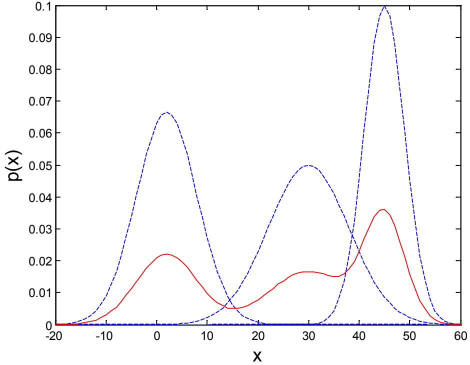
Outline

Outline

GMM distribution from three Gaussians



Gaussian mixture model

- Gaussian mixture model (GMM) is a weighted sum of Gaussians

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{i=1}^{M} \pi_i b_i(\mathbf{x})$$
$$\boldsymbol{\theta} = \{\pi_i, \mathbf{u}_i, \Sigma_i\}$$

π_i: mixture weight
$\mathbf{u}_i$: mixture mean vector
Σ_i: mixture covariance matrix

$$b_i(\mathbf{x}) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{u}_i)^\top \Sigma_i^{-1}(\mathbf{x}-\mathbf{u}_i)\right)$$

- Mixture component z_i is a latent variable which is either zero or one

Maximum likelihood

- In ML estimation, we need to
  - compute the likelihood of a sequence of features given a GMM
  - estimate the parameters of GMM given a set of feature vectors
- Assuming independence between features in a sequence, we have

$$p(X|\boldsymbol{\theta}) = p(\mathbf{x}_1, \ldots, \mathbf{x}_T|\boldsymbol{\theta}) = \prod_{t=1}^{T} p(\mathbf{x}_t|\boldsymbol{\theta})$$

- ML estimation is performed by

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} p(X|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \sum_{t=1}^{T} \log \left[\sum_{i=1}^{M} \pi_i b_i(\mathbf{x}_t) \right]$$

Parameter estimation

- E-step is to calculate the auxiliary function

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \mathbb{E}_{\mathbf{z}}[\log p(X, \mathbf{z}|\boldsymbol{\theta})|X, \boldsymbol{\theta}^{\text{old}}]$$
$$= \sum_{t=1}^{T} \sum_{i=1}^{M} p(z_{ti} = 1|X, \boldsymbol{\theta}^{\text{old}}) \log p(\mathbf{x}_t, z_{ti} = 1|\boldsymbol{\theta})$$

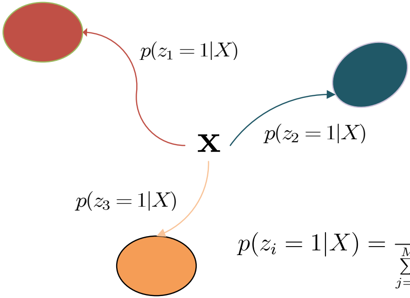
- ML estimates are obtained via M-step as

$$\pi_i^{\text{new}} = \frac{T_i}{T}$$
$$\boldsymbol{\mu}_i^{\text{new}} = \frac{1}{T_i} \sum_{t=1}^{T} \gamma(z_{ti}) \mathbf{x}_t = \frac{1}{T_i} \mathbb{E}_i[\mathbf{x}]$$
$$\Sigma_i^{\text{new}} = \frac{1}{T_i} \sum_{t=1}^{T} \gamma(z_{ti})(\mathbf{x}_t - \boldsymbol{\mu}_i)(\mathbf{x}_t - \boldsymbol{\mu}_i)^\top = \frac{1}{T_i} \mathbb{E}_i[\mathbf{x}\mathbf{x}^\top] - \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top$$

where $T_i = \sum_t \gamma(z_{ti})$ and $\gamma(z_{ti}) = p(z_{ti} = 1|X, \boldsymbol{\theta}^{\text{old}})$

E-step

Probabilistically align samples to each mixture



$$p(z_i = 1|X) = \frac{\pi_i b_i(\mathbf{x})}{\sum_{j=1}^{M} \pi_j b_j(\mathbf{x})}$$

Accumulate sufficient statistics

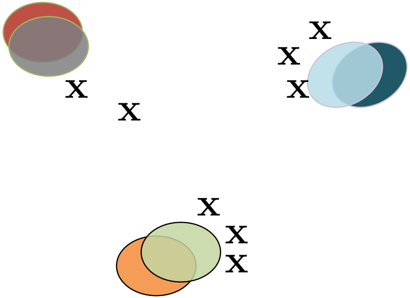
$$T_i = \sum_{t=1}^{T} p(z_{ti} = 1|X)$$

$$\mathbb{E}_i(\mathbf{x}) = \sum_{t=1}^{T} p(z_{ti} = 1|X)\mathbf{x}_t$$

$$\mathbb{E}_i(\mathbf{x}\mathbf{x}^T) = \sum_{t=1}^{T} p(z_{ti} = 1|X)\mathbf{x}_t\mathbf{x}_t^T$$

M-step

Update model parameters



GMM parameters

$$\pi_i^{\text{new}} = \frac{T_i}{T}$$

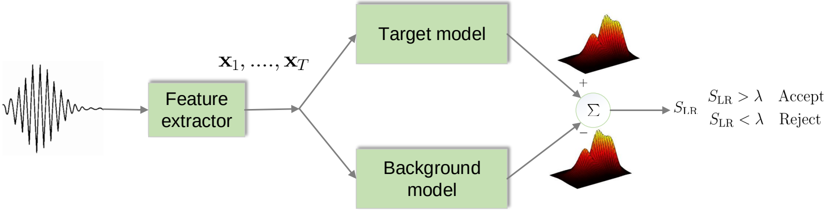
$$\boldsymbol{\mu}_i^{\text{new}} = \frac{1}{T_i}\mathbb{E}_i[\mathbf{x}]$$

$$\Sigma_i^{\text{new}} = \frac{1}{T_i}\mathbb{E}_i[\mathbf{x}\mathbf{x}^T] - \boldsymbol{\mu}_i\boldsymbol{\mu}_i^T$$

Speaker verification

- Realization of log likelihood ratio test from signal detection theory

$$S_{\mathsf{LR}}(X|\boldsymbol{\theta}^{\mathsf{target}}, \boldsymbol{\theta}^{\mathsf{ubm}}) = \log(X|\boldsymbol{\theta}^{\mathsf{target}}) - \log(X|\boldsymbol{\theta}^{\mathsf{ubm}})$$



- GMMs are used for both target and background models
  - target model trained using enrollment speech
  - universal background model trained using speech from many speakers

Target model & UBM

- Target model is adapted from universal background model (UBM)
  - good with limited target training data
- Maximum *a posteriori* (MAP) adaptation
  - align target training vectors to UBM
  - accumulate sufficient statistics
  - update target model parameters with smoothing to UBM parameters
- Adaptation for those parameters of seen acoustic events
  - sparse regions of feature space filled in by UBM parameters
- Side benefits
  - keep correspondence between target and UBM mixtures
  - allow for fast scoring when using many target models (top-M scoring)

Maximum *a posteriori adaptation*

- Prior density for GMM mean vector $\boldsymbol{\mu} = \{\boldsymbol{\mu}_i\}$ is introduced

$$p(\boldsymbol{\mu}_i) = \mathcal{N}(\boldsymbol{\mu}_i | \boldsymbol{\mu}_i^{\text{ubm}}, \sigma^2 \mathbf{I})$$

- MAP estimation [Gauvain and Lee, 1994] is performed by using the enrollment data $X_s = \{\mathbf{x}\}_{t=1}^{T_s}$ from a target speaker s
  - E-step is to calculate

$$Q(\boldsymbol{\mu}_i, \boldsymbol{\mu}_i^{\text{old}}) = \sum_{t=1}^{T_s} \gamma(z_{ti}) \log p(\mathbf{x}_t | z_{ti} = 1, \boldsymbol{\mu}_i) + \log p(\boldsymbol{\mu}_i | \boldsymbol{\mu}_i^{\text{ubm}}, \sigma^2 \mathbf{I})$$

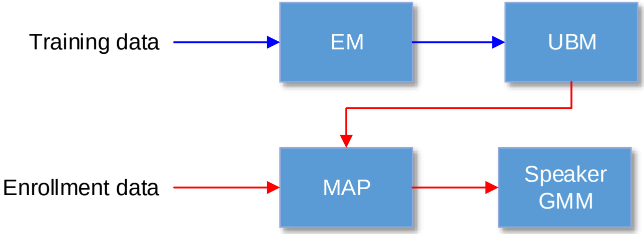
  - M-step is to maximize $Q(\boldsymbol{\mu}_i, \boldsymbol{\mu}_i^{\text{old}})$ to find

$$\boldsymbol{\mu}_i^{\text{new}} = \frac{\sum_t \gamma(z_{ti}) \mathbf{x}_t}{\sum_t \gamma(z_{ti}) + r} + \frac{r \boldsymbol{\mu}_i^{\text{ubm}}}{\sum_t \gamma(z_{ti}) + r} = \alpha_i \mathbb{E}_i[\mathbf{x}] + (1 - \alpha_i) \boldsymbol{\mu}_i^{\text{ubm}}$$

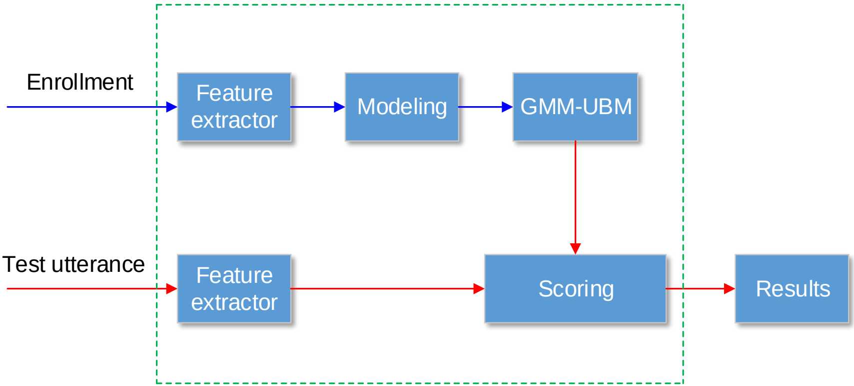
where $\alpha_i = \frac{\sum_t \gamma(z_{ti})}{\sum_t \gamma(z_{ti}) + r}$

MAP adaptation for GMM-UBM

- UBM is based on GMM and trained by using EM algorithm
- Speaker GMM is established by adjusting UBM by using MAP adaptation



Speaker recognition procedure



Outline

Joint factor analysis

- Factor analysis is a statistical method which is used to describe the variability among the observed variables in terms of potentially lower number of unobserved variables called factors
- Factor analysis is a latent variable model for feature extraction
- Joint factor analysis (JFA) was the initial paradigm for speaker recognition

$$\underbrace{\mathbf{u}}_{\text{Speaker Supervector}} = \underbrace{\mathbf{m}}_{\text{UBM}} + \mathbf{V}\overbrace{\mathbf{y}}^{\text{Speaker factors}} + \mathbf{U}\underbrace{\mathbf{x}}_{\text{Channel factors}} + \mathbf{D}\overbrace{\mathbf{z}}^{\text{Residual factors}}$$

Intuition & interpretation

- A supervector for a speaker should be decomposable into speaker independent, speaker dependent, channel dependent, and residual components
- Each component is represented by low-dimensional factors, which operate along the principal dimensions of the corresponding component
- Speaker dependent component, known as the eigenvoice, and the corresponding factors

Eigenvoice matrix

$$\mathbf{V} \cdot \mathbf{y} = \begin{bmatrix} | & | & | & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_N \\ | & | & | & | \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

Each speaker factor controls an eigendimension of the eigenvoice matrix

Low dimensional eigenvoice factors

Factor decomposition

- GMM supervector $\mathbf{u}$ for a speaker can be decomposed as

$$\underbrace{\mathbf{u}}_{\text{Speaker supervector}} = \underbrace{\mathbf{m}}_{\text{Speaker-independent component}} + \overbrace{\mathbf{V}\mathbf{y}}^{\text{Speaker-dependent component}} + \underbrace{\mathbf{U}\mathbf{x}}_{\text{Channel-dependent component}} + \overbrace{\mathbf{D}\mathbf{z}}^{\text{Speaker-dependent resuidual component}}$$

  where
  - $\mathbf{m}$ is a speaker-independent supervector from UBM
  - $\mathbf{V}$ is the eigenvoice matrix
  - $\mathbf{y} \sim \mathcal{N}(0, \mathbf{I})$ is the speaker factor vector
  - $\mathbf{U}$ is the eigenchannel matrix
  - $\mathbf{x} \sim \mathcal{N}(0, \mathbf{I})$ is the channel factor vector
  - $\mathbf{D}$ is the residual matrix, and is diagonal
  - $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ is the speaker-specific residual factor vector

Dimensionality

- For a 512-mixture GMM-UBM system, the dimensions of each JFA component are typically as follows
  - **V** 20,000 by 300 (300 eigenvoices)
  - **y** 300 by 1 (300 speaker factors)
  - **U** 20,000 by 100 (100 eigenchannels)
  - **x** 100 by 1 (100 channel factors)
  - **D** 20,000 by 20,000 (20,000 residuals)
  - **z** 20,000 by 1 (20,000 speaker-specific residuals)
- These dimensions have been empirically determined to produce the best results
- Bayesian model selection can help
- Judge by the marginal likelihood over latent component under different dimensions

Training procedure

- We train the JFA matricies in the following order [Kenny et al., 2007a]
  1. Train the eigenvoice matrix $\mathbf{V}$, assuming that $\mathbf{U}$ and $\mathbf{D}$ are zero
  2. Train the eigenchannel matrix $\mathbf{U}$ given the estimate of $\mathbf{V}$, assuming that $\mathbf{D}$ is zero
  3. Train the residual matrix $\mathbf{D}$ given the estimates of $\mathbf{V}$ and $\mathbf{U}$
- Using these matrices, we compute $\mathbf{y}$ for speaker, $\mathbf{x}$ for channel, and $\mathbf{z}$ for residual factors
- We compute the final score by using these matrices and factors

Total variability

- Subspaces **U** and **V** are not completely independent
- A combined total variability space was used [Dehak et al., 2011]

Speaker factors

Residual factors

$$\mathbf{u} = \mathbf{m} + \mathbf{V}\mathbf{y} + \mathbf{U}\mathbf{x} + \mathbf{D}\mathbf{z}$$

Speaker supervector

UBM

Channel factors

$$\mathbf{u} = \mathbf{m} + \mathbf{T}\mathbf{w}$$

Intermediate/identity vector (i-vector)

Speaker supervector

UBM

Total variability matrix

Training total variability space

- Rank of $\mathbf{T}$ is set prior to training
- $\mathbf{T}$ and $\mathbf{w}$ are latent variables
- EM algorithm is used
- Training total variability matrix $\mathbf{T}$ is similar to training $\mathbf{V}$ except that training $\mathbf{T}$ is performed by using all utterances from a given speaker but as produced by different speakers
- Random initialization for $\mathbf{T}$
- Each $\mathbf{o}_t$ has dimension D. Number of Gaussian components is M. Dimension of supervector is $M \cdot D$
- UBM diagonal covariance matrix $\mathbf{\Sigma}$ $(MD \times MD)$ is introduced to model the residual variability not captured by $\mathbf{T}$

Sufficient statistics

- 0^{th} order statistics $N_c(u) = \sum_t \gamma_c(\mathbf{o}_t)$ of an utterance u
- 1^{th} order statistics $F_c(u) = \sum_t \gamma_c(\mathbf{o}_t)\mathbf{o}_t$
- 2^{nd} order statistics $S_c(u) = \mathrm{diag}\left(\sum_t \gamma_c(\mathbf{o}_t)\mathbf{o}_t\mathbf{o}_t^\top\right)$ where

$$\gamma_c(\mathbf{o}_t) = p(c|\mathbf{o}_t, \boldsymbol{\theta}_{\text{ubm}}) = \frac{\pi_c p(\mathbf{o}_t|\mathbf{m}_c, \boldsymbol{\Sigma}_c)}{\sum_{j=1}^{M} \pi_j p(\mathbf{o}_t|\mathbf{m}_j, \boldsymbol{\Sigma}_j)}$$

- Centralized 1^{th} and 2^{nd} order statistics

$$\tilde{F}_c(u) = \sum_{t=1}^{T} \gamma_c(\mathbf{o}_t)(\mathbf{o}_t - \mathbf{m}_c)$$

$$\tilde{S}_c(u) = \mathrm{diag}\left(\sum_{t=1}^{T} \gamma_c(\mathbf{o}_t)(\mathbf{o}_t - \mathbf{m}_c)(\mathbf{o}_t - \mathbf{m}_c)^\top\right)$$

where $\mathbf{m}_c$ is the subvector corresponding to mixture component c

EM algorithm

- Sufficient statistics

$$N(u) = \begin{bmatrix} N_1(u)\cdot \mathbf{I}_{D\times D} & 0 & \cdots & 0 \\ 0 & N_2(u)\cdot \mathbf{I}_{D\times D} & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & N_M(u)\cdot \mathbf{I}_{D\times D} \end{bmatrix} \quad \tilde{F}(u) = \begin{bmatrix} \tilde{F}_1(u) \\ \tilde{F}_2(u) \\ \vdots \\ \tilde{F}_M(u) \end{bmatrix}$$

- EM algorithm [Kenny et al., 2005]
  - Initialize $\mathbf{m}$, $\mathbf{\Sigma}$ and $\mathbf{T}$
  - E-step: for each utterance u, calculate the parameters of the posterior distribution of $\mathbf{w}(u)$ using the current estimates of $\mathbf{m}, \mathbf{\Sigma}, \mathbf{T}$
  - M-step: update $\mathbf{T}$ and $\mathbf{\Sigma}$ by solving a set of linear equations in which $\mathbf{w}(u)$'s play the role of explanatory variables
  - Iterate until data likelihood given the estimated parameters converges

E-step: posterior distribution of $\mathbf{w}(u)$

- For each utterance u, we calculate the matrix

$$\mathbf{L}(u) = \mathbf{I} + \mathbf{T}^\top \mathbf{\Sigma}^{-1} N(u) \mathbf{T}$$

- Posterior distribution of $w(u)$ conditioned on the acoustic observations of an utterance u is Gaussian with mean

$$\mathbb{E}[\mathbf{w}(u)] = \mathbf{L}^{-1}(u) \mathbf{T}^\top \mathbf{\Sigma}^{-1} \tilde{F}(u)$$

  and covariance matrix

$$\mathrm{Cov}(\mathbf{w}(u), \mathbf{w}(u)) = \mathbf{L}^{-1}(u)$$

- Variational Bayesian JFA was developed for speaker verification [Zhao and Dong, 2012]

Linear discriminant analysis

- I-vectors from JFA model are used in linear discriminant analysis (LDA)

$$\mathbf{u} = \mathbf{m} + \mathbf{T}\mathbf{w} \longleftarrow \text{Factor analysis}$$

$$\mathcal{W} = \mathbf{A}\mathbf{w} \longleftarrow \text{Linear discriminant analysis}$$

- Both methods used to reduce the dimensionality of speaker model
- **A** is chosen such that within-speaker variability S_w is minimized and between-speaker variability S_b is maximized within the space
- **A** is found by eigenvalue method via maximizing

$$\mathcal{J}(\mathbf{A}) = \mathrm{Tr}\{S_w^{-1} S_b\}$$

Outline

Factor analysis

- Assuming a factor analysis model of the i-vectors of the form

$$\mathbf{w} = \mathbf{u} + \mathbf{F}\mathbf{h} + \varepsilon$$

- $\mathbf{w}$ is the i-vector, $\mathbf{u}$ is the mean of i-vectors, and $\mathbf{h} \sim \mathcal{N}(0, \mathbf{I})$ is the latent factors
- First compute the maximum likelihood estimate of the factor loading matrix $\mathbf{F}$, also known as the eigenvoice subspace
- Full covariance of residual noise ε explains the variability not captured through the latent variables

PLDA

Under Gaussian assumption, this model is known in face recognition as PLDA [Prince and Elder, 2007]

Gaussian PLDA

- Assume that there are low dimensional, normally distributed hidden variables $\mathbf{x}_1$ and $\mathbf{x}_{2r}$ such that

$$D_r = \mathbf{m} + \underbrace{U_1 \mathbf{x}_1}_{S} + \underbrace{U_2 \mathbf{x}_{2r} + \varepsilon_r}_{C_r}$$

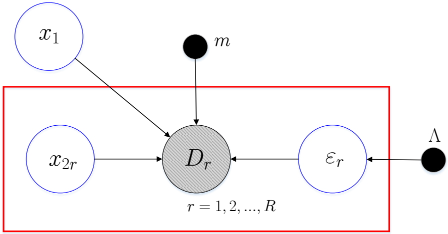
- Residual ε_r is normally distributed with mean 0 and precision matrix Λ
- $\mathbf{m}$ is the center of acoustic space and $\mathbf{x}_1$ is the speaker factors
- Columns of U_1 are the eigenvoices

$$\mathrm{Cov}(S, S) = U_1 U_1^\top$$

- $\mathbf{x}_{2r}$ varies from one recording to another (channel factors)
- Columns of U_2 are the eigenchannels

$$\mathrm{Cov}(C_r, C_r) = \Lambda^{-1} + U_2 U_2^\top$$

Graphical representation



- Including $\mathbf{x}_{2r}$ enables the decomposition of speaker and channel factors
- $\mathbf{x}_{2r}$ can *always* be eliminated at recognition time
- Between-speaker covariance matrix $\mathrm{Cov}(S, S)$ & within-speaker covariance matrix $\mathrm{Cov}(C_r, C_r)$
- These matrices cannot be treated as full rank

PLDA speaker recognition

- Given two i-vectors D_1 and D_2, we would like to perform the hypothesis test

  H_1: the speakers are the same

  H_0: the speakers are different

- Likelihood ratio is calculated by

$$\frac{p(D_1, D2|H_1)}{p(D_1|H_0)p(D_2|H_0)}$$

- Likelihood ratio for any type of speaker recognition or speaker clustering problem
- The evidence integral should be calculated

$$\int p(D, \mathbf{z})d\mathbf{z}$$

Model assumption

- Assume that
  - we have succeeded in estimating the model parameters $\boldsymbol{\theta} = \{\mathbf{m}, U_1, U_2, \Lambda\}$
  - given a collection $D = (D_1, \ldots, D_R)$ of i-vectors associated with a speaker, we have figured out how to evaluate the marginal likelihood or the evidence

$$p(D) = \int p(D, \mathbf{z}) d\mathbf{z} = \int p(D|\mathbf{z}) p(\mathbf{z}) d\mathbf{z}$$

  - $\mathbf{z} = \{\mathbf{x}_1, \mathbf{x}_{2r}\}$ is the hidden variables associated with the speaker

- We show how to do speaker recognition in this situation and how both problems are tackled by using variational Bayes to approximate the posterior distribution $p(\mathbf{z}|D)$

Variational approximation

- Evidence $p(D)$ can be evaluated exactly in the Gaussian case but this involves inverting the large sparse block matrices
- If $q(\mathbf{z})$ is any distribution on $\mathbf{z}$, variational lower bound is yielded as

$$\mathcal{L} \triangleq \mathbb{E}_q \left[\log \frac{p(D, \mathbf{z})}{q(\mathbf{z})} \right]$$

  where $\log p(D) \geq \mathcal{L}$ with equality iff $q(\mathbf{z}) = p(\mathbf{z}|D)$
- Variational Bayes provides a principled way to find a good approximation $q(\mathbf{z})$ to $p(\mathbf{z}|D)$
- Model parameters $\boldsymbol{\theta} = \{\mathbf{m}, U_1, U_2, \Lambda\}$ are estimated by maximizing the evidence lower bound (ELBO) $\mathcal{L}$ which is calculated over all of the speakers in a training set

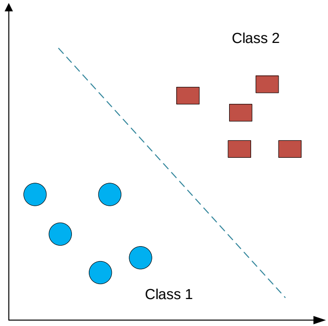
Bayesian speaker recognition

- Full Bayesian avoids the point estimates of model parameters
- Coupling of multiple latent variables is tackled in VB [Villalba and Lleida, 2014]
- Uncertainties are compensated for model regularization in speaker recognition
- Prior densities $p(U_1)$ and $p(U_2)$ can be flexibly incorporated
- Selection for the number of speaker factors or channel factors
- Manually tuning for unknown variables is avoided
- Analogous to the the treatment of the number of mixture components in Bayesian estimation of GMM
- Bayesian mixture of PLDA [Mak et al., 2016] was recently developed

Outline

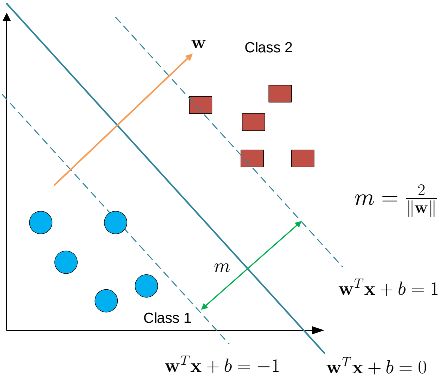
What is a good decision boundary?

- Consider a two-class, linearly separable classification problem
- Many decision boundaries!
  - perceptron algorithm can be used to find such a boundary
  - different algorithms have been proposed
- Are all decision boundaries equally good?



Large-margin decision boundary

- Decision boundary should be as far away from the data of both classes as possible [Vapnik, 2013]
  - we should maximize the margin m
  - distance between the origin and the line $\mathbf{w}^\top \mathbf{x} = k$ is $\frac{k}{\|\mathbf{w}\|}$



Finding the decision boundary

- $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ is the data set and $y_i \in \{1, -1\}$ is the class label of $\mathbf{x}_i$
- Decision boundary should classify all points correctly

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad \text{for } i = 1, \ldots, n$$

- Decision boundary can be found by solving the constrained optimization problem

$$\begin{aligned} &\text{Minimize } \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{subject to } y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad \text{for } i = 1, \ldots, n \end{aligned}$$

Constrained optimization

- Original problem

$$\text{Minimize } \frac{1}{2}\|\mathbf{w}\|^2$$
$$\text{subject to } y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad \text{for } i = 1, \ldots, n$$

- We introduce the Lagrange multipliers $\alpha_i \geq 0$ to form the Lagrangian function

$$L = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 \right)$$

- Setting the gradient of L w.r.t $\mathbf{w}$ and b to zero, we have

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$
$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

Dual problem

- New objective function is expressed in terms of α_i only
- If we know **w**, we know all α_i. If we know all α_i, we know **w**
- Original problem is known as the primal problem
- A quadratic programming objective is formed by

$$\text{maximize } L(\alpha) = \sum_{i}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \ \sum_{i=1}^{n} \alpha_i y_i = 0$$

- Global maximum of α_i can be found

Characteristics of the solution

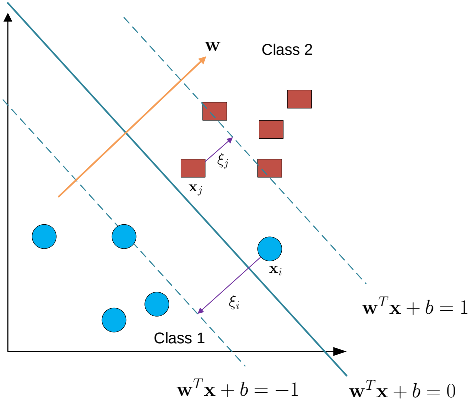
- Many of α_i are zero
  - $\mathbf{w}$ is a linear combination of a small number of data points
  - This sparse representation can be viewed as data compression as in the construction of KNN classifier
- $\mathbf{x}_i$ with non-zero α_i are called support vectors
  - decision boundary is determined only by support vectors
  - $t_j, j = 1, \dots, s$ are the indices of s support vectors

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

- For testing with a new data $\mathbf{z}$
  - compute $f = \mathbf{w}^\top \mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}^\top \mathbf{z} + b$ and classify $\mathbf{z}$ as class 1 if the sum is positive, and class 2 otherwise

Non-separable problem

- We allow error ξ_i in classification. It is based on the output of the discriminant function $\mathbf{w}^\top \mathbf{x} + b$
- ξ_i approximates the number of misclassified samples



Soft margin hyperplane

- If we minimize $\sum_i \xi_i$, ξ_i can be computed by

$$\begin{cases} \mathbf{w}^\top \mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^\top \mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

  - ξ_i are slack variables in optimization
  - $\xi_i = 0$ if there is no error for $\mathbf{x}_i$
  - ξ_i is an upper bound of the number of errors
- We want to minimize $\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^n \xi_i$
  - C is a tradeoff parameter between error and margin
- Optimization problem becomes

$$\text{Minimize } \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i$$

$$\text{subject to } y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

Dual problem

- Dual of this new constrained optimization problem is

$$\text{maximize } L(\alpha) = \sum_{i}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j$$

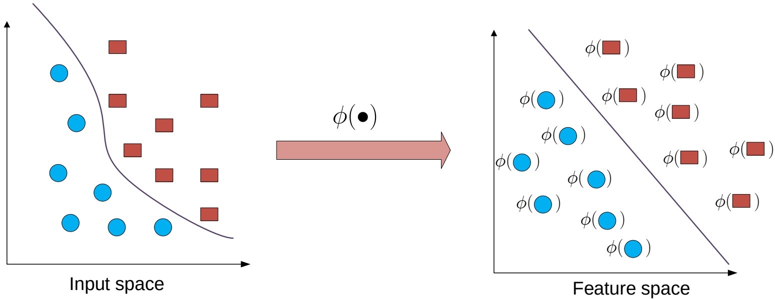
$$\text{subject to } C \geq \alpha_i \geq 0, \ \sum_{i=1}^{n} \alpha_i y_i = 0$$

- $\mathbf{w}$ is recovered as $\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$
- This is very similar to the optimization problem in the linear separable case, except that there is an upper bound C on α_i now
- Once again, a quadratic programming solver can be used to find α_i

Non-linear decision boundary

- So far, we have only considered large-margin classifier with a linear decision boundary
- How to generalize it to become non-linear?
- Key idea: transform $\mathbf{x}_i$ to a higher dimensional space to make life easier
  - input space: the points $\mathbf{x}_i$ are located
  - feature space: the space of $\phi(\mathbf{x}_i)$ after transformation
- Why transform?
  - linear operation in the feature space is equivalent to non-linear operation in input space
  - classification can become easier with a proper transformation.
  - In the XOR problem, for example, adding a new feature make the problem linearly separable

Dimensionality in feature space



- Computation in the feature space can be costly because it is highly dimensional
  - feature space is typically infinite-dimensional!
- Kernel trick comes to rescue

Kernel trick

$$\text{maximize } L(\alpha) = \sum_{i}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \ \sum_{i=1}^{n} \alpha_i y_i = 0$$

- $\mathbf{x}_i^\top \mathbf{x}_j$ is inner product
- As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly
- Many common geometric operations, e.g. angle, distance, can be expressed by inner products
- Define the kernel function K by

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$$

Kernel function in SVM

- Change all inner products to kernel functions
- For training
  - original

$$\text{maximum } L(\alpha) = \sum_{i}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \ \sum_{i=1}^{n} \alpha_i y_i = 0$$

  - with kernel function

$$\text{maximum } L(\alpha) = \sum_{i}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to } C \geq \alpha_i \geq 0, \ \sum_{i=1}^{n} \alpha_i y_i = 0$$

Kernel function in SVM

- For testing, the new data $\mathbf{z}$ is classified as class 1 if $f \geq 0$ and as class 2 if $f < 0$
  - original

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

$$f = \mathbf{w}^\top \mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}^\top \mathbf{z} + b$$

  - with kernel function

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \phi(\mathbf{x}_{t_j})$$

$$f = \langle \mathbf{w}, \phi(\mathbf{z}) \rangle + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} K(\mathbf{x}_{t_j}, \mathbf{z}) + b$$

Similarity measure

- Since the training of SVM only requires the value of $K(\mathbf{x}_i, \mathbf{x}_j)$, there is no restriction of the form of $\mathbf{x}_i$ and $\mathbf{x}_j$
  - $\mathbf{x}_i$ can be a sequence or a tree instead of a feature vector
- $K(\mathbf{x}_i, \mathbf{x}_j)$ is just a similarity measure comparing $\mathbf{x}_i$ and $\mathbf{x}_j$
- Kernel function needs to satisfy the Mercer function, i.e., the function is positive-definite
- For a test object $\mathbf{z}$, the discriminant function essentially is a weighted sum of the similarity between $\mathbf{z}$ and a pre-selected set of objects, also called the support vectors

$$f(\mathbf{z}) = \sum_{\mathbf{x}_i \in S} \alpha_i y_i K(\mathbf{z}, \mathbf{x}_i) + b$$

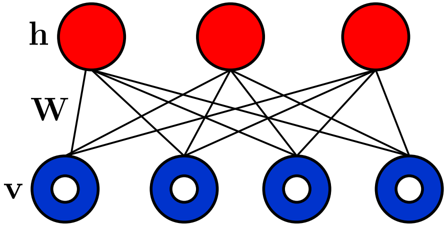
where S denotes the set of support vectors

Outline

Restricted Boltzmann machine

- Bipartite the undirected graphical model with visible variable **v** and hidden variable **h**
- Building-block for deep belief networks and deep Boltzmann machines
- RBMs are generative models of **v** based on the marginal distribution
- Joint distribution of (**v**, **h**) is an exponential family
- Discriminative fine-tuning can be applied
- Variables are typically binary, however no such restriction exists
- Bidirectional graphical model

Graphical model



- No connection between nodes of the same layer (i.e. sparsity)
- Allow fast training (blocked-Gibbs sampling)
- Correlations between nodes in v are still present in the marginal $p(\mathbf{v}|\mathbf{W})$
- Hidden variable **h** captures the higher level information

Joint distribution

- Energy-based distribution is defined by

$$p(\mathbf{v},\mathbf{h}|\boldsymbol{\theta}) = Z(\boldsymbol{\theta})^{-1} p^*(\mathbf{v},\mathbf{h}|\boldsymbol{\theta})$$

  where

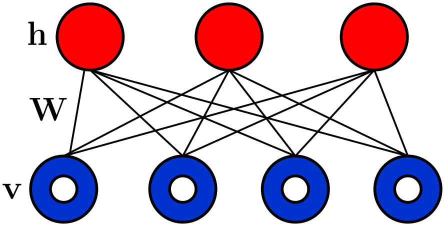
$$p^*(\mathbf{v},\mathbf{h}|\boldsymbol{\theta}) = \exp\left(\sum_i v_i b_i + \sum_j h_j a_j + \underbrace{\sum_{i,j} v_i h_j W_{ij}}_{-E(\mathbf{v},\mathbf{h})}\right)$$

  and $\boldsymbol{\theta} = \{W, \mathbf{b}, \mathbf{a}\}$

- $\{\mathbf{b}, \mathbf{a}\}$ denote the biases and are usually assumed to be zero for compact notation
- $Z(\boldsymbol{\theta}) = \sum_{\mathbf{v},\mathbf{h}} p^*(\mathbf{v},\mathbf{h}|\boldsymbol{\theta})$ is the partition function

Individual distribution

- Consider binary $(\mathbf{v}, \mathbf{h})$ with zero biases $\{\mathbf{b}, \mathbf{a}\}$
- $p(\mathbf{h}|\mathbf{v}) = \prod_j p(h_j|\mathbf{v})$ where $p(h_j|\mathbf{v}) = \frac{1}{1+\exp(-\sum_i v_i W_{ij})}$
- $p(\mathbf{v}|\mathbf{h}) = \prod_i p(v_i|\mathbf{h})$ where $p(v_i|\mathbf{h}) = \frac{1}{1+\exp(-\sum_j h_j W_{ij})}$
- Product form is due to the restricted structure



Learning with RBM

- Maximize the likelihood of $\boldsymbol{\theta}$ given $\mathbf{v}^n$

$$p(\mathbf{v}^n) = \prod_n Z^{-1} \sum_h \exp\left(\sum_{ij} v_i^n h_j^n W_{ij} \right)$$

- We obtain $\frac{\partial \log p(\mathbf{v}^n)}{\partial W_{ij}} = E_{p_{data}}[v_i h_j] - E_{p_{model}}[v_i h_j]$
  where $p_{data}(\mathbf{v}^n, \mathbf{h}^n) = p(\mathbf{h}^n|\mathbf{v}^n)p(\mathbf{v}^n)$
- $p(\mathbf{h}^n|\mathbf{v}^n)$ is an easy and exact calculation for RBM
- $p(\mathbf{v}^n)$ is an empirical distribution
- $\frac{\partial \log Z(\boldsymbol{\theta})}{\partial W_{ij}} = E_{p_{model}}[v_i h_j]$ is hard to compute
- Learning using contrastive divergence with mini-batches is performed