Nuisance Attribute Projection

BY DAVID CINCIRUK
ASPIRG 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
• **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

![Diagram showing the process of training and testing stages with data flow](image-url)
Training Stage

- Background Speech
- MFCC Conversion
- Background MFCCs
- UBM Generation
- UBM Adaptation
- Background Statistical Model
- Target Speech
- Target MFCCs
MFCC Conversion

- MFCCs are coefficients that represent the power of a sound over a short window
- Nonlinear scaling
- The Process:
  1. Take Fourier Transform of windowed excerpt
  2. Map powers on mel scale using triangular overlapping windows
  3. Take logs of powers at each mel frequency
  4. Take the amplitudes of the result as the MFCCs
  5. Take the Discrete Cosine Transform of the list of mel log powers
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
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

MFCC Conversion

Unknown Speech

MFCCs

Scoring Algorithm

Score

Decision Process

Accept or Reject?

Background Statistical Model

Target Statistical Model
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.
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.
• Since the means are the only component being adapted in an adapted GMM system, one could represent them as the following:

\[ \mu_k(s) = \mu_k + d_k z_k(s) \]

• Where \( \mu_k \) is the UBM mean vector. \( d_k \) a D x D diagonal matrix and \( z_k(s) \) is a speaker dependent random vector distributed according to a standard normal distribution.

• \( \mu_k(s) \) is a random variable distributed according to a normal distribution with mean \( \mu_k \) and diagonal covariance \( d_k^2 \)
Why We Need It

- Given some training utterance: \( X(s) = \{x_1(s), \ldots x_T(s)\} \)
- Represent the means (determined via MAP) as
\[
E[\mu_k(s)] = \mu_k + (I + d_k^2 \Sigma_k^{-1} N_k)^{-1} d_k^2 \Sigma_k^{-1} (F_k - N_k \mu_k)
\]
- Where
\[
N_k = \sum_{t=1}^{T} \gamma_{k,t}
\]
\[
F_k = \sum_{t=1}^{T} \gamma_{k,t} x_t(s)
\]
- Where \( \gamma_{k,t} \) is the responsibility of the mixture coefficient \( k \) given observation \( t \)
- The matrix \( d_k \) can usually be assumed to be related to the covariance matrix in the form of \( d_k^2 = \tau^{-1} \Sigma_k \) where \( \tau \) 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 \( M - M_0 = S + C \).

- We want to separate the supervector \( S \) (the speaker dependent supervector) from \( C \) (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 \((N \times M)\).
- Only a few of them will be represented due to finite length.
The channel dependent portion is much lower dimensional (spanning the subspace $U$) 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 ($C$) as the following:

$$C = UU^*(M - M_0)$$

- The Subspace $U$ has to be estimated from the data.
- Requires sufficiently large number of recordings from target data.
Estimating Channel Subspace Matrix

1. Estimate Speaker and Channel Dependent Supervector
2. Compute mean supervector of each speaker from all recordings
3. Estimate the first $n$ largest eigenvalues from covariance matrix of channel supervectors
4. Calculate Channel Component by subtracting mean supervector
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

1. Find Covariance Matrix
   \[ C = \frac{1}{N} \sum (x - \mu)(x - \mu)^T \]

2. Find Eigenvalues and Eigenvectors of Covariance Matrix
   \[ V^{-1}CV = D \]
   - 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 $V$ corresponding to the $M$ largest eigenvalues of $C$ to form $U$
- 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 $C$
- Then transform each feature vector $\mathbf{x}$ with the following equation

$$\hat{x}_t = x_t - \sum_{k=1}^{K} \gamma_{k,t} c_k$$

- Where $\gamma_{k,t}$ is the responsibility of the observation $x_t$ by the kth mixture component and $c_k$ is part of the supervector $C$ 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
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
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.
Difference between the speaker and the UBM can be represented by the following supervector

\[ M - M_0 = S + C \]

Channel dependent portion is much lower dimensional (spanning the subspace \( U \)) than the speaker dependent portion

\[ C = UU^*(M - M_0) \]
Estimate Speaker and Channel Dependent Supervector

Compute mean supervector of each speaker from all recordings

Estimate the first $n$ largest eigenvalues from covariance matrix of channel supervectors using PCA

Calculate Channel Component by subtracting mean supervector
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
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
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
Consider a set of training data $D$, a set of points given by the following

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

Definition of a line/hyperplane:

$$w \cdot x - b = 0$$

$w$ is the normal vector to the hyperplane and $\frac{b}{\|w\|}$ is the offset of the hyperplane from the origin along direction $w$.
Select two parallel hyperplanes so that they separate the data and don’t leave any points between them.

\[ w \cdot x - b = 1 \]
\[ w \cdot x - b = -1 \]

The distance between the two can be represented as \( \frac{2}{\|w\|} \)

We want to maximize the distance between the two hyperplanes so we want to minimize \( \|w\| \)
Generating a Decision Boundary

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

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

Subject to

$$y_i (w \cdot x_i - b) \geq 1$$
Solving the Optimization Problem

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

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

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

\[
\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i
\]

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

$$
\max_{\alpha_i} \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \right\}
$$

- Subject to $\alpha_i \geq 0$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$
- Where $k(x_i, x_j)$ 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.
Some example kernels include

- Polynomial: \( k(x_i, x_j) = (x_i \cdot x_j)^d \)
- Gaussian Radial Basis function: \( k(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2} \)
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.
Method introduces a slack variable $\xi_i$ which measures the degree of misclassification of data point $x_i$

Constraint becomes

$$y_i(w \cdot x_i - b) \geq 1 - \xi_i$$

With Lagrange Multipliers, the program becomes

$$\min_{w,\xi,b} \max_{\alpha,\beta} \left\{ \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i \\
- \sum_{i=1}^{n} \alpha_i [y_i(w \cdot x_i - b) - 1 + \xi_i] - \sum_{i=1}^{n} \beta_i \xi_i \right\}$$
Once again a dual version of this problem can be found

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

Subject to

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

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$
SVMs and Speaker Recognition
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
Represent the speaker recognition problem as a two class problem.
- If $\omega$ 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 $g(y) = w^T b(y)$
- $w$ is the vector of classifier parameters, the model.
- $b$ 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:

$$p(y_1, ..., y_n | \omega) = \prod_{i=1}^{n} p(y_i | \omega) = \prod_{i=1}^{n} \frac{p(\omega | y_i) p(y_i)}{p(\omega)}$$
Generalized Linear Discriminant Scoring

- Take the log of the probability (and discarding $p(y_i)$ to get the discriminant function

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

- Using the first order Taylor Series approximation, we get

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

- The $-1$ in the term has been dropped and the function was normalized by the number of frames. This does not affect classification decision
Assume we have \( g(y) \approx p(\omega = 1|y) \)

Substituting in we have:

\[
d(y|\omega = 1) = \frac{1}{n} \sum_{i=1}^{n} \frac{w^T b(y_i)}{p(\omega = 1)} = \frac{1}{np(\omega = 1)} w^T \sum_{i=1}^{n} b(y_i)
\]

\[
= \frac{1}{p(\omega = 1)} w^T b_y
\]

Where we defined the mapping \( y \rightarrow b_y \) as

\[
y \rightarrow \frac{1}{n} \sum_{i=1}^{n} b(y_i)
\]
Monomials

- A monomial is a polynomial of the form $x_{i1}x_{i2} \ldots x_{ik}$
- Where $k$ is less than or equal to the polynomial form
- Input vector: $x = [x_1 \ x_2 \ \ldots \ x_m]^T$
- The vector $b(x)$ is the vector of monomials of the input vector up to and including degree $k$
Let $w$ be the desired target model. We can represent it as

$$w^* = \arg\min_w \mathbb{E} \left[ (w^T b(x) - \omega)^2 \right]$$

This criterion can be approximated using the training set as

$$w^* = \arg\min_w \mathbb{E} \left[ \sum_{i=1}^{N_{tgt}} |w^T b(x_i) - 1|^2 + \sum_{i=1}^{N_{non}} |w^T b(z_i)|^2 \right]$$

Target training data is $x_i$ and non-target data is $z_i$.
Training method can be written in matrix form.

Define $\mathbf{M}_{tgt}$ as the matrix whose rows are the expansion of the target’s data:

$$\mathbf{M}_{tgt} = \begin{bmatrix} \mathbf{b}(x_1) & \mathbf{b}(x_2) & \ldots & \mathbf{b}(x_{N_{tgt}}) \end{bmatrix}^T$$

You can define a similar matrix for nontarget data $\mathbf{M}_{non}$.

Then define: $\mathbf{M} = [\mathbf{M}_{tgt}^T \quad \mathbf{M}_{non}^T]^T$

The problem then becomes:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \|\mathbf{Mw} - \mathbf{o}\|_2$$

Where $\mathbf{o}$ is the ideal output (consisting of $N_{tgt}$ ones and $N_{non}$ zeros).
Using the method of normal equations, the equation can be solved as

\[ M^T M w = M^T o \]

Rearranging it, it becomes

\[ (M^T M)w = M^T_{tgt} 1 + M^T_{non} = M^T_{tgt} \]

If we define \( R = M^T M \) the solution becomes

\[ w = R^{-1} M^T_{tgt} 1 \]
Generalized Linear Discriminant Sequence Kernels

- Combine previous equations to obtain the score

\[ S = \frac{1}{p(\omega = 1)} \bar{b}_y^T \mathbf{w} = \frac{1}{p(\omega = 1)} \bar{b}_y^T R^{-1} M_{tgt}^T \mathbf{1} \]

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

\[ S = \bar{b}_y^T \bar{R}^{-1} \bar{b}_x \]

- Where

\[ \bar{b}_x = \frac{1}{N_{tgt}} M_{tgt}^T \mathbf{1} \]

\[ \bar{R} = \frac{1}{N_{non} + N_{tgt}} R \]
Generalized Linear Discriminant Sequence Kernels

- This scoring method is the basis of the sequence kernel
- Map a speech sequence \( x \rightarrow \overline{b}_x \) and another speech sequence \( y \rightarrow \overline{b}_y \)
- The kernel becomes
  \[
  K_{GLDS}(x, y) = \overline{b}_x R^{-1} \overline{b}_y
  \]
- The function is not symmetric so it’s not yet a kernel
A few approximations about $\bar{R}$ can be made

Nontarget data typically dominates the calculation of $\bar{R}$

We can approximate $\bar{R} \approx \frac{1}{N_{non}} R_{non}$

We don’t need additional target data to approximate the average $\bar{R}$ 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 $\bar{R}$ is diagonal – this decreases calculations.
- If $\bar{R}$ is a full correlation matrix, $\bar{R} = U^T U$ via the Cholesky decomposition.
- The kernel can then be expressed as:

$$K_{GLDS}(x, y) = (U \bar{b}_x)^T (U \bar{b}_y)$$
Score Simplification

- Simplify the scoring with the GLDS kernel
- Suppose $f(\{x_i\})$ is the output of the SVM
  
  $$f(\{x_i\}) = \sum_{i=1}^{N} \alpha_i t_i \bar{b}_i^T R^{-1} \bar{b}_x + d$$

  Where $\bar{b}_i$ are the support vectors
  - Simplify it so
  
  $$f(\{x_i\}) = \left( \sum_{i=1}^{N} \alpha_i t_i R^{-1} \bar{b}_i + d \right)^T \bar{b}_x$$

  Where $d = [d \ 0 \ 0 \ 0]^T$
Final Model

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

\[ w = \sum_{i=1}^{N} \alpha_i t_i R^{-1} b_i + d \]

- Each target score is then simply an inner product \( w_{tgt}^T \overline{b_x} \)
Algorithm for Creating a Nontarget Model

Table 1
Creating a nontarget background

1) Given: $N_{\text{utt}}$ nontarget utterances
2) $N_{\text{tot}} = 0$
3) $r = 0$
4) For $i = 1$ to $N_{\text{utt}}$
   5) Let $\{z_i\}, i = 1, \ldots, N_z$, be the features extracted from the $i$th nontarget utterance
   6) Calculate and store $\bar{b}_z^i = (1/N_z) \sum_{i=1}^{N_z} b(z_i)$
   7) $r = r + \sum_{i=1}^{N_z} b(z_i) \cdot b(z_i)$
   8) $N_{\text{tot}} = N_{\text{tot}} + N_z$
9) Next $i$
10) Let $r = (1/N_{\text{tot}})r$
11) Let $r_{\text{sqrt}} = 1./\sqrt{r}$
12) For all $i = 1, \ldots, N_{\text{utt}}$, replace $\bar{b}_z^i = r_{\text{sqrt}} \cdot \bar{b}_z^i$
13) The set of vectors $\{\bar{b}_z^i\}$ 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 $\{x_i\}, i = 1, \ldots, N_x$, be the features extracted from the $i$th target utterance
   4) $\bar{b}_x^i = (1/N_x) \sum_{i=1}^{N_x} b(x_i)$
   5) $b_x^i = r_{sqrt}*\bar{b}_x^i$ where $r_{sqrt}$ is from the background training algorithm in Table 1
   6) Next $i$
   7) Train an SVM using: a linear kernel ($K(x, y) = x^t y$), ideal outputs of 1 for $\{\bar{b}_x^i\}$, and ideal outputs of $-1$ for $\{\bar{b}_x^i\}$ (computed in Table 1). For the trained SVM, call the resulting weights, $\alpha_i$, the support vectors, $b_i$, and the constant, $d$.
   8) Compute the target model as $w = r_{sqrt}*(\sum_{i=1}^{l} \alpha_i t_i b_i) + d$ where $d = [d \ 0 \ldots \ 0]^t$, and $t_i$ is the ideal output for the $i$th 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.
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
OUTLINE

- Review of Speaker Recognition
- Review of SVMs
- Multiple Implementations of SVMs in 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
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.
Consider a set of training data $D$, a set of points given by the following:

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

And the hyperplane separating the two classes:

$$w \cdot x - b = 0$$

Can create a quadratic program of the form:

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

Subject to:

$$y_i(w \cdot x_i - b) \geq 1$$
With Lagrange multipliers, the optimal solution is:

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

The solutions for \(w\) and \(b\) can then be found as:

\[
w = \sum_{i=1}^{n} \alpha_i y_i x_i
\]

\[
b = \frac{1}{N_{SV}} \sum_{i=1}^{N} w \cdot x_i - y_i
\]
Dual form can be expressed as:

$$\max_{\alpha_i} \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \right\}$$

Subject to

$$\alpha_i \geq 0$$

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

Where $k(x_i, x_j)$ is a kernel function
- Projecting nonlinearly separable data into a higher dimension to separate
- Some example kernels include
- Polynomial: \( k(x_i, x_j) = (x_i \cdot x_j)^d \)
- Gaussian Radial Basis function: \( k(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2} \)
The Kernel can be represented as:

\[ k(x_i, x_j) = \varphi(x_i) \cdot \varphi(x_j) \]

The value of \( w \) in this space can be calculated as

\[ w = \sum_i \alpha_i y_i \varphi(x_i) \]

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

\[ w \cdot \varphi(x_j) = \sum_i \alpha_i y_i k(x_i, x_j) \]

No general way to represent a value \( w' \) such that \( w \cdot \varphi(x) = k(w', x) \)
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 $\xi_i$ which measures the degree of misclassification of data point $x_i$

Constraint becomes

$$y_i(w \cdot x_i - b) \geq 1 - \xi_i$$
With Lagrange Multipliers, the program becomes

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

Dual version of this problem can be found

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

Subject to

\[
0 \leq \alpha_i \leq C \\
\sum_{i=1}^{n} \alpha_i y_i = 0
\]
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.
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
Kernel is defined as:

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

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:

\[ M = E[U_\theta(x)U_\theta(x)^T] \]

The parameters \( \theta \) 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:
  \[ k(x, x') = \Psi(x)^T M^{-1} \Psi(x') \]
- 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:
  \[ \Psi(x) = \begin{bmatrix} \nabla_\theta \frac{\log P(x|\theta_1)}{\log P(x|\theta_2)} \\
  \frac{\log P(x|\theta_1)}{\log P(x|\theta_2)} \end{bmatrix} \]
Based on comparing means of a GMM

KL Divergence Between two GMMs:

\[ D(g_a \| g_b) = \int_{\mathbb{R}^N} g_a(x) \log \left( \frac{g_a(x)}{g_b(x)} \right) \]

Log Sum Inequality:

\[ D(g_a \| g_b) \leq \sum_{i=1}^{N} \lambda_i D(N(\cdot \mu_i^a, \Sigma_i) \| N(\cdot \mu_i^b, \Sigma_i)) \]
Approximation for diagonal covariance

\[ d(\mu^a, \mu^b) = \frac{1}{2} \sum_{i=1}^{N} \lambda_i (\mu_i^a - \mu_i^b) \Sigma_i^{-1} (\mu_i^a - \mu_i^b) \]

Final Inequality

\[ 0 \leq D(g_a || g_b) \leq d(\mu^a, \mu^b) \]
From this distance we can create an inner product between the means which is the kernel

\[
k(\mathbf{utt}_a, \mathbf{utt}_b) = \sum_{i=1}^{N} \lambda_i \mathbf{\mu}_i^a \Sigma^{-1} \mathbf{\mu}_i^b
\]

\[
= \sum_{i=1}^{N} \left( \sqrt{\lambda_i \Sigma}^{-\frac{1}{2}} \mathbf{\mu}_i^a \right)^T \left( \sqrt{\lambda_i \Sigma}^{-\frac{1}{2}} \mathbf{\mu}_i^b \right)
\]
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:

\[
P(y = 1|f) = \frac{1}{1 + e^{Af+B}}
\]
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
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 $f_i$'s are evaluated on the remaining third.
- The union of the three sets of $f_i$'s form the training set of the sigmoid.
NAP IN SVMS
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.
A Projection, $P = I - U_m U_m^T$ is used to filter out the nuisance attributes.

Optimize the following criteria

$$U_m^* = \arg \min U_m \sum_{i,j} M_{i,j} \| P \cdot \Phi(v_i^d) - P \cdot \Phi(v_j^d) \|$$

Such that

$$P = I - U_m U_m^T$$
$$U_m^T U_m = I$$

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

$$M_{i,j} = \begin{cases} 1, & \text{if } v_i^d \text{ and } v_j^d \text{ from the same speaker} \\ 0, & \text{else} \end{cases}$$
\( U_m^* \) can be used to find the \( m \) largest eigenvalues of the symmetric eigenvalue problem:
\[
AZ(M)A^TU_m = U_m\Lambda
\]

Where
\[
Z(M) = \text{diag}(M \cdot 1) - M
\]

\( 1 \) is a column vector of all ones

And
\[
A = \begin{bmatrix}
\Phi(v_1^d) & \Phi(v_1^d) & \ldots & \Phi(v_n^d)
\end{bmatrix}
\]
For nonlinear kernels, kernel PCA is employed.

Eigenvectors are represented as
\[ U_m^* = AZ^{1/2}Y_m \]

Where
\[ Z^{1/2} = (\text{diag}(M \cdot 1))^{1/2} - (\text{diag}(M \ast 1))^{-1/2}M \]

And the columns of \( Y_m \) represent how to construct the eigenvectors of \( U_m^* \) as a combination of feature vectors.
Substituting it in

\[ AZ^{1/2} Z^{1/2} A^T AZ^{1/2} Y_m = AZ^{1/2} Y_m \Lambda \]

\( Y_m \) can be derived through \( m \) eigenvectors with largest eigenvalues in the following problem

\[ Z^{1/2} G Z^{1/2} Y_m = Y_m \Lambda \]

Where \( G = A^T A \) is the Gram matrix for development set with entries

\[ G_{i,j} = k(\nu_i^d, \nu_j^d) \]
NAP IN FEATURE SPACE

- The eigenvalue problem is reduced to that of $Z^{1/2} G Z^{1/2}$ whose size is determined by the number of features in the development set.
**FEATURE SPACE NAP KERNEL**

\[ k_{nap}(\mathbf{v}_1, \mathbf{v}_2) = \langle P \cdot \Phi(\mathbf{v}_1), P \cdot \Phi(\mathbf{v}_2) \rangle \]

\[ = \langle \Phi(\mathbf{v}_1), \Phi(\mathbf{v}_2) \rangle - \langle U_m^T \cdot \Phi(\mathbf{v}_1), U_m^T \cdot \Phi(\mathbf{v}_2) \rangle \]

\[ = k(\mathbf{v}_1, \mathbf{v}_2) - \langle U_m^T \cdot \Phi(\mathbf{v}_1), U_m^T \cdot \Phi(\mathbf{v}_2) \rangle \]

- Using the previous representations we can express the kernel as

\[ k_{nap}(\mathbf{v}_1, \mathbf{v}_2) = k(\mathbf{v}_1, \mathbf{v}_2) - \mathbf{\tilde{v}}_1 \cdot Z^{1/2}Y_mY^T_mZ^{1/2} \cdot \mathbf{\tilde{v}}_2 \]

- Where

\[ \mathbf{\tilde{v}}_i = \begin{bmatrix} k(\mathbf{v}_i, \mathbf{v}^d_1) & k(\mathbf{v}_i, \mathbf{v}^d_2) & \ldots & k(\mathbf{v}_i, \mathbf{v}^d_n) \end{bmatrix} \]
Projection is found in the input space through the following optimization problem:

\[ U_m^* = \text{argmin} \sum_{i,j} M_{i,j} \| \Phi(P \cdot \nu_i^d) - \Phi(P \cdot \nu_j^d) \| \]

Such that

\[ P = I - U_m U_m^T \]
\[ U_m^T U_m = I \]
Using the relationship between feature transformation and kernel functions, we get
\[ \| \Phi(P \cdot v_i^d) - \Phi(P \cdot v_j^d) \| \]
\[ = k(P \cdot v_i^d, P \cdot v_i^d) + k(P \cdot v_j^d, P \cdot v_j^d) \\
- 2k(P \cdot v_i^d, P \cdot v_j^d) \]

Write the objective function as
\[ Q = 2 \sum_{i,j} M_{i,j} [k(P \cdot v_i^d, P \cdot v_i^d) + k(P \cdot v_j^d, P \cdot v_j^d)] \]
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
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.