MPhil in Machine Learning and Machine Intelligence

Module MLMI2: Speech Recognition

L11: Front-End Processing, Feature Extraction & Transforms

Phil Woodland
pcw@eng.cam.ac.uk

Michaelmas 2021

Cambridge University Engineering Department

Introduction

This lecture concentrates on **front-end features** used for acoustic modelling, and includes techniques for other front-end processing techniques and feature extraction.

We will first include the following which you can find useful for traditional GMM literature

- Linear transforms to remove correlations
- Principal Component Analysis (PCA) & Linear Discriminant Analysis (LDA)
- Heteroscedastic LDA (HLDA)

Then discuss ways of using non-linear DNN-generated features (with GMM-HMMs or DNNs):

- Tandem posterior features
- Tandem bottleneck features
- Joint training of bottleneck models
- Stacked DNNs

Then, introduce other topics forms of front-end-processing, primarily with DNN models

- Direct waveform methods
- Front-end data augmentation

For many speech recognition problems, need to first identify regions of spoken data

- **speech activity** or voice activity detection
- Use in audio **segmentation**

We will also look at this problem.

Front-End Processing

The aim of the front-end processing unit is to take the raw waveform and convert it into a set of speech parameters that are (hopefully)

1. **Compact**. Reduce the high data rate (typical sampling rate of 16kHz) to a more manageable rate (typically 100Hz, or 10ms frame rate).
2. **Discriminatory**. The parameters should retain all the discriminatory information from the speech. For the example of a speaker-independent word recognition system the speech parameters should retain all the word-discriminating information whilst removing the speaker-dependent aspects of the signal (traditionally pitch in English, probably volume).

Many forms of front-end have been used, ranging from non-stationary analysis to auditory modelling. Currently the majority of front-ends are based on the use of a short-term spectral envelope (some add **log pitch** and **probability of voicing**)

This spectral analysis is usually performed by either **filter-bank** based or **linear prediction** analysis based techniques.

We will initially look at various types of **linear transforms** of the original features to model via GMMs (or a DNN).

We will then look at **non-linear** transforms of the original features via MLPs/DNNs.

A recent research direction with ANNs has been to use **direct waveform input** which should be able to learn the best features to use.

Advantages of Cepstra

Cepstral parameters are normally used with GMM-HMMs rather than features based in the spectral or log-spectral domains as

1. **More compact**. The same information may be contained in fewer parameters. High order cepstra can be removed as these represent high frequency variations in the log spectrum.
2. **Roughly Independent**. Approximately, the cepstral parameters of a particular component are uncorrelated. This allows diagonal covariance matrices to be used.
3. **Gain Independent**. Only the zeroth cepstral value (energy) is dependent on the energy level of the signal.

Using the standard feature vector good performance has been achieved.
However, is this the “best” feature set for GMM-HMM-based speech recognisers.

Questions to be asked of features used with GMM-HMMs

1. **De-correlated?**Preferable to use diagonal covariance matrices for fewer parameters and also reducing computation & storage requirements. This will only be a good model if the elements of the feature vector is (approximately) de-correlated.
2. **Compact?** Is this standard feature vector the most compact. Having the smallest possible feature vector has advantages for both training (fewer system parameters) and recognition (speed/size).
3. **Discriminating?** Has the best set of parameters, in terms of the discrimination for the required task, been selected.

Transformation to Remove Correlations

For GMMs, preferable to use diagonal covariance matrices: **de-correlate** feature vector. (rotate the space using eigenvector transformation). Note that MFCCs (via DCT) do this approximately.

Estimate a linear transformation: $\hat{\boldsymbol{o}} = \boldsymbol{A}' \boldsymbol{o}$

Require $\mathcal{E}\left\{(\hat{\boldsymbol{o}} - \hat{\mu})(\hat{\boldsymbol{o}} - \hat{\mu})'\right\} = \boldsymbol{\Lambda}$ where $\boldsymbol{\Lambda}$ is a diagonal matrix.

Rewriting the above equation

$$\boldsymbol{\Lambda} = \mathcal{E}\left\{\boldsymbol{A}'(\boldsymbol{o} - \mu)(\boldsymbol{o} - \mu)'\boldsymbol{A}\right\} = \boldsymbol{A}'\boldsymbol{W}\boldsymbol{A}$$

where

$$\boldsymbol{W} = \mathcal{E}\left\{(\boldsymbol{o} - \mu)(\boldsymbol{o} - \mu)'\right\}$$

is original covariance matrix. Express $\boldsymbol{W}$ in terms of it's eigenvectors and eigenvalues

$$\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Lambda}_w\boldsymbol{U}'$$

and substituting in previous expression

$$\boldsymbol{\Lambda} = \boldsymbol{A}'\boldsymbol{U}\boldsymbol{\Lambda}_w\boldsymbol{U}'\boldsymbol{A}$$

If $\boldsymbol{A} = \boldsymbol{U}$ then $\boldsymbol{\Lambda} = \boldsymbol{\Lambda}_w$ and the covariance matrix has been diagonalised.
An alternative transformation is

$$\boldsymbol{A} = \boldsymbol{U}_w\boldsymbol{\Lambda}_w{}^{-\frac{1}{2}}$$

which results in having the identity matrix as the covariance matrix.

Principal Component Analysis

PCA is a method of trying to **extract** the most important features to best describe the data.

A total covariance matrix, $\boldsymbol{W_g}$, is generated

$$\boldsymbol{W_g} = \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{o}(t) - \mu_{\boldsymbol{g}})(\boldsymbol{o}(t) - \mu_{\boldsymbol{g}})'$$

where $\mu_{\boldsymbol{g}}$ is the global mean of the data.

Again decomposing $\boldsymbol{W_g}$ into its eigenvalues and eigenvectors

$$\boldsymbol{W_g} = \boldsymbol{U_g}\boldsymbol{\Lambda_g}\boldsymbol{U_g}'$$

The set of features having the largest eigenvalues, $\Lambda_{g_{ii}}$, are selected.

The feature vector selected is

$$\hat{\boldsymbol{o}} = \boldsymbol{U_g}^{(r)'}\boldsymbol{o}$$

and $\boldsymbol{U_g}^{(r)}$ has the eigenvectors associated with the smallest eigenvalues set to zero.

Problems

1. The eigenvalues are not independent of scaling.
2. Does not take into account the class definitions i.e. assumes that maximum data variation corresponds to discriminatory directions.

Linear Discriminant Analysis

So far only considered within-class covariance in orthogonalisation procedure.
Extend to directly select features that are better at **discrimination**.

LDA selects features that have a small average **within-class** (co)-variance, $\boldsymbol{W}$, and a large **between-class** (co)-variance, $\boldsymbol{B}$ (computed between means of classes and global mean).

Mathematically

$$\text{Select } \hat{\boldsymbol{o}} \text{ to maximise } \left\{ \text{tr}\left(\boldsymbol{W}^{-1}\boldsymbol{B} \right) \right\}$$

where $\text{tr}\left(\boldsymbol{A}\right) = \sum_{i=1}^{n} a_{ii}$ is the **trace** of matrix $\boldsymbol{A}$.

Since $\text{tr}\left(\boldsymbol{W}^{-1}\boldsymbol{B} \right)$ is invariant to a linear transform so dimensionality of $\hat{\boldsymbol{o}}$ less than that of $\boldsymbol{o}$.

Assumptions

1. Each class can be represented by single Gaussian, all within-class covariances equal.
2. The class centroids (means) can be represented by a single Gaussian.

First de-correlate feature vector space so within-class is an identity matrix (transform $\boldsymbol{U}_w\boldsymbol{\Lambda}_w^{-\frac{1}{2}}$).

Between-class covariance is obtained in this new domain.

$$\boldsymbol{B}' = \boldsymbol{\Lambda}_w^{-\frac{1}{2}}\boldsymbol{U}_w'\boldsymbol{B}\boldsymbol{U}_w\boldsymbol{\Lambda}_w^{-\frac{1}{2}}$$

A rotation to diagonalise the between-class covariance is performed (transform $\boldsymbol{U}_b'$).

The largest elements of the resulting diagonal covariance matrix are selected.

LDA - Transform & Example

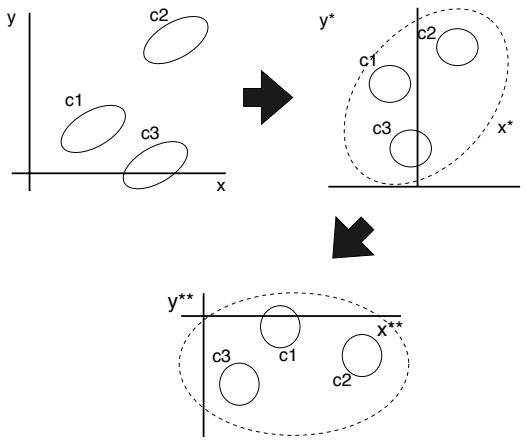
The complete transformation is

$$\boldsymbol{A}_{LDA} = \boldsymbol{U}_w\boldsymbol{\Lambda}_w^{-\frac{1}{2}}\boldsymbol{U}_b'\boldsymbol{F}$$

$\boldsymbol{F}$ is a truncation matrix which selects the dimensions with the largest d eigenvalues.

Take a 2-d example.

- Need a single dimension that discriminates classes $c1$, $c2$ and $c3$.
- From diagram can be seen that dimension x^{**} is the best in terms of the LDA criterion.



Can be applied to combined vectors of static and delta features (various orders).

HLDA and STC

LDA assumes that all within-class covariance matrix are the same.

An alternative that doesn't make this assumption is **Heteroscedastic LDA (HLDA)**.

HLDA can be estimated in an elegant maximum likelihood framework.

Aspects of HLDA are:

- Naturally falls into EM training procedure;
- Dimensions to be ignored are modelled by a global distribution
- Makes no assumptions about within class covariance matrices being the same;
- Finding the HLDA transformation:
  - requires covariance matrices to be stored for each Gaussian component
  - requires use of iterative optimisation schemes to find ML value

HTK systems have used cepstra with 1st+2nd+3rd order differentials and HLDA (52d to 39d).

Related to HLDA are **semi-tied covariance** (STC) matrices.

- STC is not a projection scheme, but aims to de-correlate the feature vector in ML fashion.
- Covariance matrix associated with a particular GMM component is

$$\boldsymbol{\Sigma}_m = \mathbf{H}^{(r)T} \boldsymbol{\Sigma}_{\text{diag}m} \mathbf{H}^{(r)}$$

Many Gaussians make use of the same transform: or even a globally estimated de-correlation.

Neural Network Features

We have so far looked at linear transforms of the original feature space.

We will now turn our attention to non-linear features extracted using neural networks.

Originally this was done with fairly shallow MLPs, and more recently with DNNs.

The original idea of **Tandem** approach was to use the MLP posterior probability outputs (or the log posterior probability) directly as additional features in a GMM-HMM.

1. Train an MLP with phone level targets
2. Use MLP log phone posteriors as a ≈ 40 dimensional representation of each speech frame
3. Use a combined feature vector of the log phone posteriors and e.g. MFCCs

The use of log phone posteriors as features provides

- very direct way of generating discriminatory features.
- final feature dimensionality depends on the size of the phone set / number of MLP output classes.
- preferable to have independent control of final feature dimension
- difficult to use context-dependent state MLP targets!

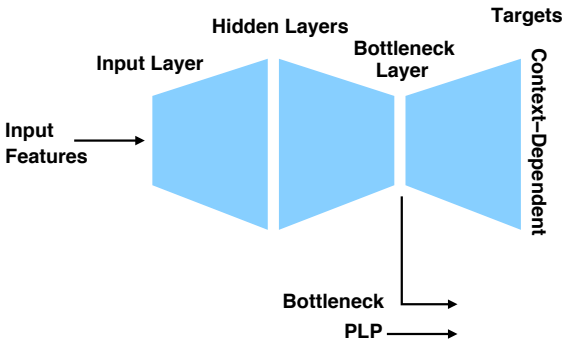
More recently an alternative, more flexible way of extracting suitably discriminatory features from an MLP has become popular: use of a bottleneck layer.

Bottleneck Feature Extraction

All information that is necessary to generate the output flows through each DNN layer.

To generate a compact set of discriminatory features use

- output of a reduced dimensionality hidden layer in the DNN
- called a **bottleneck** (or BN) layer
- BN layer typically $<$ 100 dimensions (often about 40)



- use context dependent DNN targets (possibly smaller number than for hybrid)
- after training remove the layers after bottleneck (and non-linear bottleneck activation)
- need to decorrelate bottleneck features

Bottleneck features contains all info to distinguish between the output targets.

Key design decisions:

- Size of bottleneck layer
- Position in DNN of bottleneck layer (just before output or earlier)
- Can use BN features by themselves or in tandem with e.g. MFCC/PLPs.

Comparison of Hybrid and Bottleneck DNN Approaches

The use of a bottleneck features retains benefits from both the DNN and the GMM approaches.

- All standard GMM-HMM techniques can be used with BN features
- Can apply GMM-HMM based adaptation approaches with BN features
- BN features give significant reductions in WER (more before discriminative training!)
- Modelling framework of DNN-hybrid and Tandem BN GMM approaches very different (complementary)

BN features can use many sources of data (tasks, languages) for BN network training

- Provides a way to **transfer information** from well-trained tasks to another task

Comparison of hybrid and BN tandem on MGB data:

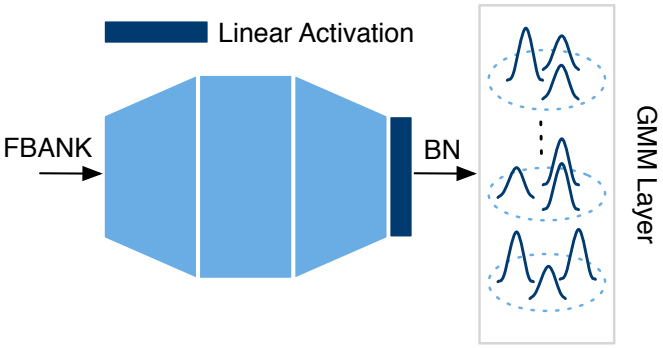
- Hybrid DNNs use 5 layers of 1000 sigmoid nodes and 6000 output targets.
- 40 FBANK inputs context window 9 frames.
- Bottleneck DNNs of 39 dimensions inserted into same config. (but with ReLU)
- GMMs model 78 features (incl. PLP+HLDA).
- 700h training set from distributed data, manual segmentation, 64k vocab, 4-gram LM.

AM	%WER
GMM-HMM ML HLDA	42.7
GMM-HMM MPE	40.7
DNN-HMM Sigmoid CE	28.4
Tandem (ReLU) BN MPE	27.0

- Tandem approach with bottleneck features competitive with DNN-HMM hybrid models
- Can be combined in “system combination” / ensemble model approaches

Joint Training of Bottleneck DNNs and GMMs

- Ideal if feature-extraction stage and GMMs trained jointly.
- Use GMMs as DNN output layer
- Initialise:
  1. conventional bottleneck DNN
  2. maximum likelihood trained GMMs trained on bottleneck features



- Find derivatives of objective function w.r.t. GMM parameters
- Find derivatives for all parameters by back-propagation through all layers
- Use SGD to train the system (need different learning rates for GMM layer)
- Training whole system with a discriminative sequence objective function works well[1].

Method has advantages:

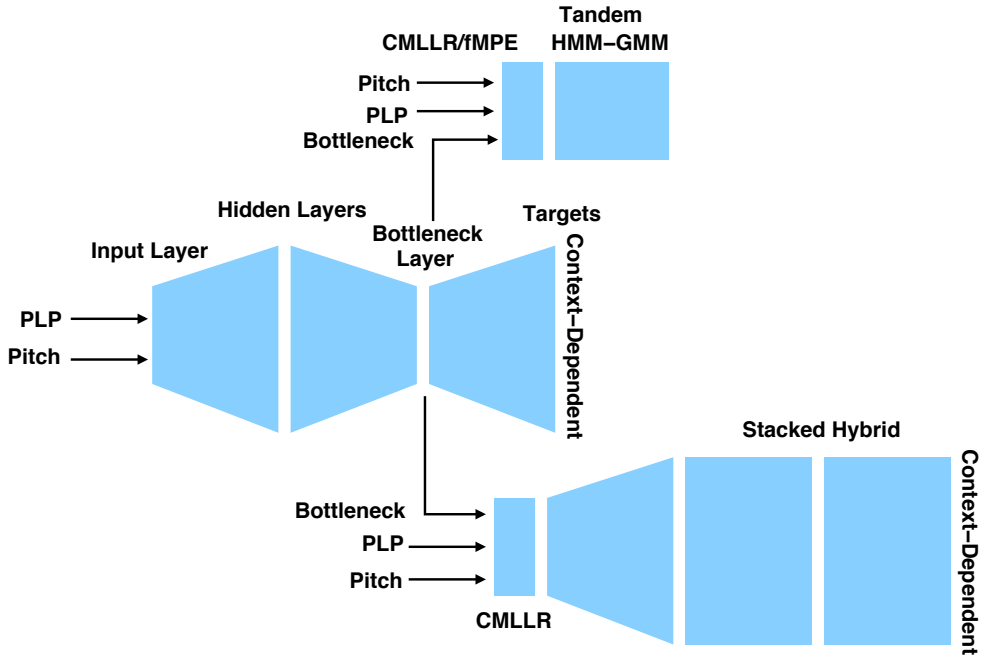
- Ensures that extracted features and GMMs are well matched
- Can still apply GMM-based adaptation methods (see future lecture)

[1] C. Zhang & P.C. Woodland. Tandem System Joint Optimisation using Discriminative Sequence Training for DNNs with GMM Output Layers. Proc. ICASSP'17

Stacked DNNs

Using a separate DNN for feature extraction leads to the idea of a stacked DNN system.

- Same BN features, different acoustic model
- Also possible to get wider temporal modelling



Direct waveform methods

Recently there has been a trend to try and learn the front-end analysis directly from data

- uses CNNs to generate features
- retains all available information!
- normally set up to be similar to traditional feature processing (CNN kernel size, stride)
- features learned often similar to Mel filterbanks (in terms of filter spacing / response)
- standard setups don't tend to improve WER on traditional processing

Issues with this type of approach

- computationally expensive
- might need more data
- front-end may perform more poorly if data is mismatched

If a front-end is learned directly from data it can in principle be applied more flexibly.

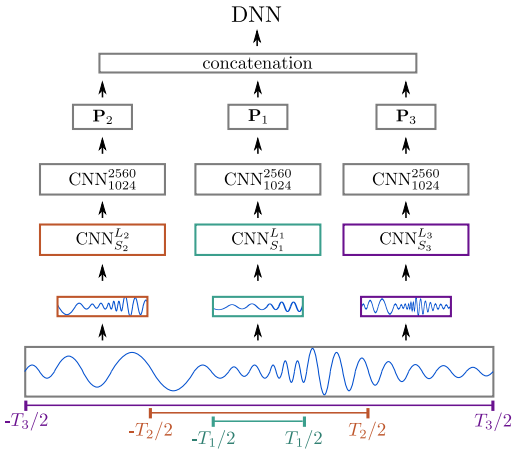
1. This includes use in **multi-channel data** from a **microphone array**
   - Std processing uses a **beamformer**: generates single signal from multiple mics
   - Can use a **neural beamformer** on either the multiple waveforms or the spectrum
2. Can have multiple views (e.g. time scales) of the data integrated into a single front-end

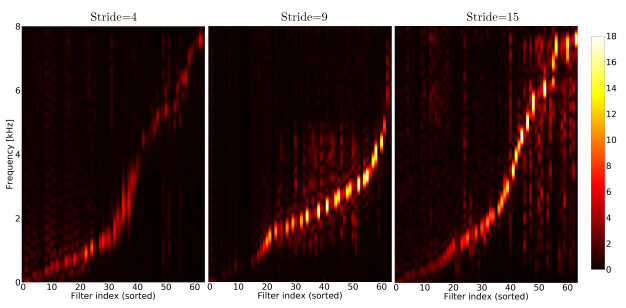
Multi-scale Waveform Processing

Combines multiple time-scales into a single front-end representation that is learned.[2]

Allows

- multiple time-scale CNN features can be computed and integrated
- filters from different strides have different responses: extract set of complementary features
- Can give rise to lower WERs than traditional front-end





[2] Patrick von Platen, Chao Zhang & Phil Woodland. "Multi-Span Acoustic Modelling using Raw Waveform Signals". Proc. Interspeech, 2019

Front-end data augmentation

For training large scale DNN based speech recognition systems it is important that

- the training data **matches** the range of conditions that will be encountered
- **large amounts** of data are available

There are numerous approaches that have been used to increase the quantity / type of data including

- Use multiple speed versions of the input data e.g. 0.9x, 1x, 1.1x (changes the pitch as well as duration)[3] An alternative is to change speed but keep pitch constant.
- Add noise to the signal of various types (stationary and non-stationary) depending on expected use-cases

Especially important for **far-field** speech recognition

- Not enough far-field data available (and few parallel recordings of close-talking, far-field data)
- Normally **add noise** of different types and account for far field response by the use of a **room simulator** (with a **room impulse response**).

The use of multiple versions of the data that are all trained on as a block (rather than any form of normalisation/adaptation) is known as **Multi-Condition Training**.

One recent (widely used) form of augmentation is called **SpecAugment**

[3] T. Ko, V. Peddinti, D. Povey, S. Khudanpur "Audio Augmentation for Speech Recognition", Proc. Interspeech 2015

SpecAugment

During training SpecAugment[4] randomly modifies the spectrogram time-frequency representation using one or more of

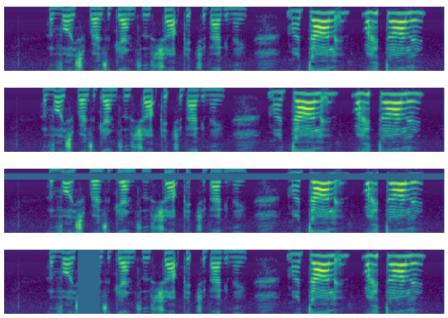
Time Warping Warps the spectrogram image at random points within a region left or right in time

Frequency Masks A set of randomly chosen consecutive frequency channels are maskd

Time Masks A number of time steps are masked.

Figure shows the effect of these changes. From top to bottom:

- original spectrogram
- with time warping
- with frequency masking
- with time masks



Normally applied so that different masks are generated each time a training utterance is used.

Can be viewed as both a method of augmentation & also **regularisation** [prevents networks overly relying on specific features]. Widely used in training **end-to-end trainable** systems.

[4] D. Park, W. Chan,Y. Zhang, C. Chiu, B. Zoph, E. Cubuk, Q. Le. "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition". Proc. Interspeech 2019

Audio Segmentation

Audio is a **continuous stream**. Stage of processing not yet discussed: audio **segmentation**.

- Needs to be somewhat different for different tasks

Different task characteristics include

- Number of channels (single, two-channel telephone recording, one per speaker etc)
- Bandwidth and audio quality / background audio
- Number of speakers (& if some speakers dominant)
- Average speaker/turn utterance length
- If there is significant **speaker overlap**

The first stage, discussed here, is to identify the spoken audio.

- speech (or voice) activity detection (**SAD** or **VAD**)
- background noise esp non-stationary noise incl. music, laughter etc makes this difficult.

Systems for speech/non-speech/audio classification used to be based on GMMs, but normally now based on **DNN-segmentation**.

Once identified speech/non-speech (& possibly audio-type), find acoustically homogenous segments & **change-points** between speakers (if no separating silence) (see next lecture).

DNN Based Speech/Non-Speech System

Often systems will include a DNN-based system at the front-end to detect regions of speech. Typically

- Two output labels (speech and non-speech)
- May have further outputs for other speech conditions
- Can have feed-forward or recurrent structure
- Need to have a **low false-rejection** rate
- Saves computation / improves performance of ASR systems.

Example segmentation speech/non-speech system for broadcast audio (MGB-1)

- **Feed-forward DNN** with **speech/non-speech output labels**
- Trained on large amount of speech/non-speech (100s of hours, **carefully aligned**)
- **wide input window** can help e.g. 0.5s or 50 frames of filter-bank coefs
- Deep structure e.g. $2200 \times 200^4 \times 2$
- Use multi-state DNN-HMM for speech & silence for **minimum duration**
- MGB-1 segmentation system state-of-the-art for multi-genre broadcast audio processing [5]
  - fairly small increase in WER over manual segmentation even for challenging data.

To get final audio segments need to be able to identify **change-points** within audio regions.

[5] L. Wang, C. Zhang, P. Woodland, M. Gales, P. Karanasou, P. Lanchantin, X. Liu, & Y. Qian (2016). Improved DNN-based segmentation for multi-genre broadcast audio. Proc. ICASSP'16, Shanghai.

Summary

Described transformation and **extraction of features**.

Linear features

- rotation to remove correlation
- principal component analysis & linear discriminant analysis
- extensions to HLDA and STC

Non-linear features from (deep) neural networks.

- original tandem formulation (phone log posteriors) & bottleneck (BN)
- improved BN features from deep networks and content-dependent targets
- BN features are convenient method to transfer powerful learned transforms across tasks
- similar SI performance from BN tandem features in GMM-HMMs and DNN-HMMs
- BN features can be jointly trained with HMM-GMM parameters by SGD

Also relevant for DNN feature extraction

- Direct learning of front-end features using CNNs
- Front-end data augmentation approaches
- In a future lecture will also talk about **wav2vec**

Finally discussed

- **speech activity detection** & **segmentation**.