Nuisance Attribute Projection

BY DAVID CINCIRUK

ASPITRG GROUP MEETING

JULY 13, 2012

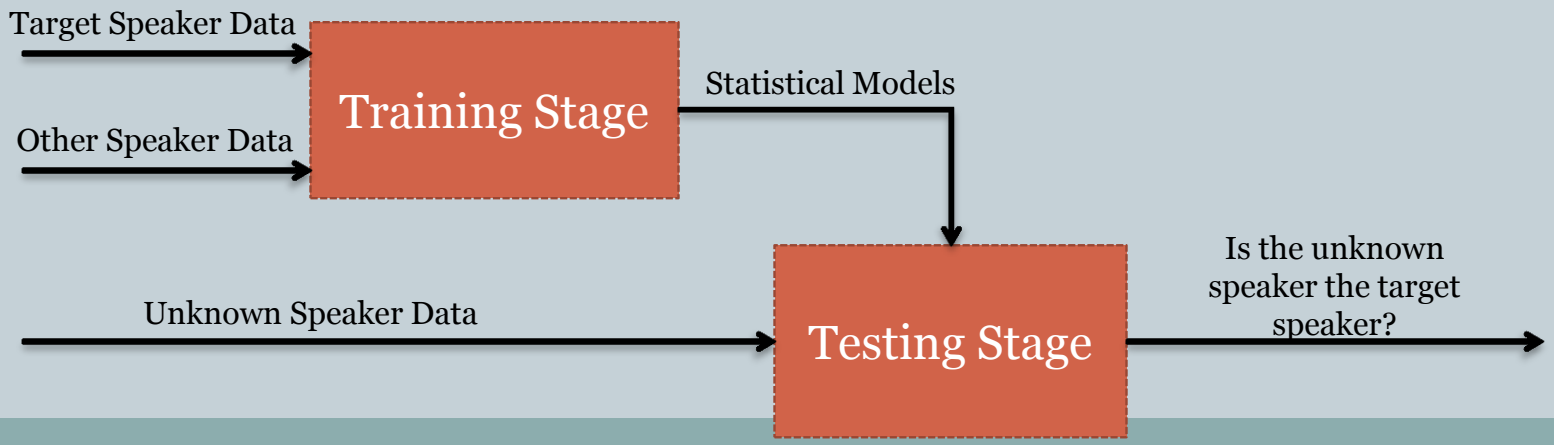
Outline

- Review of Speaker Recognition
  - Overview
  - Training Stage
  - Testing Stage
- Nuisance Attribute Projection
  - What is NAP?
  - Background
  - Estimating Channel Subspace Matrix
  - f-NAP

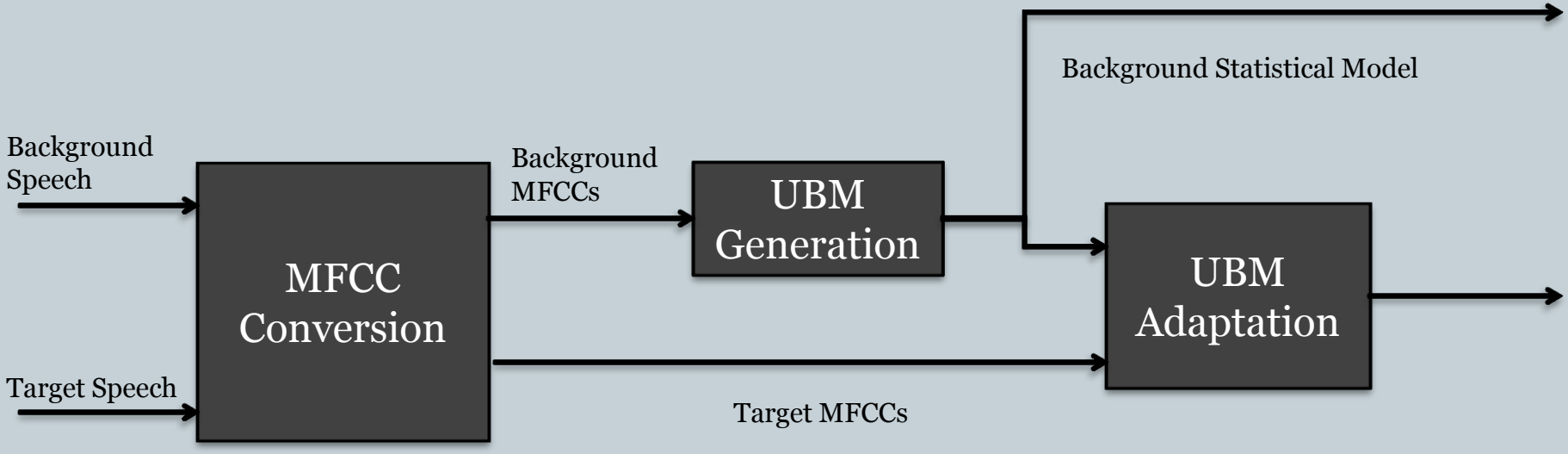
A Review of Speaker Recognition

Overview

- Training Stage
  - A background model is formed
  - The model is adapted to a target speaker
- Testing Stage
  - The background model and adapted model are used to determine if an unknown speaker is a target speaker

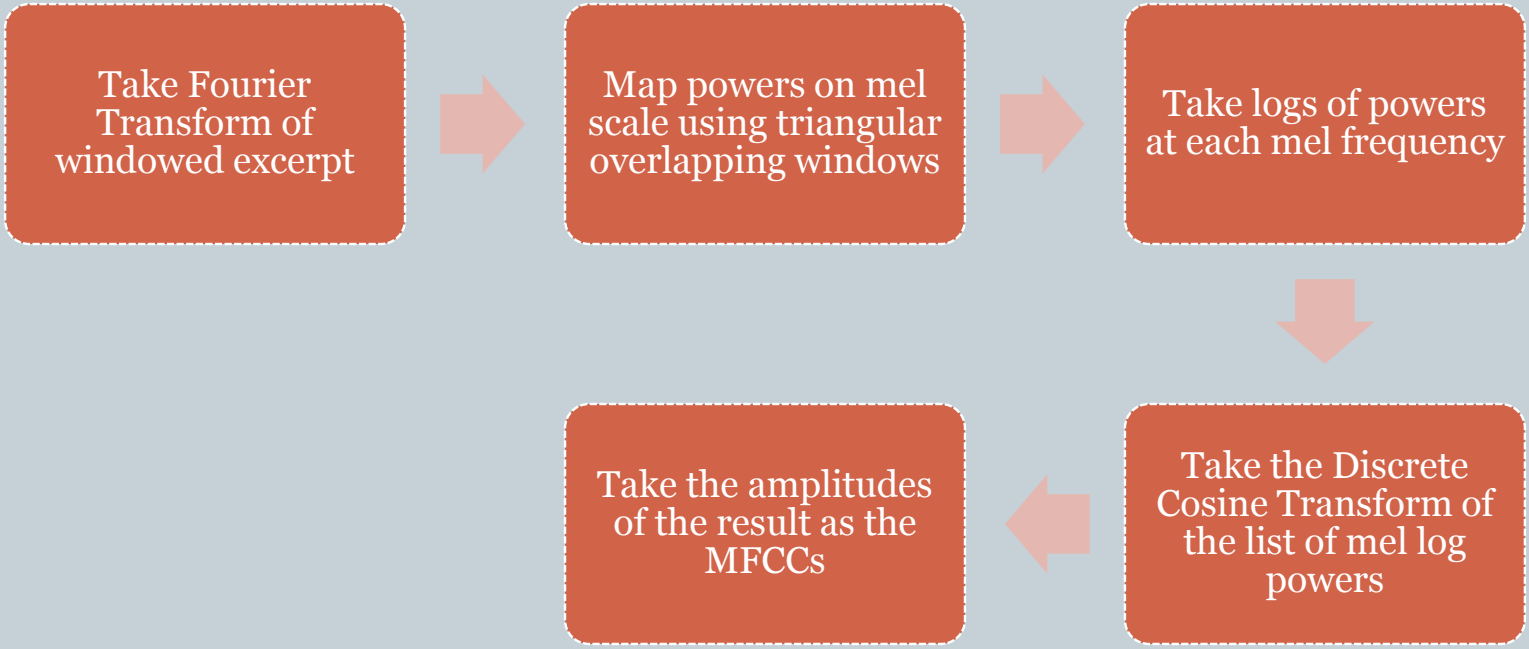


Training Stage



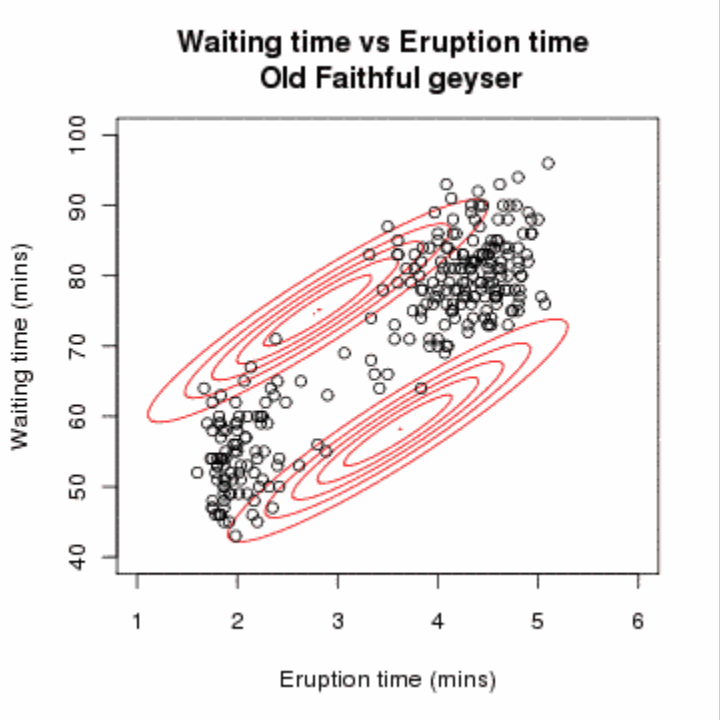
MFCC Conversion

- MFCCs are coefficients that represent the power of a sound over a short window
- Nonlinear scaling
- The Process:



UBM Generation

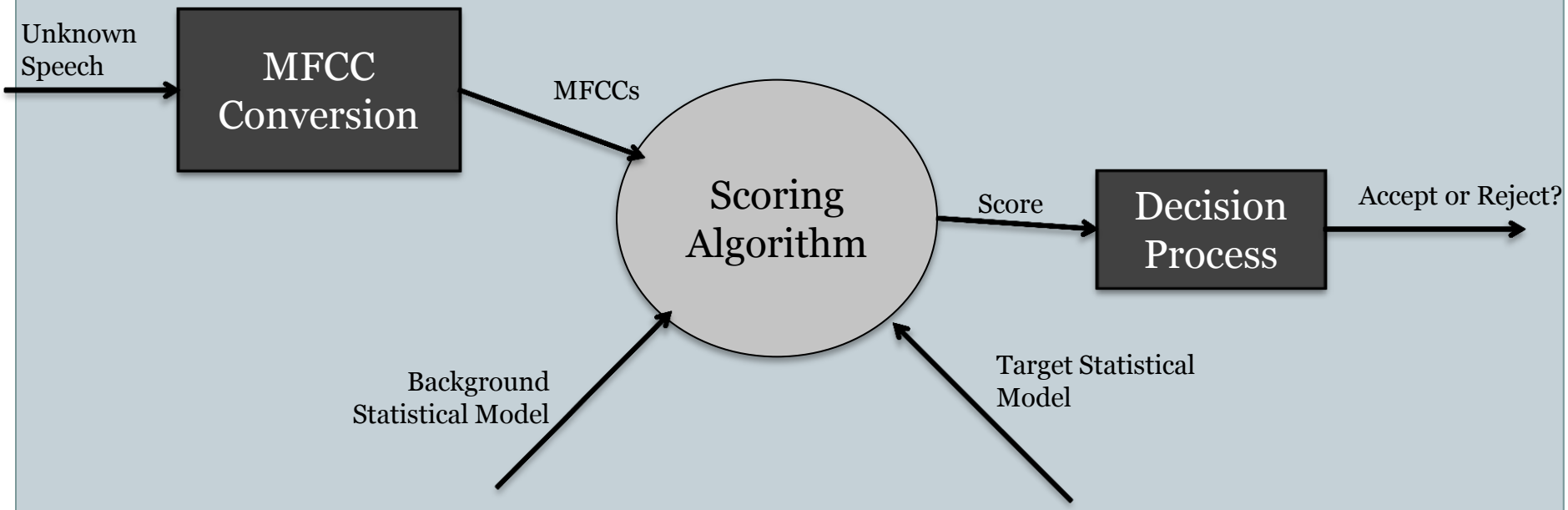
- One of the techniques to perform speaker verification is through the use of Gaussian Mixture Models
- The Model is generated using data from many speakers
- Perform the EM algorithm



UBM Adaptation

- Typically the target speaker does not provide a lot of available speech.
  - Full models trained on this speech may not be complete
- Adapting the background model instead of creating a completely new model can provide better results
- Usually, without a lot of speech, only the means are adapted.

Testing Stage



Scoring

- When unknown speech comes in, a log likelihood test is performed using the background model and the adapted model.
- If it's above a certain threshold, the person is determined to be the target. Else, it's not.

Nuisance Attribute Projection

What is Nuisance Attribute Projection?

- A method to filter out nuisance attributes in classifiers.
- Nuisance attributes are qualities that affect the appearance of an observation without actually being affiliated with a specific person
- This method tries to make all given speech appear to be performed using the same conditions

Example

- Type of microphone used
- Different types of microphones produce different variations on the same person's speech.
- Depending on the other speakers and how vastly different certain microphones are, target speakers may be confused for imposters and vice versa

Why we Need it

- Since the means are the only component being adapted in an adapted GMM system, one could represent them as the following:

$$\boldsymbol{\mu}_k(s) = \boldsymbol{\mu}_k + \boldsymbol{d}_k \boldsymbol{z}_k(s)$$

- Where $\boldsymbol{\mu}_k$ is the UBM mean vector. $\boldsymbol{d}_k$ a D x D diagonal matrix and $\boldsymbol{z}_k(s)$ is a speaker dependent random vector distributed according to a standard normal distribution.
- $\boldsymbol{\mu}_k(s)$ is a random variable distributed according to a normal distribution with mean $\boldsymbol{\mu}_k$ and diagonal covariance $\boldsymbol{d}_k^2$

Why We Need It

- Given some training utterance: $\boldsymbol{X}(s) = \{\boldsymbol{x}_1(s), \ldots \boldsymbol{x}_T(s)\}$
- Represent the means (determined via MAP) as

$$E[\boldsymbol{\mu}_k(s)] = \boldsymbol{\mu}_k + \left(\boldsymbol{I} + \boldsymbol{d}_k^2 \boldsymbol{\Sigma}_k^{-1} N_k\right)^{-1} \boldsymbol{d}_k^2 \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{F}_k - N_k \boldsymbol{\mu}_k)$$

- Where

$$N_k = \sum_{t=1}^{T} \gamma_{k,t}$$

$$\boldsymbol{F}_k = \sum_{t=1}^{T} \gamma_{k,t} \boldsymbol{x}_t(s)$$

- Where $\gamma_{k,t}$ is the responsibility of the mixture coefficient k given observation t
- The matrix $\boldsymbol{d}_k$ can usually be assumed to be related to the covariance matrix in the form of $\boldsymbol{d}_k^2 = \tau^{-1} \boldsymbol{\Sigma}_k$ where τ is a fixed relevance parameter typically chosen as a number between 8 and 16

Why We Need It

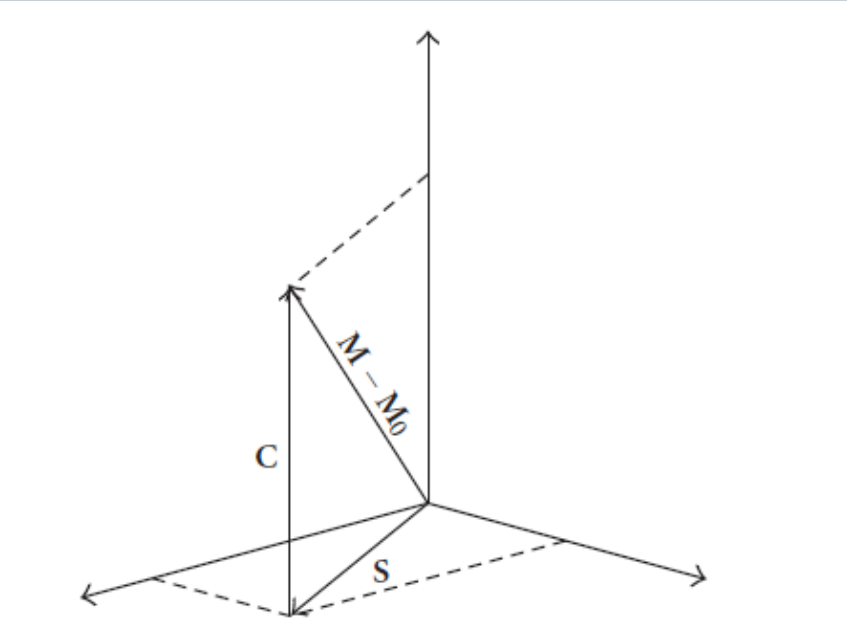
- Ideally the estimate should be the same no matter what training utterance is used.
- However, different channels cause speakers to sound a little different even from themselves.
- Using NAP it is possible to compensate for the channel effects if one makes some assumptions about the channel.

The Two Dependences

- The difference between the recording, one dependent on both the speaker and the channel, and the UBM can be represented by the following supervector

$$\boldsymbol{M - M_0 = S + C}$$

- We want to separate the supervector $\boldsymbol{S}$ (the speaker dependent supervector) from $\boldsymbol{C}$ (the channel dependent supervector).



The Channel Effect

- The channel effect can be considered unwanted noise.
- The noise is convolutionary mixed with the speech. In frequency domain, it is additive
- The channel effect and the speech can be though of as mixtures of gaussians with means N and M respectively
- Together they're summed together.
- Total number of gaussian mixtures is (N*M)
- Only a few of them will be represented due to finite length.

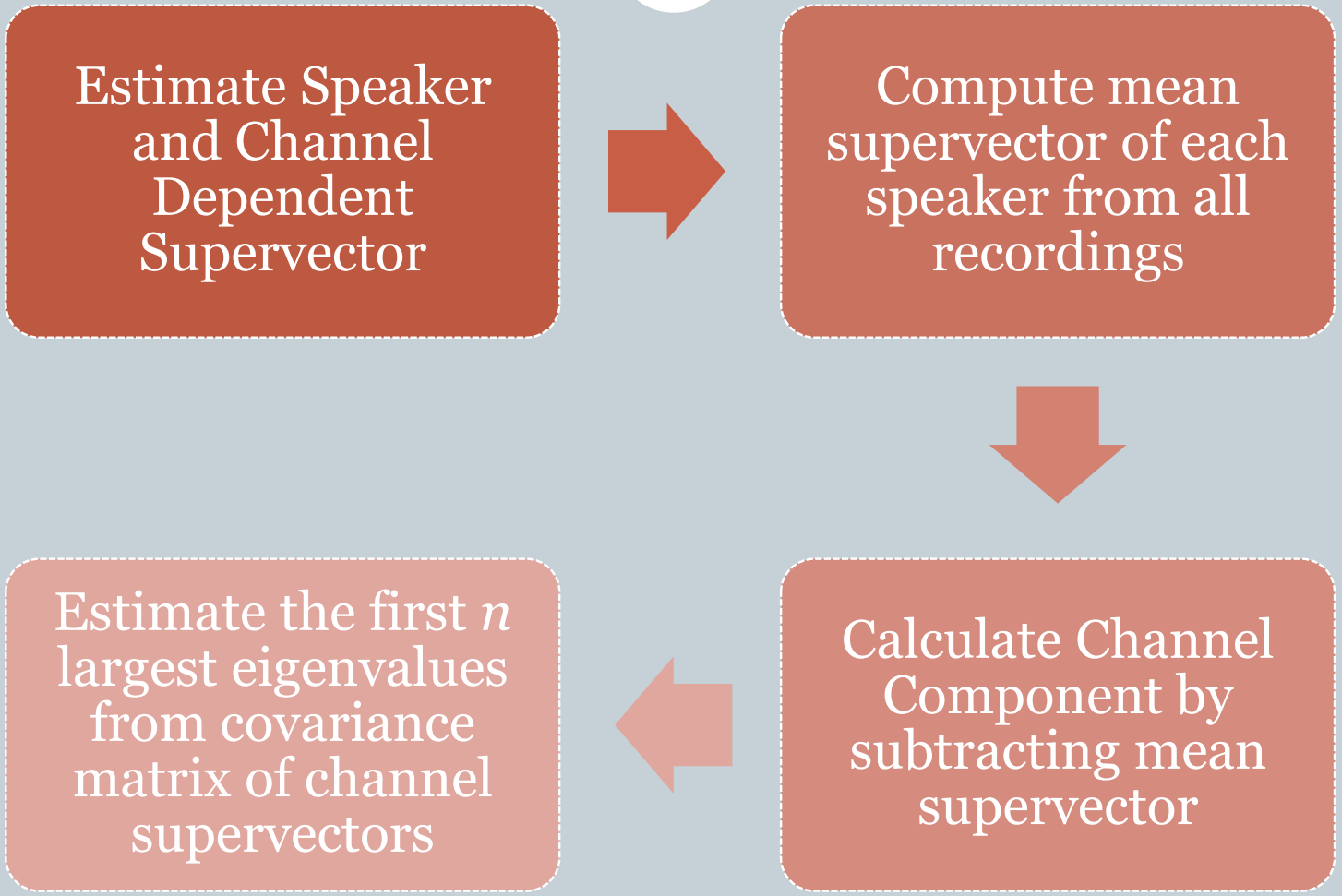
Channel Dependent Portion

- The channel dependent portion is much lower dimensional (spanning the subspace $\boldsymbol{U}$) than the speaker dependent portion
  - If it was the same order, the channel could transform one person to sound like another.
- One can then represent the Channel Dependent portion of the speech segment ($\boldsymbol{C}$) as the following

$$\boldsymbol{C} = \boldsymbol{U}\boldsymbol{U}^{*}(\boldsymbol{M} - \boldsymbol{M}_0)$$

- The Subspace $\boldsymbol{U}$ has to be estimated from the data.
- Requires sufficiently large number of recordings from target data

Estimating Channel Subspace Matrix



Estimating Eigenvalues

- To do the final step, Principal Component Analysis is used
- PCA can be used to find the eigenvectors and eigenvalues of a covariance matrix given a set of values

PCA Procedure

- Put data into an M x N matrix (M dimensional data by N data points)
- Calculate empirical mean
- Subtract mean from data

Procedure

- Find Covariance Matrix

$$C = \frac{1}{N}\sum(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^{\mathrm{T}}$$

- Find Eigenvalues and Eigenvectors of Covariance Matrix

$$\boldsymbol{V^{-1}CV = D}$$

  - Where V is the matrix of eigenvalues and D is a diagonal matrix of the eigenvalues of C

Forming Channel Subspace Matrix

- Take the components of $\boldsymbol{V}$ corresponding to the M largest eigenvalues of $\boldsymbol{C}$ to form $\boldsymbol{U}$
- However, the dimension of the covariance matrix is generally too high to perform normal PCA
- Variants including a probabilistic variant can be performed instead.

Problems with GMM Systems

- NAP is usually performed on SVM systems.
- GMM approaches have not been used because it is asymmetric in the sense that only the training utterances is used to estimate the speaker model and thus projected to a supervector
- SVMs are symmetric in the sense that both the training and the testing utterances are projected to supervectors

fNAP

A Solution to this?

- NAP has been extended into the feature domain.
- This allows GMMs to be used with NAP.
- In this situation, the feature vectors are altered and these altered feature vectors are used in the training and testing

Feature Mapping

- First calculate the channel component $\boldsymbol{C}$
- Then transform each feature vector $\boldsymbol{x}$ with the following equation

$$\hat{\boldsymbol{x}}_t = \boldsymbol{x}_t - \sum_{k=1}^{K} \gamma_{k,t} \boldsymbol{c}_k$$

- Where $\gamma_{k,t}$ is the responsibility of the observation $\boldsymbol{x}_t$ by the kth mixture component and $\boldsymbol{c}_k$ is part of the supervector $\boldsymbol{C}$ corresponding to the kth mixture component
- From here we can then perform training and testing

Conclusion

- Microphone types can affect the quality of an audio recording
  - Different microphones will make someone sound slightly different
- NAP can be used to lessen the effect of these different microphones
- However, NAP by itself cannot be used in GMM based systems. Instead an alternate procedure, fNAP, is used instead

Support Vector Machines

By David Cinciruk
ASPITRG Group Meeting
July 18, 2012

Outline

- Review
  - Speaker Recognition
  - NAP
- Introduction to SVMs
- SVM based Speaker Recognition
- Conclusion
- Next Time…

Review

Review of Speaker Recognition

- Problem of having a computer determining if a person is a target speaker or not
- Process using GMMs involves:
  - Training:
    - A world model for a generic speaker from many different speakers
    - A target model for a given target based off of the data given
  - Testing
    - Performing log-likelihood tests to determine if the person is a target or not

Review of NAP – Why We Need It

- Speaker models are typically trained on very little data
- Effects from the type of microphone can severely affect the training.
- NAP works to remove these effects

Review of NAP – Channel Dependent Supervector

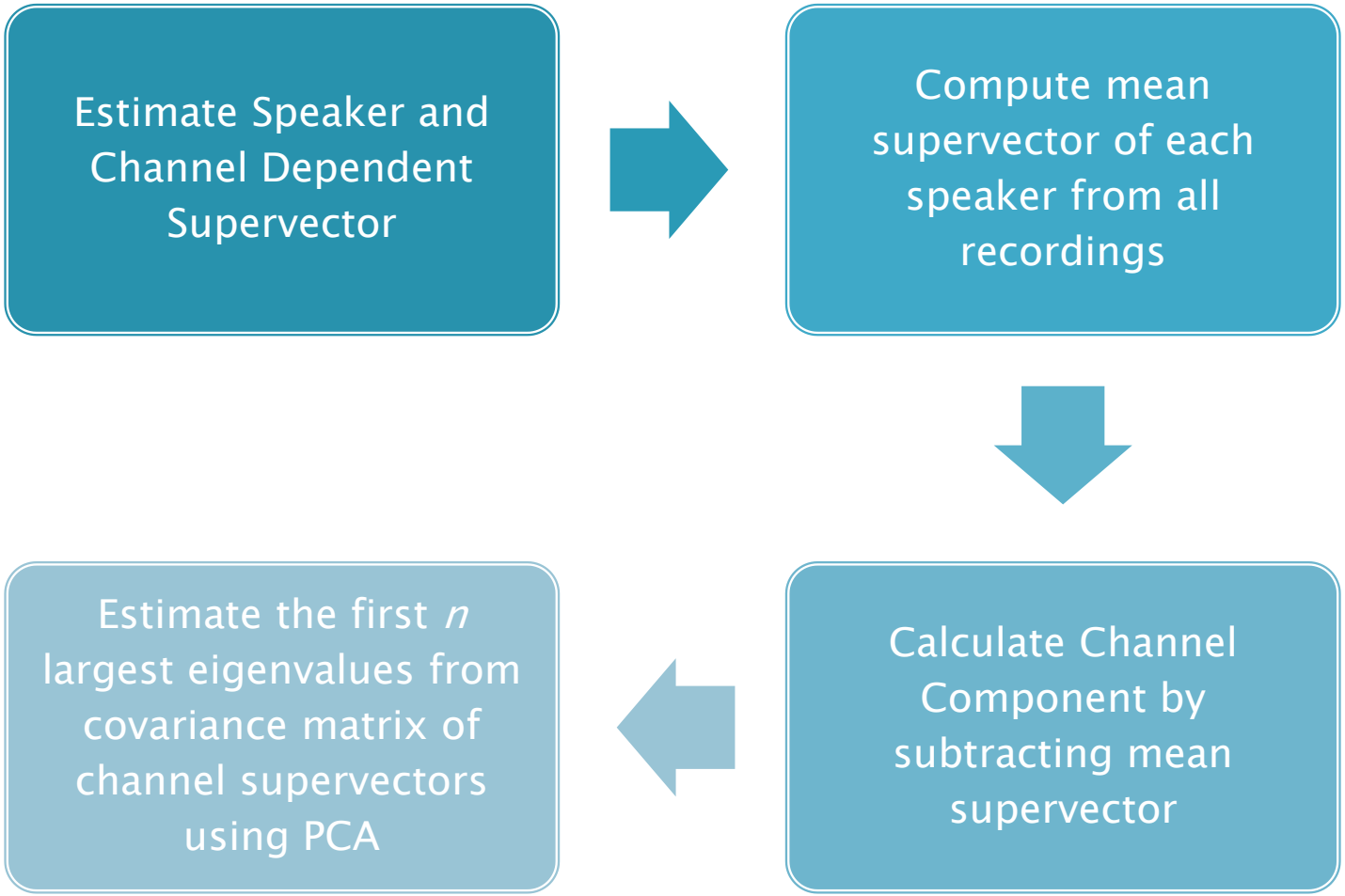
- Difference between the speaker and the UBM can be represented by the following supervector

$$\boldsymbol{M} - \boldsymbol{M}_0 = \boldsymbol{S} + \boldsymbol{C}$$

- Channel dependent portion is much lower dimensional (spanning the subspace $\boldsymbol{U}$) than the speaker dependent portion

$$\boldsymbol{C} = \boldsymbol{U}\boldsymbol{U}^*(\boldsymbol{M} - \boldsymbol{M}_0)$$

Review of NAP – Estimating Channel Subspace Matrix



Review of NAP – How to Implement

- Typically implemented in an SVM based system
- For GMM based systems, a variant called fNAP is used instead

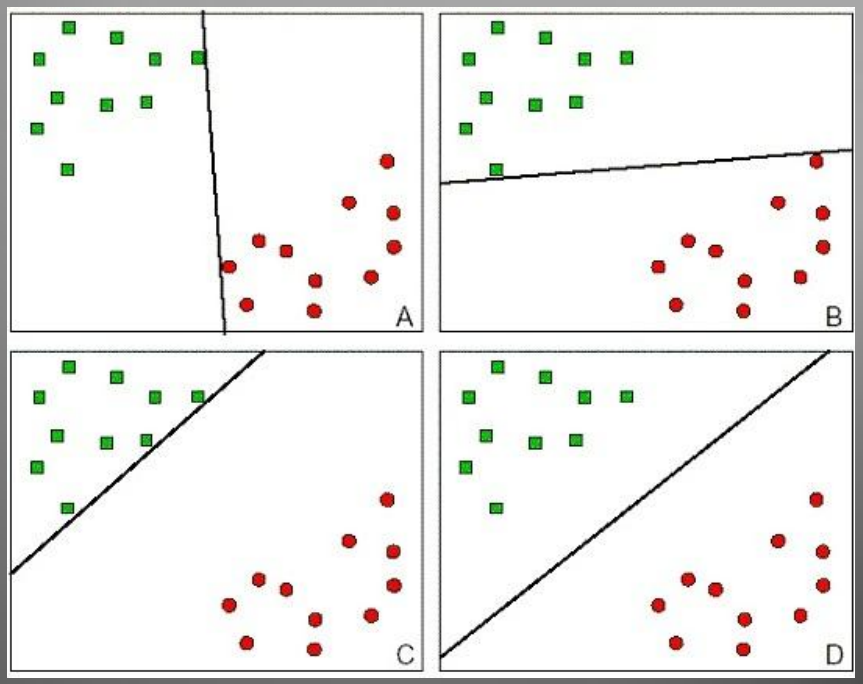
Introduction to SVMs

Support Vector Machines

- Supervised learning technique that attempts to classify data into one of two categories
- New data is then classified into one of two categories depending on how far it is from a decision boundary
- Decision boundary is linear
  - By applying a kernel trick, it is possible to express nonlinear data is linear

Decision Boundary

- Want to maximize the distance between the data and a decision boundary
- New data is more likely to be classified in the right category then
- Can formulate as an optimization problem



Generating a Decision Boundary

- Consider a set of training data D, a set of points given by the following

$$D = \{(\boldsymbol{x}_i, y_i) | \boldsymbol{x}_i \in \mathbb{R}^p, y_i \in \{-1,1\}\}_{i=1}^{n}$$

- Definition of a line/hyperplane:

$$\mathbf{w} \cdot \mathbf{x} - b = 0$$

- **w** is the normal vector to the hyperplane and $\frac{b}{\|\mathbf{w}\|}$ is the offset of the hyperplane from the origin along direction **w**

Generating a Decision Boundary

- Select two parallel hyperplanes so that they separate the data and don't leave any points between them.

$$\mathbf{w} \cdot \mathbf{x} - b = 1$$

$$\mathbf{w} \cdot \mathbf{x} - b = -1$$

- The distance between the two can be represented as $\frac{2}{\|\mathbf{w}\|}$
- We want to maximize the distance between the two hyperplanes so we want to minimize $\|\mathbf{w}\|$

Generating a Decision Boundary

- All data points $\boldsymbol{x}_i$ will either lie above or below this decision boundary.
- Can represent this by the following constraints
- $\mathbf{w} \cdot \boldsymbol{x}_i - b \geq 1$ for $\boldsymbol{x}_i$ from the first class
- $\mathbf{w} \cdot \boldsymbol{x}_i - b \leq -1$ for $\boldsymbol{x}_i$ from the second class
- Given y_i these constraints can be represented as the following:

$$y_i(\mathbf{w} \cdot \boldsymbol{x}_i - b) \geq 1$$

Generating a Decision Boundary

- The following quadratic program can be created

$$\min \frac{1}{2} \|\mathbf{w}\|^2$$

- Subject to

$$y_i(\mathbf{w} \cdot \boldsymbol{x}_i - b) \geq 1$$

Solving the Optimization Problem

- Setting up Lagrange multipliers we can then find the optimal solution via the following

$$\min_{\mathbf{w},b} \max_{\boldsymbol{\alpha} \geq 0} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i [y_i (\mathbf{w} \cdot \boldsymbol{x}_i - b) - 1] \right\}$$

- Using the KKT conditions we can then express the solution as the following:

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

$$b = \frac{1}{N_{SV}} \sum_{i=1}^{N} \mathbf{w} \cdot \boldsymbol{x}_i - y_i$$

Dual Form

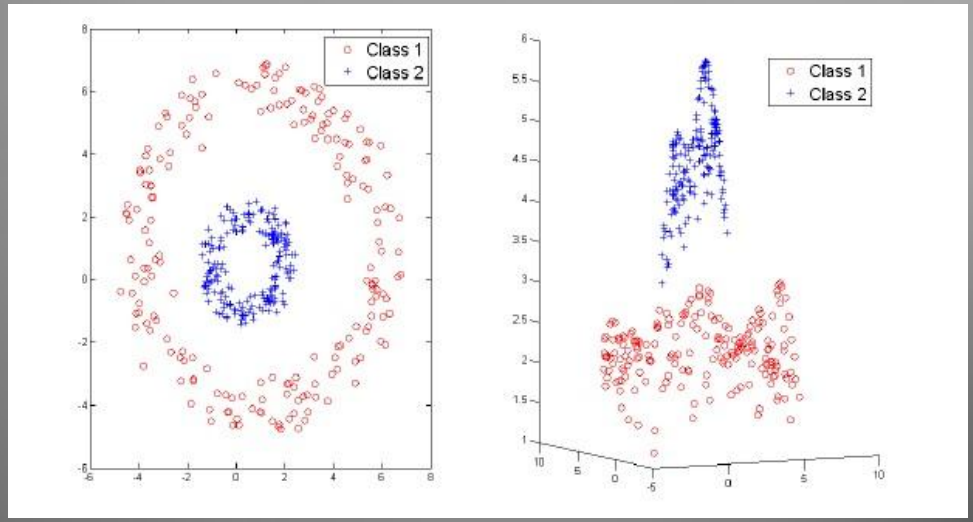
- Able to write this problem in a dual form.
- Task becomes dependent on just support vectors instead
- Using the definition for **w** as defined previously, we can write the problem as:

$$\max_{\alpha_i} \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x}_j) \right\}$$

- Subject to $\alpha_i \geq 0$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$
- Where $\mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a kernel function

Kernel Trick

- Nonlinear data can be projected onto higher dimensional space to make it appear linear.
- SVMs can be performed on this now linear data

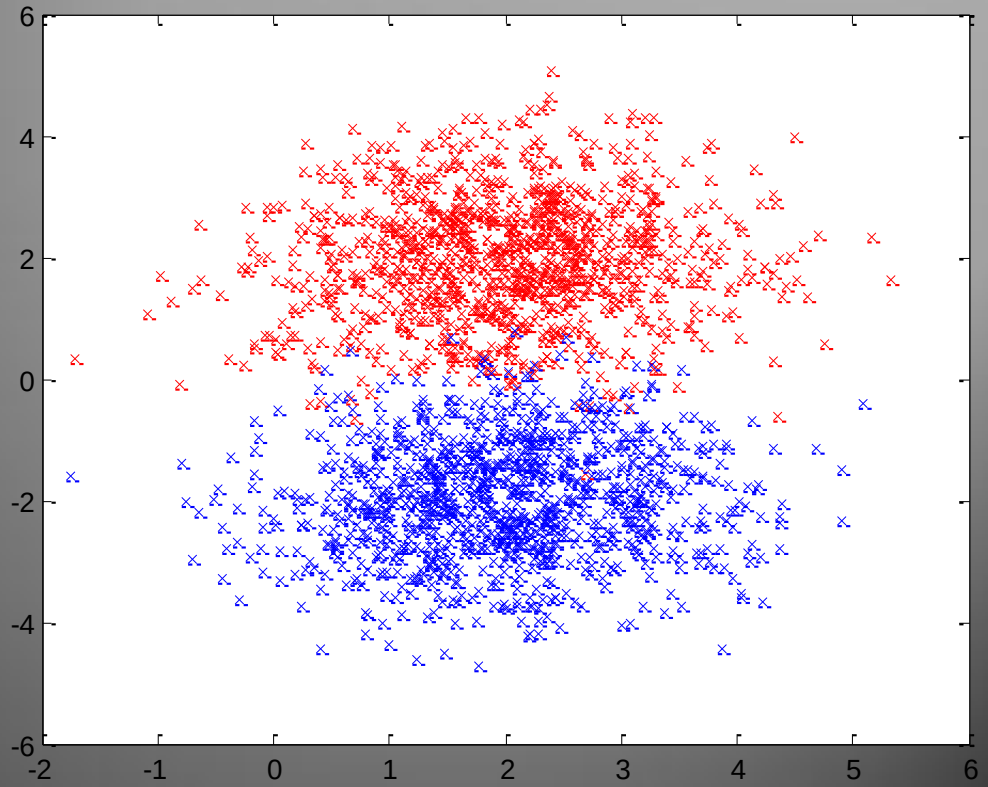


Example Kernels

- Some example kernels include
- Polynomial: $\mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i \cdot \boldsymbol{x}_j)^d$
- Gaussian Radial Basis function: $\mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}$

Soft Margin

- The previous definition can only separate when there's a clear difference between the two classes
- A softer margin is needed to separate data where this separation doesn't exist



Soft Margin – Primal Form

- Method introduces a slack variable ξ_i which measures the degree of misclassification of data point $\boldsymbol{x}_i$
- Constraint becomes

$$y_i(\mathbf{w} \cdot \boldsymbol{x}_i - b) \geq 1 - \xi_i$$

- With Lagrange Multipliers, the program becomes

$$\min_{\mathbf{w},\boldsymbol{\xi},b} \max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \left\{ \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i [y_i(\mathbf{w} \cdot \boldsymbol{x}_i - b) - 1 + \xi_i\ ] - \sum_{i=1}^{n} \beta_i \xi_i \right\}$$

Soft Margin – Dual Form

- Once again a dual version of this problem can be found

$$\max_{\boldsymbol{\alpha}} \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \right\}$$

- Subject to

$$0 \leq \alpha_i \leq C$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

SVMs and Speaker Recognition

General Structure

- First need to generate a kernel function.
- A kernel based on the sequences of MFCCs uttered is of value
- One way is to generate a model using one set of utterances and then scoring it against another.
- This produces a measure of similarity of the two utterances

Generalized Linear Discriminant Scoring

- Represent the speaker recognition problem as a two class problem.
  - If ω is a random variable representing the hypothesis, $\omega = 1$ represents the target and $\omega = 0$ represents an imposter
- Score is calculated from a sequence of observations from speech
- Scoring is of the form $g(y) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{b}(\boldsymbol{y})$
  - $\boldsymbol{w}$ is the vector of classifier parameters, the model
  - $\boldsymbol{b}$ is a mapping of the supervector of utterances down to a N-dimensional space

Generalized Linear Discriminant Scoring

- Commonly used linear discriminants are polynomials and radial basis functions
  - Higher ordered ones aren't used in order to find a closed form expression for training
- Assuming independence of the observations we get:

$$p(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n|\omega) = \prod_{i=1}^{n} p(\boldsymbol{y}_i|\omega) = \prod_{i=1}^{n} \frac{p(\omega|\boldsymbol{y}_i)p(\boldsymbol{y}_i)}{p(\omega)}$$

Generalized Linear Discriminant Scoring

- Take the log of the probability (and discarding $p(\boldsymbol{y}_i)$ to get the discriminant function

$$d'(\boldsymbol{y}|\omega) = \sum_{i=1}^{n} \log\left(\frac{p(\omega|\boldsymbol{y}_i)}{p(\omega)}\right)$$

- Using the first order Taylor Series approximation, we get

$$d'(\boldsymbol{y}|\omega) = \frac{1}{n}\sum_{i=1}^{n} \frac{p(\omega|\boldsymbol{y}_i)}{p(\omega)}$$

- The -1 in the term has been dropped and the function was normalized by the number of frames. This does not affect classification decision

Generalized Linear Discriminant Scoring

- Assume we have $g(\boldsymbol{y}) \approx p(\omega = 1|\boldsymbol{y})$
- Substituting in we have:

$$d(\boldsymbol{y}|\omega = 1) = \frac{1}{n}\sum_{i=1}^{n}\frac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{b}(\boldsymbol{y}_i)}{p(\omega = 1)} = \frac{1}{np(\omega = 1)}\boldsymbol{w}^{\mathrm{T}}\sum_{i=1}^{n}\boldsymbol{b}(\boldsymbol{y}_i)$$

$$= \frac{1}{p(\omega = 1)}\boldsymbol{w}^{\mathrm{T}}\overline{\boldsymbol{b}}_y$$

- Where we defined the mapping $\boldsymbol{y} \rightarrow \overline{\boldsymbol{b}}_y$ as

$$\boldsymbol{y} \rightarrow \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{b}(\boldsymbol{y}_i)$$

Monomials

- A monomial is a polynomial of the form $x_{i1}x_{i2} \dots x_{ik}$
- Where k is less than or equal to the polynomial form
- Input vector: $\boldsymbol{x} = [x_1 \quad x_2 \quad \dots \quad x_m]^{\mathrm{T}}$
- The vector $\boldsymbol{b}(\boldsymbol{x})$ is the vector of monomials of the input vector up to and including degree k

Generalized Linear Classifier Training

- Let $\boldsymbol{w}$ be the desired target model. We can represent it as

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmin}}\, \mathbf{E}\left[\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{b}(\boldsymbol{x}) - \omega\right)^2\right]$$

- This criterion can be approximated using the training set as

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmin}}\, \mathbf{E}\left[\sum_{i=1}^{N_{tgt}} \left|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{b}(\boldsymbol{x}_i) - 1\right|^2 + \sum_{i=1}^{N_{non}} \left|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{b}(\boldsymbol{z}_i)\right|^2\right]$$

- Target training data is $\boldsymbol{x}_i$ and non-target data is $\boldsymbol{z}_i$

Generalized Linear Classifier Training

- Training method can be written in matrix form
- Define $\boldsymbol{M}_{tgt}$ as the matrix whose rows are the expansion of the target's data

$$\boldsymbol{M}_{tgt} = \left[\boldsymbol{b}(\boldsymbol{x}_1) \quad \boldsymbol{b}(\boldsymbol{x}_2) \quad \cdots \quad \boldsymbol{b}(\boldsymbol{x}_{N_{tgt}})\right]^{\mathrm{T}}$$

- You can define a similar matrix for nontarget data $\boldsymbol{M}_{non}$
- Then define: $\boldsymbol{M} = \left[\boldsymbol{M}_{tgt}^{\mathrm{T}} \quad \boldsymbol{M}_{non}^{\mathrm{T}}\right]^{\mathrm{T}}$
- The problem then becomes:

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmin}} \|\boldsymbol{M}\boldsymbol{w} - \boldsymbol{o}\|_2$$

- Where o is the ideal output (consisting of N_{tgt} ones and N_{non} zeros)

Generalized Linear Classifier Training

- Using the method of normal equations, the equation can be solved as

$$\boldsymbol{M}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{w} = \boldsymbol{M}^{\mathrm{T}}\boldsymbol{o}$$

- Rearranging it, it becomes

$$\left(\boldsymbol{M}^{\mathrm{T}}\boldsymbol{M}\right)\boldsymbol{w} = \boldsymbol{M}_{tgt}^{\mathrm{T}}\mathbf{1} + \boldsymbol{M}_{non}^{\mathrm{T}} = \boldsymbol{M}_{tgt}^{T}$$

- If we define $\boldsymbol{R} = \boldsymbol{M}^{\mathrm{T}}\boldsymbol{M}$ the solution becomes

$$\boldsymbol{w} = \boldsymbol{R}^{-1}\boldsymbol{M}_{tgt}^{\mathrm{T}}\mathbf{1}$$

Generalized Linear Discriminant Sequence Kernels

- Combine previous equations to obtain the score

$$S = \frac{1}{p(\omega = 1)} \overline{\boldsymbol{b}}_y^{\mathrm{T}} \boldsymbol{w} = \frac{1}{p(\omega = 1)} \overline{\boldsymbol{b}}_y^{\mathrm{T}} \boldsymbol{R}^{-1} \boldsymbol{M}_{tgt}^{\mathrm{T}} \mathbf{1}$$

- We can now define $p(\omega = 1) = \frac{N_{tgt}}{N_{non} + N_{tgt}}$ so the score becomes

$$S = \overline{\boldsymbol{b}}_y^{\mathrm{T}} \overline{\boldsymbol{R}}^{-1} \overline{\boldsymbol{b}}_x^{\mathrm{T}}$$

- Where

$$\overline{\boldsymbol{b}}_x = \frac{1}{N_{tgt}} \boldsymbol{M}_{tgt}^{\mathrm{T}} \mathbf{1}$$

$$\overline{\boldsymbol{R}} = \frac{1}{N_{non} + N_{tgt}} \boldsymbol{R}$$

Generalized Linear Discriminant Sequence Kernels

- This scoring method is the basis of the sequence kernel
- Map a speech sequence $\boldsymbol{x} \rightarrow \overline{\boldsymbol{b}}_x$ and another speech sequence $\boldsymbol{y} \rightarrow \overline{\boldsymbol{b}}_y$
- The kernel becomes

$$K_{GLDS}(\boldsymbol{x}, \boldsymbol{y}) = \overline{\boldsymbol{b}}_x \boldsymbol{R}^{-1} \overline{\boldsymbol{b}}_y$$

- The function is not symmetric so it's not yet a kernel

Symmetrizing the Kernel

- A few approximations about $\bar{\boldsymbol{R}}$ can be made
- Nontarget data typically dominates the calculation of $\bar{\boldsymbol{R}}$
- We can approximate $\bar{\boldsymbol{R}} \approx \frac{1}{N_{non}} \boldsymbol{R}_{non}$
- We don't need additional target data to approximate the average $\bar{\boldsymbol{R}}$ if we have a large nontarget set.
- The kernel is now symmetric with respect to the roles of the two sequences. We can use either for training or testing.

Another Approximation

- Another approximation would be to assume $\bar{\boldsymbol{R}}$ is diagonal – this decreases calculations
- If $\bar{\boldsymbol{R}}$ is a full correlation matrix, $\bar{\boldsymbol{R}} = \boldsymbol{U}^{\mathrm{T}}\boldsymbol{U}$ via the Cholesky decomposition
- The kernel can then be expressed as:

$$K_{GLDS}(\boldsymbol{x}, \boldsymbol{y}) = \left(\boldsymbol{U}\bar{\boldsymbol{b}}_x\right)^{\mathrm{T}}\left(\boldsymbol{U}\bar{\boldsymbol{b}}_y\right)$$

Score Simplification

- Simplify the scoring with the GLDS kernel
- Suppose $f(\{\boldsymbol{x}_i\})$ is the output of the SVM

$$f(\{\boldsymbol{x}_i\}) = \sum_{i=1}^{N} \alpha_i t_i \, \overline{\boldsymbol{b}}_i^{\mathrm{T}} \boldsymbol{R}^{-1} \overline{\boldsymbol{b}}_x + d$$

- Where $\overline{\boldsymbol{b}}_i$ are the support vectors
- Simplify it so

$$f(\{\boldsymbol{x}_i\}) = \left(\sum_{i=1}^{N} \alpha_i t_i \, \boldsymbol{R}^{-1} \overline{\boldsymbol{b}}_i + \boldsymbol{d} \right)^{\mathrm{T}} \overline{\boldsymbol{b}}_x$$

- Where $\boldsymbol{d} = [d \quad 0 \quad 0 \quad 0]^{\mathrm{T}}$

Final Model

- We can collapse all the support vectors down into a single model:

$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i t_i \boldsymbol{R}^{-1} \overline{\boldsymbol{b}}_i + \boldsymbol{d}$$

- Each target score is then simply an inner product $\boldsymbol{w}_{tgt}^{\mathrm{T}} \overline{\boldsymbol{b}}_x$

Algorithm for Creating a Nontarget Model

Table 1
Creating a nontarget background

1) Given: N_{utt} nontarget utterances
2) $N_{\text{tot}} = 0$
3) $\mathbf{r} = \mathbf{0}$
4) For $i = 1$ to N_{utt}
   5) Let $\{\mathbf{z}_i\}$, $i = 1, \ldots, N_z$, be the features extracted from the ith nontarget utterance
   6) Calculate and store $\bar{\mathbf{b}}_z^i = (1/N_z) \sum_{i=1}^{N_z} \mathbf{b}(\mathbf{z}_i)$
   7) $\mathbf{r} = \mathbf{r} + \sum_{i=1}^{N_z} \mathbf{b}(\mathbf{z}_i) .* \mathbf{b}(\mathbf{z}_i)$
   8) $N_{\text{tot}} = N_{\text{tot}} + N_z$
9) Next i
10) Let $\mathbf{r} = (1/N_{\text{tot}})\mathbf{r}$
11) Let $\mathbf{r}_{\text{sqrt}} = 1./\sqrt{\mathbf{r}}$
12) For all $i = 1, \ldots, N_{\text{utt}}$, replace $\bar{\mathbf{b}}_z^i = \mathbf{r}_{\text{sqrt}} .* \bar{\mathbf{b}}_z^i$.
13) The set of vectors $\{\bar{\mathbf{b}}_z^i\}$ is the nontarget background

Algorithm for Creating a Target Model

Table 2
Creating a target model

1) Given: N_{tgt} target utterances
2) For $i = 1$ to N_{tgt}
   3) Let $\{\mathbf{x}_i\}$, $i = 1, \ldots, N_x$, be the features extracted from the ith target utterance
   4) $\bar{\mathbf{b}}_x^i = (1/N_x) \sum_{i=1}^{N_x} \mathbf{b}(\mathbf{x}_i)$
   5) $\bar{\mathbf{b}}_x^i = \mathbf{r}_{\text{sqrt}} .* \bar{\mathbf{b}}_x^i$ where $\mathbf{r}_{\text{sqrt}}$ is from the background training algorithm in Table 1
6) Next i
7) Train an SVM using: a linear kernel ($K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^t\mathbf{y}$), ideal outputs of 1 for $\{\bar{\mathbf{b}}_x^i\}$, and ideal outputs of -1 for $\{\bar{\mathbf{b}}_z^i\}$ (computed in Table 1). For the trained SVM, call the resulting weights, α_i, the support vectors, $\mathbf{b}_i$, and the constant, d.
8) Compute the target model as $\mathbf{w} = \mathbf{r}_{\text{sqrt}} .* \left(\sum_{i=1}^{l} \alpha_i t_i \bar{\mathbf{b}}_i \right) + \mathbf{d}$ where $\mathbf{d} = \begin{bmatrix} d & 0 & \ldots & 0 \end{bmatrix}^t$, and t_i is the ideal output for the ith support vector.

Fusion with GMM-based Systems

- SVMs by themselves don't offer an improvement over using GMMs
- However, by fusing them together, the results can improve by a few percentage

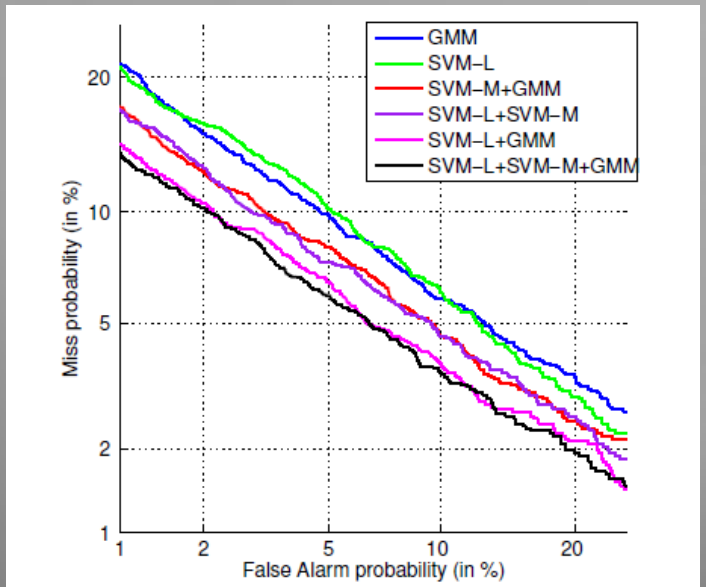


Fig. 5. NIST 2003 1sp limited data fusion results

Table 3
Comparison of EER and minDCF for different systems on the 2003 NIST SRE 1sp limited data evaluation

System	EER	minDCF
GMM	7.47 %	0.0306
SVM-L	7.72 %	0.0303
SVM-M	9.57 %	0.0374
SVM-M+GMM	6.74 %	0.0266
SVM-L+SVM-M	6.46 %	0.0262
SVM-L+GMM	5.73 %	0.0237
SVM-L+SVM-M+GMM	5.55 %	0.0230

Conclusion

Conclusion

- Support Vector Machines can be used in speaker recognition systems as well
- SVMs require generating a decision boundary for which to determine which class certain data belongs to.

Next Time

»

Next Time

- NAP using SVMs
- fNAP using GMMs
- Probabilistic PCA with respect to NAP/fNAP

By David Cinciruk
ASPITRG Group Meeting
July 21, 2012

SPEAKER RECOGNITION IN SVMS

OUTLINE

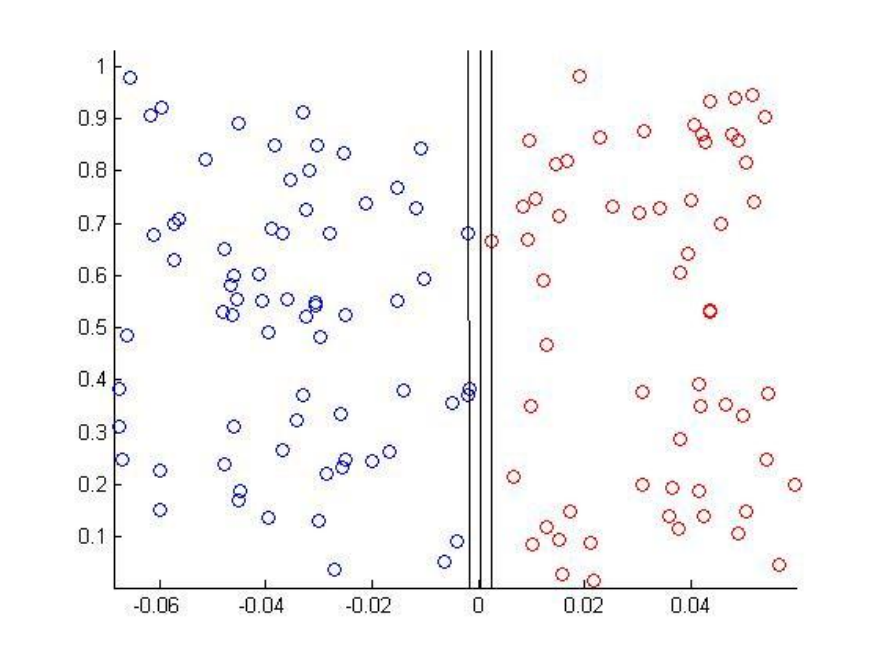
- Review of Speaker Recognition
- Review of SVMs
- Multiple Implementations of SVMs in Speaker Recognition

REVIEW

REVIEW OF SPEAKER RECOGNITION

- Process of having a computer determine if a person is a target speaker or not
- Can be performed using GMM or SVM based systems
- People are interested in determining error rates based on a specific threshold and on minimizing equal error rates

REVIEW OF SVMS



- SVMs are a tool used to discriminate data points into two different classes
- It splits the two classes linearly.
- Nonlinearly separable data can be converted into linearly separable by using a Kernel Trick
- Nonseparable data can be separated by using a Soft Margin

REVIEW OF SVMS – DECISION BOUNDARY

- Consider a set of training data D, a set of points given by the following

$$D = \{(\boldsymbol{x}_i, y_i) | \boldsymbol{x}_i \in \mathbb{R}^p, y_i \in \{-1,1\}\}_{i=1}^{n}$$

- And the hyperplane separating the two classes

$$\mathbf{w} \cdot \mathbf{x} - b = 0$$

- Can create a quadratic program of the form

$$\min \frac{1}{2} \|\mathbf{w}\|^2$$

- Subject to

$$y_i(\mathbf{w} \cdot \boldsymbol{x}_i - b) \geq 1$$

REVIEW OF SVMS – DECISION BOUNDARY

- With Lagrange multipliers, the optimial solution is

$$\min_{\mathbf{w},b} \max_{\boldsymbol{\alpha}\geq 0} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i [y_i (\mathbf{w} \cdot \boldsymbol{x}_i - b) - 1] \right\}$$

- The solutions for **w** and b can then be found as

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

$$b = \frac{1}{N_{SV}} \sum_{i=1}^{N} \mathbf{w} \cdot \boldsymbol{x}_i - y_i$$

REVIEW OF SVMS – DECISION BOUNDARY

- Dual form can be expressed as:

$$\max_{\alpha_i} \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x}_j) \right\}$$

- Subject to

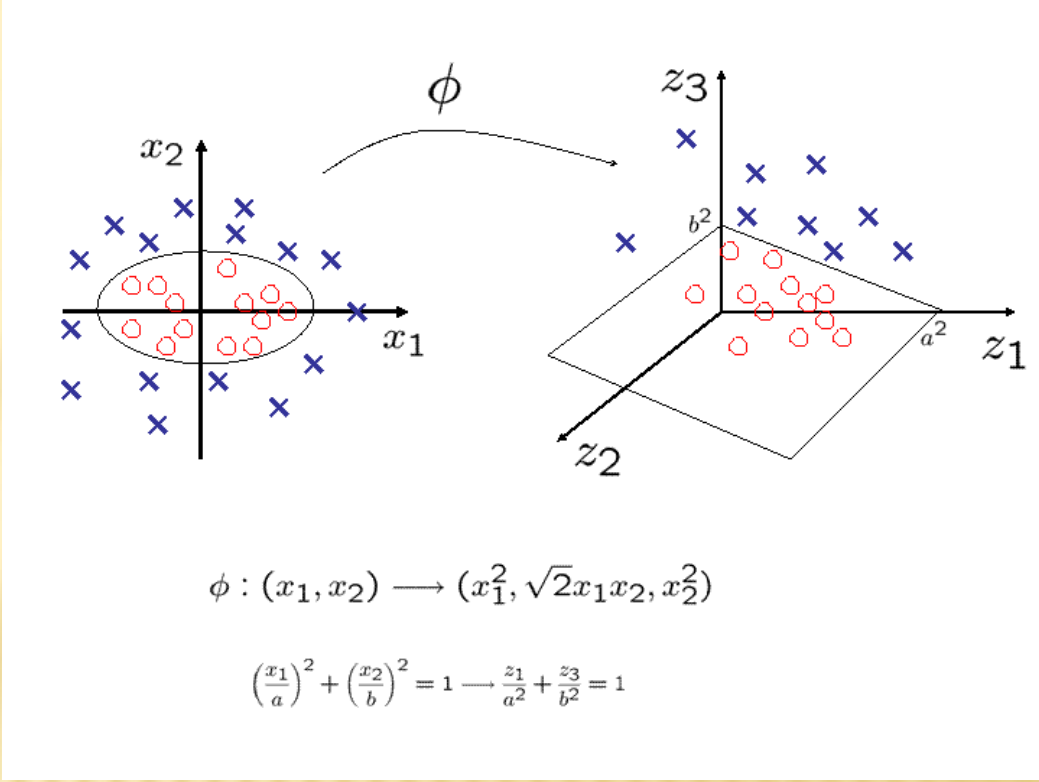
$$\alpha_i \geq 0$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

- Where $\mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a kernel function

REVIEW OF SVMS – KERNEL TRICK

- Projecting nonlinearly separable data into a higher dimension to separate
- Some example kernels include
- Polynomial: $\mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i \cdot \boldsymbol{x}_j)^d$
- Gaussian Radial Basis function: $\mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}$



REVIEW OF SVMS – KERNEL TRICK

- The Kernel can be represented as:

$$k(\boldsymbol{x_i}, \boldsymbol{x_j}) = \varphi(\boldsymbol{x_i}) \cdot \varphi(\boldsymbol{x_j})$$

- The value of **w** in this space can be calculated as

$$\boldsymbol{w} = \sum_i \alpha_i y_i \varphi(\boldsymbol{x}_i)$$

- For classification, you can use the kernel trick via the following:

$$\boldsymbol{w} \cdot \varphi(\boldsymbol{x}_j) = \sum_i \alpha_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

- No general way to represent a value **w'** such that $\boldsymbol{w} \cdot \varphi(\boldsymbol{x}) = k(\boldsymbol{w}', \boldsymbol{x})$

REVIEW OF SVMS – SOFT MARGIN

- Previous method cannot be used easily with overlapping data (overfitting a kernel may be possible but it's not desirable)
- Soft Margin introduces a slack variable ξ_i which measures the degree of misclassification of data point $\boldsymbol{x}_i$
- Constraint becomes

$$y_i(\mathbf{w} \cdot \boldsymbol{x}_i - b) \geq 1 - \xi_i$$

REVIEW OF SVMS – SOFT MARGIN

- With Lagrange Multipliers, the program becomes

$$\min_{\mathbf{w},\boldsymbol{\xi},b}\max_{\boldsymbol{\alpha},\boldsymbol{\beta}}\left\{\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i[y_i(\mathbf{w}\cdot\boldsymbol{x}_i - b) - 1 + \xi_i\,] - \sum_{i=1}^{n}\beta_i\xi_i\right\}$$

- Dual version of this problem can be found

$$\max_{\boldsymbol{\alpha}}\left\{\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j k(\boldsymbol{x}_i,\boldsymbol{x}_j)\right\}$$

- Subject to

$$0 \le \alpha_i \le C$$

$$\sum_{i=1}^{n}\alpha_i y_i = 0$$

SVMS IN SPEAKER RECOGNITION

THE MANY DIFFERENT WAYS

- Unlike with GMM based models where only the number of mixtures is generally altered between implementations, SVM models can vary greatly between implementation
- Kernels used may vary depending on who developed it
- Some people have made success with using the kernels talked about before

THE THREE TYPES OF KERNELS

- Kernels can be broken down into three types:
  - Input Space
    - Based on just the MFCCs
    - Dealing with every single input MFCC
  - Score Space
    - Dealing with the score of the input file against a speech model
  - Feature Space
    - Dealing with model parameters of the input file

FISHER KERNEL

- This model preserves the fact that speech is still probabilistic in nature
- A fusion between a generative model like GMMs and the discriminative classifier of SVMs
- This kernel is a whole sequence kernel, where the entire sequence is used to generate a single very high dimensional point

FISHER KERNEL

- Kernel is defined as:

$$k(x, x') = U_\theta(x)^{\mathrm{T}} M^{-1} U_\theta(x')$$

- Where $U_\theta(x)$ is the Fisher Score and defined as:

$$U_\theta(x) = \nabla_\theta \log(p(x|\theta))$$

- And M is the Fisher Information Matrix defined as:

$$M = E[U_\theta(x) U_\theta(x)^{\mathrm{T}}]$$

- The parameters θ can be trained with an expectation maximization algorithm
- The Fisher Score can be augmented by the actual log score to form the following:

$$\Psi(x) = \begin{bmatrix} \nabla_\theta \log(p(x|\theta)) \\ \log(p(x|\theta)) \end{bmatrix}$$

LIKELIHOOD RATIO KERNEL

- Similar to the Fisher Kernel
- Kernel is defined as:

$$k(x, x') = \Psi(x)^{\mathrm{T}} M^{-1} \Psi(x')$$

- This kernel is based on a likelihood ratio between two models:

$$\Psi(x) = \nabla_\theta \frac{\log P(x|\theta_1)}{\log P(x|\theta_2)}$$

- Like the Fisher Kernel, this may be augmented with the original likelihood ratio to form:

$$\Psi(x) = \begin{bmatrix} \nabla_\theta \dfrac{\log P(x|\theta_1)}{\log P(x|\theta_2)} \\ \dfrac{\log P(x|\theta_1)}{\log P(x|\theta_2)} \end{bmatrix}$$

GMM SUPERVECTOR KERNEL

- Based on comparing means of a GMM
- KL Divergence Between two GMMs:

$$D(g_a||g_b) = \int_{R^N} g_a(\boldsymbol{x}) \log\left(\frac{g_a(x)}{g_b(x)}\right)$$

- Log Sum Inequality:

$$D(g_a||g_b) \leq \sum_{i=1}^{N} \lambda_i D(N(\cdot\, \boldsymbol{\mu}_i^a, \boldsymbol{\Sigma}_\mathrm{i})||N(\cdot\, \boldsymbol{\mu}_i^b, \boldsymbol{\Sigma}_\mathrm{i}))$$

GMM SUPERVECTOR KERNEL

- Approximation for diagonal covariance

$$d(\boldsymbol{\mu}^a, \boldsymbol{\mu}^b) = \frac{1}{2}\sum_{i=1}^{N} \lambda_i (\boldsymbol{\mu}_i^a - \boldsymbol{\mu}_i^b)\boldsymbol{\Sigma}_{\mathrm{i}}^{-1}(\boldsymbol{\mu}_i^a - \boldsymbol{\mu}_i^b)$$

- Final Inequality

$$0 \leq D(g_a || g_b) \leq d(\boldsymbol{\mu}^a, \boldsymbol{\mu}^b)$$

GMM SUPERVECTOR KERNEL

- From this distance we can create an inner product between the means which is the kernel

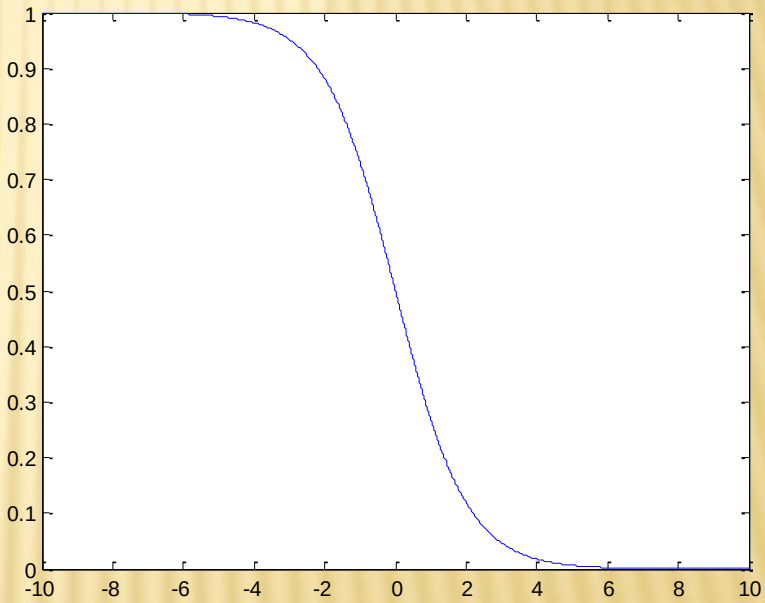
$$k(utt_a, utt_b) = \sum_{i=1}^{N} \lambda_i \boldsymbol{\mu}_i^a \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_i^b$$

$$= \sum_{i=1}^{N} \left(\sqrt{\lambda_i} \boldsymbol{\Sigma}^{-\frac{1}{2}} \boldsymbol{\mu}_i^a \right)^{\mathrm{T}} \left(\sqrt{\lambda_i} \boldsymbol{\Sigma}^{-\frac{1}{2}} \boldsymbol{\mu}_i^b \right)$$

AFTER GETTING SCORES

- After getting scores, it is necessary to calibrate the output with scores from other parts of the system
- Platt suggests to train an additional sigmoid function to map the SVM outputs to posterior probabilities
- Sigmoid Function:

$$P(y = 1|f) = \frac{1}{1 + e^{Af+B}}$$



FITTING THE SIGMOID

- Consider a training set defined as (f_i, y_i)
- Map it to a training set, (f_i, t_i) where t_i equals

$$t_i = \frac{y_i + 1}{2}$$

FITTING THE SIGMOID

- The parameters A and B are found by minimizing the negative log likelihood of the training data

$$\min -\sum_i t_i \log(p_i) + (1 - t_i) \log(1 - p_i)$$

- Where

$$p_i = \frac{1}{1 + e^{Af_i + B}}$$

- Minimization is a two-parameter minimization

TRAINING THE SIGMOID

- Easiest training set to use is the same training examples used to fit the SVM
- However, this causes bias that gets worse when one performs a nonlinear fit on the data
- One can hold a fraction of the training set (approximately 30% or so) to train the sigmoid instead of the SVM. The SVM can then be retrained using the entire set

TRAINING THE SIGMOID - CROSS-VALIDATION

- Training set is split into 3 parts
- 3 SVMs are created using permutations of 2 of the 3 parts.
- The f_is are evaluated on the remaining third.
- The union of the three sets of f_is form the training set of the sigmoid.

NAP IN SVMS

NAP

- In SVMs, Nuisance Attribute Projection is one type of kernel used.
- Like regular kernels this can be applied in the feature space or the input space
- However, most people apply it in feature space over input space

NAP IN FEATURE SPACE

- A Projection, $P = I - U_m U_m^{\mathrm{T}}$ is used to filter out the nuisance attributes
- Optimize the following criteria

$$U_m^* = \operatorname{argmin} \sum_{i,j} M_{i,j} \left\| P \cdot \Phi\left(\boldsymbol{v}_i^d\right) - P \cdot \Phi\left(\boldsymbol{v}_j^d\right) \right\|$$

- Such that

$$P = I - U_m U_m^{\mathrm{T}}$$
$$U_m^{\mathrm{T}} U_m = I$$

- Where M is a weight matrix whose elements $M_{i,j}$ are

$$M_{i,j} = \begin{cases} 1, & if\ \boldsymbol{v}_i^d\ and\ \boldsymbol{v}_j^d\ from\ the\ same\ speaker \\ 0, & else \end{cases}$$

NAP IN FEATURE SPACE

- U_m^* can be used to find the *m* largest eigenvalues of the symmetric eigenvalue problem:

$$AZ(M)A^{\mathrm{T}}U_m = U_m\Lambda$$

- Where

$$Z(M) = diag(M \cdot \mathbf{1}) - M$$

- **1** is a column vector of all ones
- And

$$A = \left[\Phi\left(\boldsymbol{v}_1^d\right) \quad \Phi(\boldsymbol{v}_1^d) \quad \cdots \quad \Phi(\boldsymbol{v}_n^d)\right]$$

NAP IN FEATURE SPACE

- For nonlinear kernels, kernel PCA is employed
- Eigenvectors are represented as

$$U_m^* = AZ^{1/2}Y_m$$

- Where

$$Z^{1/2} = \left(diag(M \cdot \mathbf{1})\right)^{1/2} - \left(diag(M * \mathbf{1})\right)^{-\frac{1}{2}}M$$

- And the columns of Y_m represent how to construct the eigenvectors of U_m^* as a combination of feature vectors

NAP IN FEATURE SPACE

- Substituting it in

$$AZ^{1/2}Z^{1/2}A^{\mathrm{T}}AZ^{1/2}Y_m = AZ^{1/2}Y_m\Lambda$$

- Y_m can be derived through m eigenvectors with largest eigenvalues in the following problem

$$Z^{1/2}GZ^{1/2}Y_m = Y_m\Lambda$$

- Where $G = A^{\mathrm{T}}A$ is the Gram matrix for development set with entries

$$G_{i,j} = k\left(\boldsymbol{v}_i^d, \boldsymbol{v}_j^d\right)$$

NAP IN FEATURE SPACE

- The eigenvalue problem is reduced to that of $Z^{1/2}GZ^{1/2}$ whose size is determined by the number of features in the development set

FEATURE SPACE NAP KERNEL

$$k_{nap}(\boldsymbol{v}_1, \boldsymbol{v}_2) = \langle P \cdot \Phi(\boldsymbol{v}_1), P \cdot \Phi(\boldsymbol{v}_2) \rangle$$
$$= \langle \Phi(\boldsymbol{v}_1), \Phi(\boldsymbol{v}_2) \rangle - \left\langle U_m^{\mathrm{T}} \cdot \Phi(\boldsymbol{v}_1), U_m^{\mathrm{T}} \cdot \Phi(\boldsymbol{v}_2) \right\rangle$$
$$= k(\boldsymbol{v}_1, \boldsymbol{v}_2) - \left\langle U_m^{\mathrm{T}} \cdot \Phi(\boldsymbol{v}_1), U_m^{\mathrm{T}} \cdot \Phi(\boldsymbol{v}_2) \right\rangle$$

- Using the previous representations we can express the kernel as

$$k_{nap}(\boldsymbol{v}_1, \boldsymbol{v}_2) = k(\boldsymbol{v}_1, \boldsymbol{v}_2) - \tilde{\boldsymbol{v}}_1 \cdot Z^{1/2} Y_m Y_m^{\mathrm{T}} Z^{1/2} \cdot \tilde{\boldsymbol{v}}_2$$

- Where

$$\tilde{\boldsymbol{v}}_i = \left[k(\boldsymbol{v}_i, \boldsymbol{v}_1^d) \quad k(\boldsymbol{v}_i, \boldsymbol{v}_2^d) \quad \dots \quad k(\boldsymbol{v}_i, \boldsymbol{v}_n^d) \right]$$

NAP IN INPUT SPACE

- Projection is found in the input space through the following optimization problem

$$U_m^* = \operatorname{argmin} \sum_{i,j} M_{i,j} \left\| \Phi\left(P \cdot \boldsymbol{v}_i^d\right) - \Phi\left(P \cdot \boldsymbol{v}_j^d\right) \right\|$$

- Such that

$$P = I - U_m U_m^{\mathrm{T}}$$
$$U_m^{\mathrm{T}} U_m = I$$

NAP IN INPUT SPACE

- Using the relationship between feature transformation and kernel functions, we get

$$\left\|\Phi\left(P\cdot\boldsymbol{v}_i^d\right)-\Phi\left(P\cdot\boldsymbol{v}_j^d\right)\right\| = k\left(P\cdot\boldsymbol{v}_i^d,P\cdot\boldsymbol{v}_i^d\right)+k\left(P\cdot\boldsymbol{v}_j^d,P\cdot\boldsymbol{v}_j^d\right) - 2k\left(P\cdot\boldsymbol{v}_i^d,P\cdot\boldsymbol{v}_j^d\right)$$

- Write the objective function as

$$Q=2\sum_{i,j}M_{i,j}\left[k\left(P\cdot\boldsymbol{v}_i^d,P\cdot\boldsymbol{v}_i^d\right)+k\left(P\cdot\boldsymbol{v}_j^d,P\cdot\boldsymbol{v}_j^d\right)\right]$$

SOLVING FOR THE OPTIMUM PROJECTION

- For nonlinear kernels this objective function is nonlinear
- Gradient Descent can be used instead
- Using a radial basis kernel and a polynomial kernel, gradient descent is guaranteed to find the global optimal solution

NAP IN INPUT SPACE

- Dimension of Input Space is much lower than Feature Space
- Projection over Input Space can be carried out efficiently

EERS VS. NUMBER OF NUISANCE ATTRIBUTES PROJECTED OUT

In feature space, the EER decreases with increasing dimensionality of U (number of nuisance attributes projected out).

In input space, the EER decreases to a point and then increases again

Overall, feature space NAP is better at lowering EER than input space NAP

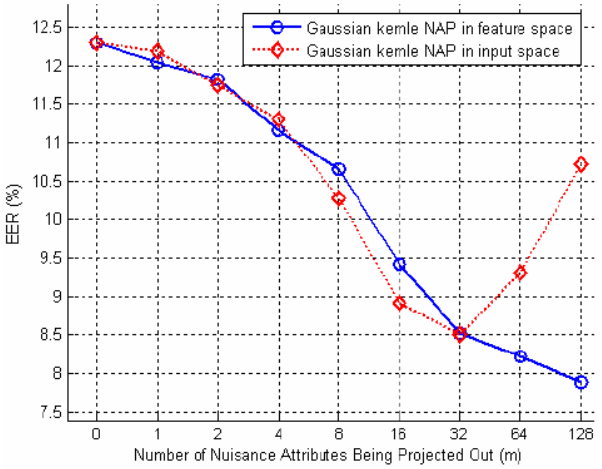


Figure 1: *EER vs. number of nuisance attributes projected out for Gaussian kernel NAP in feature and input spaces.*

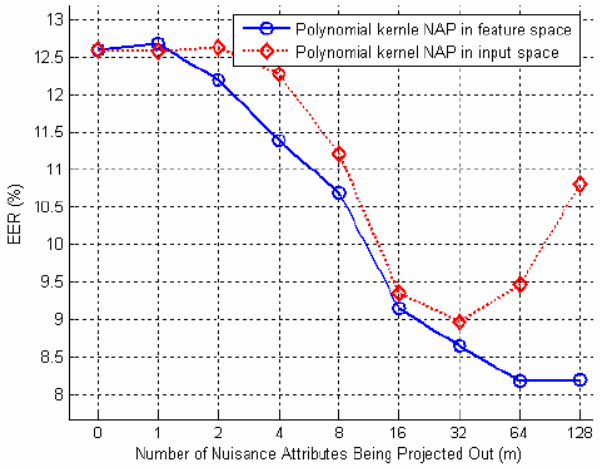


Figure 2: *EER vs. number of nuisance attributes projected out for polynomial kernel NAP in feature and input spaces.*

CONCLUSION

- SVMs are an alternative to GMMs for speaker recognition.
- Common kernels include Polynomial, Gaussian, Fisher, and Log Likelihood.
- NAP can also be represented as a kernel