM into the SVM, Campbell et al. suggest using a linear kernel of the form:[5]

$$K\left(\text{utt}^{(i)}, \text{utt}^{(j)}\right) = \sum_{c=1}^{C} \left(\sqrt{\pi_c}\boldsymbol{\Sigma}_c^{-\frac{1}{2}}\boldsymbol{\mu}_c^{(i)}\right)^{\mathsf{T}} \left(\sqrt{\pi_c}\boldsymbol{\Sigma}_c^{-\frac{1}{2}}\boldsymbol{\mu}_c^{(j)}\right), \tag{3.37}$$

where i and j index to the ith and jth utterances, respectively, and π_c, $\boldsymbol{\Sigma}_c$ and $\boldsymbol{\mu}_c$ are the weight, covariance matrix, and mean vector of the cth mixture, respectively. Eq. 3.37 can be written in a more compact form:

$$\begin{aligned} K\left(\text{utt}^{(i)}, \text{utt}^{(j)}\right) &= \left(\boldsymbol{\Omega}^{-\frac{1}{2}}\overrightarrow{\boldsymbol{\mu}}^{(i)}\right)^{\mathsf{T}} \left(\boldsymbol{\Omega}^{-\frac{1}{2}}\overrightarrow{\boldsymbol{\mu}}^{(j)}\right) \\ &\equiv K\left(\overrightarrow{\boldsymbol{\mu}}^{(i)}, \overrightarrow{\boldsymbol{\mu}}^{(j)}\right) \end{aligned} \tag{3.38}$$

where

$$\boldsymbol{\Omega} = \text{diag}\left\{\pi_1^{-1}\boldsymbol{\Sigma}_1, \ldots, \pi_C^{-1}\boldsymbol{\Sigma}_C\right\} \quad \text{and} \quad \overrightarrow{\boldsymbol{\mu}} = \left[\boldsymbol{\mu}_1^{\mathsf{T}}, \ldots, \boldsymbol{\mu}_C^{\mathsf{T}}\right]^{\mathsf{T}}. \tag{3.39}$$

In practice, $\boldsymbol{\Sigma}_c$'s are assumed to be diagonal.

For each target speaker, a set of target-speaker's supervectors are obtained from his/her enrollment utterances, one for each utterance. Then, a linear SVM is trained to discriminate his/her supervector(s) from a set of supervectors derived from a number of background speakers. After training, for target-speaker s, we have

[5] To avoid cluttering with symbols, the superscript "ubm" in $\boldsymbol{\Sigma}_c$ and π_c is omitted.

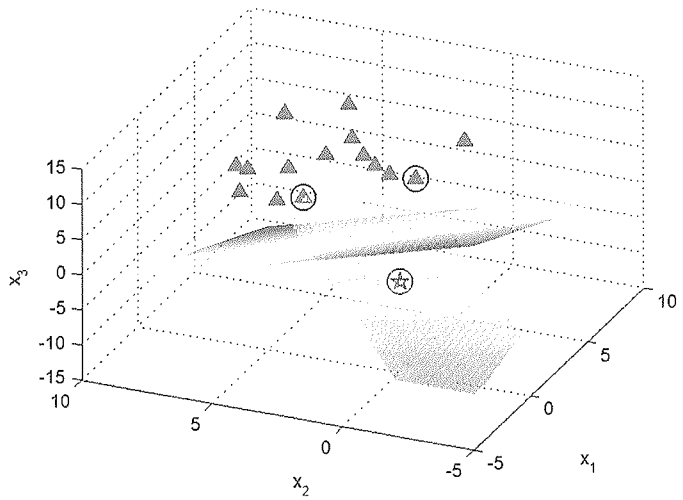

Figure 3.15 Illustration of a linear SVM in three-dimensional space when solving a data-imbalance problem in which the minority-class samples are represented by pentagrams and the majority-class samples are represented by filled triangles. The shaded area under the decision plane represents the possible locations of the minority-class samples. [Reprinted from *Acoustic vector resampling for GMMSVM-based speaker verification (Figure 1), M.W. Mak and W. Rao, Proceedings of Annual Conference of International Speech Communication Association, 2010, pp. 1449–1452, with permission of ISCA.*]

$$f^{(s)}(\vec{\mu}) = \sum_{i \in \mathcal{S}_s} \alpha_i^{(s)} K\left(\vec{\mu}^{(i)}, \vec{\mu}\right) - \sum_{i \in \mathcal{S}_{\text{bkg}}} \alpha_i^{(s)} K\left(\vec{\mu}^{(i)}, \vec{\mu}\right) + b^{(s)}, \tag{3.40}$$

where $\mathcal{S}_s$ and $\mathcal{S}_{\text{bkg}}$ are the support vector indexes corresponding to the target speaker and background speakers, respectively, and $\alpha_i^{(s)}$'s and $b^{(s)}$ are the Lagrange multipliers and the bias term of the target-speaker's SVM.

In practical situations, the number of target-speaker utterances is very small. This create a severe data-imbalance problem [35] in which the decision boundary is largely defined by the supervectors of the nontarget speakers. Another issue caused by data imbalance is that the SVM's decision boundary tends to skew toward the minority class [36, 37]. This will lead to a large number of false rejections unless the decision threshold has been adjusted to compensate for the bias. Figure 3.15 illustrates such situation. In the figure, there is a large region in the supervector space in which the target-speaker's supervector will not affect the orientation of the decision boundary.

The data-imbalance problem in GMM–SVM systems can be overcome by creating more supervectors from the utterance(s) of the target speakers [35, 38, 39]. Figure 3.16 illustrates the procedure, which is called utterance-partitioning with acoustic vector resampling (UP-VAR). The goal is to increase the number of sub-utterances without compromising their representation power. This is achieved by the following steps:

1. Randomly rearrange the sequence of acoustic vectors in an utterance;
2. Partition the acoustic vectors of an utterance into N segments;
3. Repeated Step 1 and Step 2 R times.

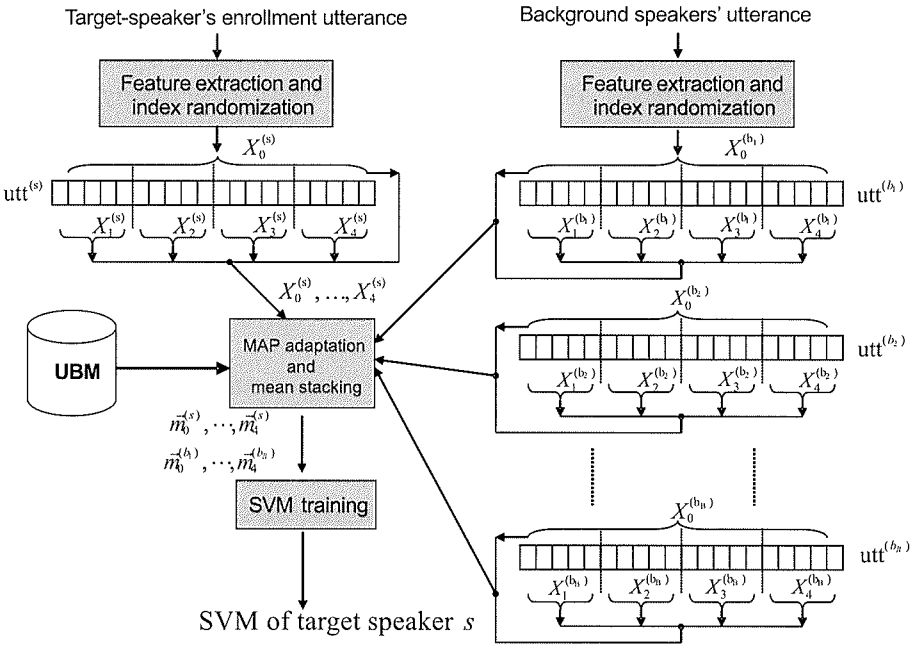

Figure 3.16 Illustration of the utterance partitioning with acoustic vector resampling (UP-AVR) process that increases the number of supervectors from a target speaker for training a speaker-dependent SVM. [Reprinted from *Utterance Partitioning with Acoustic Vector Resampling for GMM-SVM Speaker Verification (Figure 2), M.W. Mak and W. Rao, Speech Communication, vol. 53, no. 1, Jan. 2011, pp. 119–130, with permission of Elsevier.*]

By repeating Step 1 and Step 2 R times, we obtain $RN+1$ target-speaker's supervectors for training the speaker-dependent SVM.

3.2.3 GMM–SVM Scoring

During scoring, given a test utterance $\text{utt}^{(t)}$ and the SVM of a target-speaker s, we first apply MAP adaptation to the UBM to obtain the corresponding supervector $\boldsymbol{\mu}^{(t)}$ (see Figure 3.14). Then, we compute the GMM–SVM score as follows:

$$S_{\text{GMM-SVM}}(s,t) = \sum_{i\in\mathcal{S}_s} \alpha_i^{(s)} K\left(\vec{\mu}^{(i)}, \vec{\mu}^{(t)}\right) - \sum_{i\in\mathcal{S}_{\text{bkg}}} \alpha_i^{(s)} K\left(\vec{\mu}^{(i)}, \vec{\mu}^{(t)}\right) + b^{(s)}. \quad (3.41)$$

Therefore, GMM–SVM scoring amounts to finding the distance of the test supervector from the speaker-dependent hyperplane defined by the SVM of the target speaker.

Comparing the first line of Eq. 3.20 and Eq. 3.41, we can see the similarity between the GMM–UBM scoring and GMM–SVM scoring. In particular, both have two terms, one from the target speaker and one from the background speakers. However, there are also important differences. First, the two terms in GMM–UBM scoring are unweighted, meaning that they have equal contribution to the score. On the other hand, the two terms

in GMM–SVM are weighted by Lagrange multipliers, which are optimized to produce the best discrimination between the target speaker and the background speakers. Second, the speaker model $\Lambda^{(s)}$ and Λ^{ubm} in GMM–UBM are separately trained, whereas the Lagrange multipliers in GMM–SVM are jointly trained by the SVM optimizer. This means that for each speaker, the optimizer will automatically find a set of *important* utterances from his/her enrollment sessions and the background speakers. This speaker-dependent selection of utterances make the GMM–SVM systems more flexible and perform much better than GMM–UBM systems.

3.2.4 Nuisance Attribute Projection

Because the vectorization process in Figure 3.14 will collect both the speaker and nonspeaker (e.g., channel) statistics from the input utterance through Eq. 3.3 and Eq. 3.16, the GMM-supervector $\overrightarrow{\mu}$ will contain not only speaker but also other nonspeaker information. Therefore, it is important to remove such unwanted information before performing GMM–SVM scoring. One of the most promising approaches is the nuisance attribute projection (NAP) [40–42].

Basic Idea of NAP

The idea of NAP is to find a subspace within the GMM-supervector space in which all nonspeaker variabilities occur. The method requires a training set comprising multiple speakers and multiple recording sessions per speaker. Assume that we are given a training set comprising N GMM-supervectors $\mathcal{X} = \{\overrightarrow{\mu}^{(1)}, \ldots, \overrightarrow{\mu}^{(N)}\}$. Our goal is to find a subspace defined by the column vectors in $\boldsymbol{U}$ and define a projection matrix

$$\mathbf{P} \equiv \mathbf{I} - \mathbf{U}\mathbf{U}^{\mathsf{T}}, \tag{3.42}$$

where $\mathbf{I}$ is an identity matrix, such that the projected vectors

$$\tilde{\mu}^{(i)} \equiv \mathbf{P}\overrightarrow{\mu}^{(i)} = (\mathbf{I} - \mathbf{U}\mathbf{U}^{\mathsf{T}})\overrightarrow{\mu}^{(i)}, \quad i = 1, \ldots, N \tag{3.43}$$

retain most of the speaker information but with nonspeaker information suppressed. For $\tilde{\mu}^{(i)}$'s to retain most of the speaker information, the rank of $\mathbf{U}$ should be much lower than the dimension of $\overrightarrow{\mu}^{(i)}$'s.

Figure 3.17 illustrates the concept of suppressing session variability in NAP. In the figure, the superscript h is the session index, and all sessions (supervectors) from the same speaker s – indexed by (s,h) – lie on the line defined by the single column in $\mathbf{U}$. By applying Eq. 3.43, we obtain a projected vector $\tilde{\mu}^{(s)}$ independent of the session index h.

Figure 3.18 demonstrates the capability of NAP in a three-dimensional toy problem in which nonspeaker variability occurs along the x_1-axis, i.e., $\boldsymbol{U} = [1\ 0\ 0]^{\mathsf{T}}$. In the figure, the feature vectors $\mathbf{x}$'s come from two different speakers, one for each cluster. Because $\boldsymbol{U} = [1\ 0\ 0]^{\mathsf{T}}$ is of rank 1, $\mathbf{U}\mathbf{U}^{\mathsf{T}}\mathbf{x}$'s vary along a straight line and $(\mathbf{I} - \mathbf{U}\mathbf{U}^{\mathsf{T}})\mathbf{x}$'s lie on a plane perpendicular to the line. As shown in the figure, after NAP projection, there is no direction in which the two clusters (black ○) are indistinguishable.