Speaker Adaptation with an Exponential Transform

Daniel Povey, Geoffrey Zweig, Alex Acero

Microsoft Research, Microsoft, One Microsoft Way, Redmond, WA 98052, USA
dpovey@microsoft.com, gzweig@microsoft.com, alexac@microsoft.com

***Abstract*—In this paper we describe a linear transform that we call an Exponential Transform (ET), which integrates aspects of CMLLR, VTLN and STC/MLLT into a single transform with jointly trained components. Its main advantage is that a very small number of speaker-specific parameters is required, thus enabling effective adaptation with small amounts of speaker specific data. Our formulation shares some characteristics of Vocal Tract Length Normalization (VTLN), and is intended as a substitute for VTLN. The key part of the transform is controlled by a single speaker-specific parameter that is analogous to a VTLN warp factor. The transform has non-speaker-specific parameters that are learned from data, and we find that the axis along which male and female speakers differ is automatically learned. The exponential transform has no explicit notion of frequency warping, which makes it applicable in principle to non-standard features such as those derived from neural nets, or when the key axes may not be male-female. Based on our experiments with standard MFCC features, it appears to perform better than conventional VTLN.**

I. Introduction

Vocal Tract Length Normalization (VTLN) [1], [2], [3], [4] is a standard feature of modern speech recognition systems. The basic idea is to scale the frequency axis on a per-speaker basis so as to normalize the formant positions. These can vary by about 20% between speakers [5] because gender and other factors affect the length of the vocal tract. Currently, the most commonly used approach to VTLN operates by repeating feature extraction multiple times (e.g. 20 times), for a discrete set of warping factors, and selecting the warp that provides the highest likelihood features with respect to a simple model.

Linear-transform based implementations of VTLN have been investigated by various authors [1], [6], [7], [8], [9], [10]. The basic idea is to approximate the VTLN frequency warping by a linear transformation of the MFCC or PLP features. In some cases [1], [6], [8] this is based on an analysis that leads to a formula; in other cases [7], [9] it is based on linear transforms which are trained to approximate the conventional feature-level VTLN warping. At test time these techniques are equivalent to Constrained MLLR (CMLLR) [11] but choosing from a fixed set of transforms. Note that when applying a linear transform $\mathbf{x} \rightarrow \mathbf{A}\mathbf{x}+\mathbf{b}$, one should add $\log|\det \mathbf{A}|$ to the log-likelihoods, as dictated by the identity

$$\mathcal{N}(\mathbf{A}\mathbf{x}+\mathbf{b};\boldsymbol{\mu},\boldsymbol{\Sigma})|\det \mathbf{A}| = \mathcal{N}(\mathbf{x};\mathbf{A}^{-1}\boldsymbol{\mu}-\mathbf{A}^{-1}\mathbf{b},\mathbf{A}^{-1}\boldsymbol{\Sigma}\mathbf{A}^{-T}), \quad (1)$$

where the right hand side represents the "model-space" interpretation of the transformation. This is referred to as Jacobian compensation, since $\mathbf{A}$ is the Jacobian of the transformation. In practice not all authors include the Jacobian term; see [10] for an investigation of the effect of this. In conventional (feature-level) VTLN, cepstral variance normalization generally has to be applied because there is no natural way to do Jacobian compensation [9].

Linear-transform based approaches to VTLN have generally been found to be about as effective as standard VTLN, while being more efficient to implement. Typically about 20 transforms would be estimated at training time, and at test time one would select the best one of these based on the likelihood assigned by the model to the transformed data.

A natural question to ask is: once we are working in a linear transform based framework, why not estimate the (say) 20 transforms in a purely data-driven way, without reference to the original VTLN? For large training datasets, the number of parameters to estimate is relatively quite modest so such an approach may be feasible. One way to do this would be in a K-means framework, iteratively estimating transforms and reassigning speakers to transforms. One might even envision initializing these clusters from the VTLN warp factors, thereby nudging the system towards normal VTLN. However, our experiments along these lines (not described further) were not successful, motivating a more powerful approach.

II. The Exponential Transform

A. Basic Idea

We now turn to a form of transform which will give us a continuously varying set of transforms controlled by a single parameter. We first consider a pure linear transform; later an offset term will be introduced. The most basic form of the idea is to use a transform of the form

$$\mathbf{A}^{(s)} = \exp(t^{(s)}\mathbf{A}), \quad (2)$$

where $t^{(s)}$ is a speaker-specific scalar that may be positive or negative and that is analogous to the log of a VTLN warp factor, $\mathbf{A}$ is a global parameter matrix that is learned from data, and $\mathbf{A}^{(s)}$ is the speaker specific "exponential transform". Here, $\exp$ is the matrix exponential function, which is defined (for square matrices only) by the Taylor series expansion:

$$\exp(\mathbf{M}) \equiv \sum_{n=0}^{\infty} \frac{1}{n!}\mathbf{M}^n, \quad (3)$$

with $\mathbf{M}^n$ defined in the obvious way as a product of $\mathbf{M}$ with itself n times and $\mathbf{M}^0$ being the identity matrix $\mathbf{I}$. Thus, if the entries in $\mathbf{M}$ are small, we obtain a transform which is a small delta on the identity matrix. The attraction of this functional form is that the family of functions is closed under matrix multiplication, i.e. the product of two warping matrices is still a warping matrix.

B. Full Version

We turn now to an extension of the basic transform which addresses the further normalization issues of mean normalization and feature rotation. We augment the d dimensional input vector $\mathbf{x}$ with a one to form a vector $\mathbf{x}^+$, and denote the speaker-specific transform with $\mathbf{W}^{(s)}$. The "complete" exponential transform (ET) is:

$$\mathbf{W}^{(s)} = \mathbf{D}^{(s)}\exp(t^{(s)}\mathbf{A})\mathbf{B}, \quad (4)$$

where $\mathbf{D}^{(s)}$ handles mean offsets (and, at test time, diagonal scaling), the middle factor with the $\exp$ is the core "exponential transform" part, and $\mathbf{B}$ corresponds to MLLT/STC [12], [13]. Any quantities without the superscript s are globally shared. The dimensions are $\mathbf{D}^{(s)} \in \mathbb{R}^{d\times(d+1)}$, $\mathbf{A} \in \mathbb{R}^{(d+1)\times(d+1)}$, and $\mathbf{B} \in \mathbb{R}^{(d+1)\times(d+1)}$.

Reflecting its mean-offset function, $\mathbf{D}^{(s)}$ is a matrix with ones along a $d \times d$ diagonal, unconstrained entries in the last column, and zeros elsewhere. At test time, we optionally allow unconstrained

entries on the diagonals, but not at training time as this significantly complicates the reestimation formulae.

The intuition behind adding the $\mathbf{D}^{(s)}$ and $\mathbf{B}$ components is that if we want the factor $\exp(t^{(s)}\mathbf{A})$ to model a VTLN-like transform, which might correspond to a relatively subtle difference in the features, we need to normalize for gross effects such as speaker-dependent offsets and global correlations between parameters. Otherwise it is more likely that the central factor $\exp(t^{(s)}\mathbf{A})$ would learn instead to model these types of effects; by modeling such effects explicitly, we free the exponential term to model the effects that cannot modeled by $\mathbf{B}$ and $\mathbf{D}^{(s)}$.

We emphasize that, like linear VTLN, ET is a specially constrained form of CMLLR.

III. Model Estimation

A. Overview

At training time we need to compute the global parameters $\mathbf{A}$ and $\mathbf{B}$, and also train a model on suitably adapted features. The objective function we optimize is the data likelihood; the procedure is based on Expectation-Maximization (E-M). An overview of the training procedure is:

- Initialize the global parameters $\mathbf{A}$ and $\mathbf{B}$
- For a number of training iterations:
  - Compute $t^{(s)}$ and $\mathbf{D}^{(s)}$ for the training speakers
  - Update the model (means, variances, etc.)
  - On early iterations (e.g. the first 15 iterations), alternately:
    - Update the matrix $\mathbf{A}$, or:
    - Update the matrix $\mathbf{B}$.
- Compute a speaker independent model using just (the first d rows of) $\mathbf{B}$ as the feature-space transform.

The speaker independent model has the same mixture-of-Gaussians structure as the final speaker-adapted model, and is computed in one pass using Gaussian-level alignments from the speaker-adapted model. It is used at test time for the first-pass decoding and to obtain Gaussian-level alignments for estimating the transform.

B. Summary of notation

- The feature dimension is d.
- We assume zero-based indexing of vectors and matrices.
- We use $\mathbf{x}_t$ for the unadapted features on time t. We don't have an index for the utterance (we just assume distinct utterances have differently numbered time indices).
- $\mathbf{x}^+$ means the vector $\mathbf{x}$ with a 1 appended to it.
- $\mathbf{A}^-$, where $\mathbf{A}$ is a matrix, means $\mathbf{A}$ with its last row removed.
- $\mathbf{A}^+$, means $\mathbf{A}$ with a row with value $0\ 0\ \ldots\ 1$ appended.
- $\mathbf{A}^{(+0)}$, means $\mathbf{A}$ with a zero-valued row appended.
- Gaussian mixture components in a HMM-GMM system are indexed j, m where j is the state and m is the mixture component.
- The means and (diagonal) variances are $\boldsymbol{\mu}_{jm}$ and $\boldsymbol{\Sigma}_{jm}$, with σ^2_{jmi} as the i'th variance component.
- The Gaussian-level posteriors on time t are $\gamma_{jm}(t)$.
- Unless otherwise defined, $\mathbf{m}_i$ is the i'th row of $\mathbf{M}$ (viewed as a column vector), and $m_{i,j}$ is its i, j'th element.

C. Definition of CMLLR statistics

The sufficient statistics for CMLLR (for a particular speaker) are:

$$\beta = \sum_{t,j,m} \gamma_{jm}(t) \tag{5}$$

$$\mathbf{K} = \sum_{t,j,m} \gamma_{jm}(t) \boldsymbol{\Sigma}^{-1}_{jm} \boldsymbol{\mu}_{jm} \mathbf{x}_t^{+T} \tag{6}$$

$$\mathbf{G}_i = \sum_{t,j,m} \gamma_{jm}(t) \frac{1}{\sigma^2_{jmi}} \mathbf{x}_t^{+} \mathbf{x}_t^{+T} \tag{7}$$

for $0 \leq i < d$, and the auxiliary function is:

$$\mathcal{Q}(\mathbf{W}) = \mathrm{tr}(\mathbf{K}^T \mathbf{W}) + \log |\det(\mathbf{W}^{T\,-})| - \tfrac{1}{2} \textstyle\sum_{i=1}^{D} \mathbf{w}_i^T \mathbf{G}_i \mathbf{w}_i \tag{8}$$

D. Manipulations of CMLLR statistics

It will be necessary in our estimation algorithms to apply a transform in the feature space to CMLLR statistics; this is done as follows. Let $\mathbf{M} \in \mathbb{R}^{(d+1)\times(d+1)}$ be a matrix with last row $0\ 0\ \ldots\ 1$ that represents an affine transform. We do as follows, which is equivalent to having pre-multiplied $\mathbf{x}^+$ by $\mathbf{M}$ while collecting the statistics:

$$\mathbf{K} \leftarrow \mathbf{K}\mathbf{M}^T \tag{9}$$

$$\mathbf{G}_i \leftarrow \mathbf{M}\mathbf{G}_i\mathbf{M}^T. \tag{10}$$

Applying a transform in the model space to some statistics is done as follows. Let $\mathbf{W} \in \mathbb{R}^{d\times(d+1)}$ be the affine transform. The model-space transformation can only be done if $\mathbf{W}$ is a diagonal transform, i.e. $\mathbf{W} = [\mathbf{M}\ \mathbf{b}]$ with $\mathbf{M}$ diagonal. We'll write the (i, i)'th element of $\mathbf{M}$ as m_i. The transform corresponds to setting $x_i \leftarrow m_i x_i + b_i$. After working out how to equivalently apply this transform to the means and variances and obtaining the corresponding transforms on $\mathbf{K}$ and $\mathbf{G}_i$, we get as follows. The elements of $\mathbf{K}$ change with:

$$k_{i,j} \leftarrow m_i k_{i,j} - m_i b_i g_{i,d,j}, \tag{11}$$

where the index d is the feature dimension, and then the matrices $\mathbf{G}_i$ are scaled with:

$$\mathbf{G}_i \leftarrow m_i^2 \mathbf{G}_i. \tag{12}$$

E. Computing the matrix exponential function

For a review of ways to compute the matrix exponential function, see [14]. The method we used is one of the simpler methods discussed there. Suppose we are computing $\exp(\mathbf{M})$. Define $\mathbf{P} = 2^{-N}\mathbf{M}$. We choose the smallest integer $N \geq 0$ such that $||\mathbf{P}|| < 0.1$ (using the Frobenius norm). The method is a slight twist on the identity $\exp(\mathbf{P})^{2^N} = \exp(\mathbf{M})$, using successive squaring to compute the power. Define $\mathbf{B}_0 = \exp(\mathbf{P}) - \mathbf{I}$, computed with:

$$\mathbf{B}_0 = \sum_{n=1}^{K} \frac{1}{n!} \mathbf{P}^n, \tag{13}$$

the series is truncated when we detect that adding the latest term has not caused any change in $\mathbf{B}_0$ (we remember the number of terms as K). Then we use the recursion, for $1 \leq n \leq N$,

$$\mathbf{B}_n = \mathbf{B}_{n-1}\mathbf{B}_{n-1} + 2\mathbf{B}_{n-1}, \tag{14}$$

and the answer is given by $\exp(\mathbf{M}) = \mathbf{B}_N + \mathbf{I}$.

F. Reverse differentiating through the matrix exponential function

In this section we define for later use a function exp-backprop, of the form

$$\text{exp-backprop}(\mathbf{M}, \hat{\mathbf{X}}) = \hat{\mathbf{M}}, \tag{15}$$

where the elements of $\hat{\mathbf{M}}$ are the derivatives of scalar $f = \mathrm{tr}(\hat{\mathbf{X}}^T \exp(\mathbf{M}))$ w.r.t. the corresponding elements of $\mathbf{M}$. We assume that the intermediate quantities used while computing the matrix exponential function $\exp(\mathbf{M})$ (as described in the previous section) are available. We are going backwards through that computation

computing derivatives. We first set $\hat{\mathbf{B}}_N = \hat{\mathbf{X}}$. Then for $n = N-1, N-2, \ldots, 0$ we do:

$$\hat{\mathbf{B}}_n = \hat{\mathbf{B}}_{n+1}\mathbf{B}_n^T + \mathbf{B}_n^T\hat{\mathbf{B}}_{n+1} + 2\hat{\mathbf{B}}_{n+1}. \quad (16)$$

Next we want to compute $\hat{\mathbf{P}}$, and we will do so with

$$\hat{\mathbf{P}} = \sum_{n=1}^{K} \hat{\mathbf{P}}_n, \quad (17)$$

where $\hat{\mathbf{P}}_n$ is the part of the derivative arising from the n'th term of the truncated Taylor series 13. We set $\hat{\mathbf{P}}_1 = \hat{\mathbf{B}}_0$, and for $2 \leq n \leq K$, let

$$\hat{\mathbf{P}}_n = \frac{1}{n}\hat{\mathbf{P}}_{n-1}\mathbf{A}^T + \frac{1}{n!}\,\mathbf{A}^{n-1\,T}\,\hat{\mathbf{B}}_0, \quad (18)$$

where it is convenient to cache the powers of $\mathbf{A}$ from the forward computation. The final answer is given by $\hat{\mathbf{M}} = \frac{1}{2^N}\hat{\mathbf{P}}$.

G. Computing the speaker-specific transforms

In this section we describe how to compute the speaker-specific parameters $t^{(s)}$ and $\mathbf{D}^{(s)}$, given the sufficient statistics $\mathbf{K}$, $\mathbf{G}_i$ and β. At training time these statistics are computed with Gaussian-level alignments given by the previous iteration's speaker-specific transforms $\mathbf{W}^{(s)}$. At test time the Gausian-level alignments are computed using features transformed only with $\mathbf{B}$, and the speaker-independent model trained using single-pass retraining with features transformed only with $\mathbf{B}$.

We will omit the speaker superscript s. We first initialize $t \leftarrow 0$ and $\mathbf{D} \leftarrow [\mathbf{I}\ \mathbf{0}]$. Then we apply $\mathbf{B}$ as a feature-space transform to the statistics as described in Section III-D. We next do several iterations of update (we used three iterations). On each iteration we first re-estimate $\mathbf{D}$ and then re-estimate t.

1) Updating $\mathbf{D}$*:* In the update of $\mathbf{D}$, we first estimate a transform $\mathbf{D}'$ that will go to the right of any existing transform $\mathbf{D}$, and then modify $\mathbf{D}$ to take into account the new transform $\mathbf{D}'$. We estimate $\mathbf{D}'$ via Maximum Likelihood from $\mathbf{K}$, $\mathbf{G}_i$ and β as either an offset-only CMLLR transform (at training time) or a diagonal CMLLR transform (at test time). We then set $\mathbf{D} \leftarrow \mathbf{D}\mathbf{D}'^{+}$ (the meaning of $+$ was explained in Section III-B), and then apply $\mathbf{D}'$ as a model-space transformation to the statistics $\mathbf{K}$ and $\mathbf{G}_i$ as described in Section III-D.

2) Updating t*:* The update for t is similar to the update for $\mathbf{D}$ in that we always estimate an "incremental part" t' and add this to t. To compute t' we do a single iteration of Newton's method , starting from $t' = 0$. The update formulas are as follows. First define $\mathbf{J} \in \mathbb{R}^{d\times(d+1)}$ by:

$$\mathbf{J} = \mathbf{K} - \mathbf{S}, \quad (19)$$

where the i'th row $\mathbf{s}_i$ of $\mathbf{S}$ is the same as the i'th row $\mathbf{g}_{ii}$ of $\mathbf{G}_i$. This equals the auxiliary function derivative w.r.t $\exp(t'\mathbf{A})^-$, ignoring the log determinant. We will be maximizing the quadratic function $f(t') = at' - \frac{1}{2}bt'^2$, with

$$a = \mathrm{tr}(\mathbf{J}^T\mathbf{A}^-) + \beta\,\mathrm{tr}(\mathbf{A}) \quad (20)$$

$$b = b_1 - b_2 \quad (21)$$

$$b_1 = \left(\sum_{i=0}^{d-1} \mathbf{a}_i^T\mathbf{G}_i\mathbf{a}_i\right) \quad (22)$$

$$b_2 = \mathrm{tr}(\mathbf{J}^T(\mathbf{A}\mathbf{A})^-) \quad (23)$$

where $\mathbf{a}_i$ is the i'th row of $\mathbf{A}$. To ensure the correct sign of update even in pathological cases far from convergence, we replace $b_1 - b_2$ with $b_1 - \min(0.8b_1, b_2)$. We have never seen this flooring take place in practice. We set $t' = a/b$. We then set $t \leftarrow t + t'$, and apply the matrix $\exp(t'\mathbf{A})$ as a feature-space transformation to the statistics as decribed in Section III-D.

H. Updating $\mathbf{B}$

The accumulation and update formulas for $\mathbf{B}$ are based on those for MLLT (equivalently, global STC). Defining $\mathbf{x}'$ as $\mathbf{W}^{(s)}\mathbf{x}^+$, i.e. the current transformed features, we accumulate the sufficient statistics (for $0 \leq i < d$),

$$\mathbf{G}_i = \sum_t \frac{\gamma_{jm}(t)}{\sigma_{jmi}^2}\left(\boldsymbol{\mu}_{jm} - \mathbf{x}'\right)\left(\boldsymbol{\mu}_{jm} - \mathbf{x}'\right)^T, \quad (24)$$

and $\beta = \sum_t \gamma_{jm}(t)$. Let the result of the MLLT/STC update be the feature transformation matrix $\mathbf{C} \in \mathbb{R}^{d\times d}$, which we optimize starting from $\mathbf{C} = \mathbf{I}$ using the formulas from [15, Appendix A]. For convenience, we repeat them here. The auxiliary function is $\beta \log|\det \mathbf{C}| - \frac{1}{2}\sum_{i=0}^{d-1} \mathbf{c}_i^T\mathbf{G}_i\mathbf{c}_i$. To maximize it, for a number of iterations (e.g. 10), we do as follows: for $0 \leq i < d$,

$$\mathbf{F} \leftarrow \mathbf{C}^{-T} \quad (25)$$

$$\mathbf{c}_i \leftarrow \sqrt{\frac{\beta}{\mathbf{f}_i^T\mathbf{G}_i^{-1}\mathbf{f}_i}}\,\mathbf{G}_i^{-1}\mathbf{f}_i. \quad (26)$$

Let $\mathbf{C}_f$ be $\mathbf{C}$ extended with an extra row and column, with zeros except for a 1 in position (d, d). After estimating $\mathbf{C}$ we do as follows:

- Transform the model by setting $\boldsymbol{\mu}_{jm} \leftarrow \mathbf{C}\boldsymbol{\mu}_{jm}$
- Transform all the current speaker transforms by setting $\mathbf{W}^{(s)} \leftarrow \mathbf{C}\mathbf{W}^{(s)}$
- Set $\mathbf{A} \leftarrow \mathbf{C}_f\mathbf{A}\mathbf{C}_f^{-1}$, and $\mathbf{B} \leftarrow \mathbf{C}_f\mathbf{B}$.

I. Updating $\mathbf{A}$

The statistics for updating the matrix $\mathbf{A}$ are functions of the standard CMLLR statistics for the training speakers. These CMLLR statistics are computed with Gaussian alignments obtained with features transformed with $\mathbf{W}^{(s)}$, but the statistics themselves contain the original features $\mathbf{x}$, not the transformed features.

For each training speaker s, let the CMLLR statistics accumulated as in Section III-C be $\mathbf{K}^{(s)}$, $\mathbf{G}^{(s)}$ and β. Using the current values of $\mathbf{D}^{(s)}$ and $\mathbf{B}$, apply $\mathbf{B}$ as a feature transform to the statistics and apply $\mathbf{D}^{(s)}$ as a model-space transform to the statistics, as described in Section III-D. Let us write the transformed statistics as $\tilde{\mathbf{K}}^{(s)}$ and $\tilde{\mathbf{G}}_i^{(s)}$. Define

$$\mathbf{X}^{(s)} = \exp(t^{(s)}\mathbf{A}). \quad (27)$$

We will write the derivative of $\mathcal{Q}$ w.r.t. $\mathbf{X}^{(s)}$ as $\hat{\mathbf{X}}^{(s)}$, using notation where $\hat{x}_{i,j}^{(s)} = \frac{\partial\mathcal{Q}}{\partial x_{ij}^{(s)}}$. We have

$$\hat{\mathbf{X}}^{(s)} = \left(\tilde{\mathbf{K}}_i^{(s)} - \mathbf{S}^{(s)}\right)^{(+0)} \quad (28)$$

where $(+0)$ means appending a zero row, and the i'th row of $\mathbf{S}^{(s)}$ is given by:

$$\mathbf{s}_i^{(s)} = \tilde{\mathbf{G}}_i^{(s)}\mathbf{c}_i^{(s)}. \quad (29)$$

The derivative of $\mathcal{Q}^{(s)}$ w.r.t $\mathbf{A}$ is given by:

$$\hat{\mathbf{A}}^{(s)} = t^{(s)}\,\text{exp-backprop}(t^{(s)}\mathbf{A}, \hat{\mathbf{X}}^{(s)}). \quad (30)$$

The statistics for updating $\mathbf{A}$ are written as follows, where summations over s are over all training speakers. The index i takes values

$0 \leq i < d$.

$$\beta = \sum_s \beta^{(s)} \tag{31}$$

$$\beta_t = \sum_s t^{(s)} \beta^{(s)} \tag{32}$$

$$\hat{\mathbf{A}} = \sum_s \hat{\mathbf{A}}^{(s)} \tag{33}$$

$$\mathbf{G}_i = \sum_s {t^{(s)}}^2 \left(\tilde{\mathbf{G}}_i^{(s)} + \max(g_{i,i,i}^{(s)} - k_{i,i}^{(s)}, 0) \mathbf{e}_i \mathbf{e}_i^T \right), \tag{34}$$

where $\mathbf{e}_i$ is the unit vector in the i'th dimension. Note that $\mathbf{G}_i$ is not the same as the $\mathbf{G}_i$ quantities for the $\mathbf{B}$ update or the speaker-dependent $\mathbf{G}_i^{(s)}$ quantities in the CMLLR statistics. The (weak-sense) auxiliary function we optimize at test time is a quadratic function with quadratic part $-\frac{1}{2} \sum_{i=0}^{d-1} \mathbf{a}_i^T \mathbf{G}_i \mathbf{a}_i$, and a derivative (at the current value of $\mathbf{A}$) given by $\mathbf{H} = \hat{\mathbf{A}}^T + \beta_t \mathbf{I}$; the $\beta_t \mathbf{I}$ comes from the log determinant. The update equation is, for $0 \leq i < d$,

$$\mathbf{a}_i \leftarrow \mathbf{a}_i + \mathbf{G}_i^{-1} \mathbf{h}_i \tag{35}$$

where $\mathbf{a}_i$ and $\mathbf{h}_i$ are the i'th rows of $\mathbf{A}$ and $\mathbf{H}$, viewed as column vectors; the last row of $\mathbf{A}$ is not updated (it is always zero). The auxiliary function improvement is $\frac{1}{2} \sum_{i=0}^{d-1} \mathbf{h}_i^T \mathbf{G}_i^{-1} \mathbf{h}_i$. This should generally decrease as training progresses.

We want to keep the warp factors $t^{(s)}$ "centered" at training time so that they average to zero; this makes them more consistent between training runs, and makes the approximations we used in deriving the estimation formulas for $\mathbf{A}$ more valid (since we ensure smaller values of $t^{(s)}$). To do this, after updating $\mathbf{A}$ we take the "average part" of $\exp(t^{(s)} \mathbf{A})$, and put it into $\mathbf{B}$. The update equation is:

$$\mathbf{B} \leftarrow \exp\left(\frac{\beta_t}{\beta} \mathbf{A} \right) \mathbf{B}. \tag{36}$$

We then normalize $\mathbf{A}$ to have unit Frobenius norm; this keeps the $t^{(s)}$ values in a more consistent range from run to run (it doesn't affect the actual transforms produced by the method).

IV. Baseline VTLN implementation

As the first element of our baseline VTLN implementation we implemented the standard, feature-level form of VTLN. This operates by shifting the locations of the center frequencies of the triangular mel bins during the MFCC computation. The warping function is as diagrammed in Figure 1. The two solid lines are examples of warping functions for warping factors greater than, and less than, one. The longest, central line segment always "points" at the origin. Ours is similar to the approach used in the Attila speech recognition toolkit [16], which uses a bilinear function with the property that the inverse of each function is also in the functional family[1]; it handles the upper frequency cutoff differently from HTK [17], in which the knee is always at the same point on the x-axis. Our function is similar to HTK in that it also supports a lower cutoff (this would normally be zero if the lower frequency cutoff for the mel-bin computation was zero). In our experiments, the lower cutoff was 100 and the upper cutoff was always 600 Hz lower than the Nyquist frequency.

We implemented linear VTLN (LVTLN) in a way quite similar to [9], except that the linear functions are implemented as follows. On a small subset of data (the same for all warp factors), we compute the original features $\mathbf{x}_t$ and the warped features $\mathbf{y}_t^\alpha$, warped with warping factor α as described in the previous paragraph. We used 31 separate warping factors: $0.85, 0.86, \ldots 1.15$. For each warping factor, we estimate a CMLLR matrix $\mathbf{W}^\alpha$ to minimize the sum-of-squares error of predicting $\mathbf{y}_t^\alpha$ given $\mathbf{x}_t$: that is, if $\mathbf{z}_t^\alpha = \mathbf{W}^\alpha \mathbf{x}_t$, we first estimate $\mathbf{W}^\alpha$ to minimize the sum-of-squares difference betwen $\mathbf{z}$ and $\mathbf{y}$. We then scale each row of the CMLLR matrices so that the variance of z_i^α matches the variance of x_i (any shift in mean does not matter, for reasons that will become clear below).

[1]Brian Kingsbury, personal communication

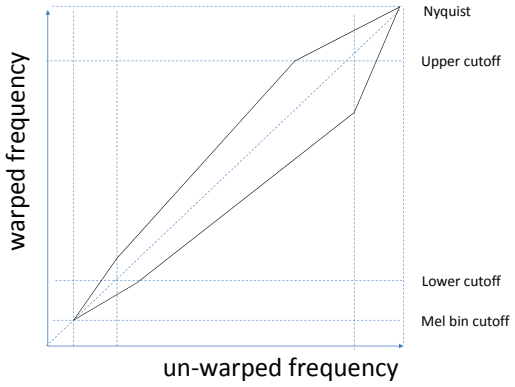


Fig. 1. VTLN warping function

Our training process for LVTLN is essentially a constrained form of speaker adapted training. On selected iterations of the training process (we used iterations 2, 4, 8 and 12), we compute sufficient statistics for CMLLR and for each training speaker, choose the $\mathbf{W}^\alpha$, that maximizes the likelihood, but treating the offset term in the last column as a variable to be optimized (we compare the auxiliary function values after optimizing this offset term). Thus, we combine VTLN with offset-only CMLLR. At test time, we optionally extend this to estimating a diagonal CMLLR matrix, applied after the $\mathbf{W}^{(s)}$ transform. We train the model on the adapted features. In experiments we reported here, we always used the Jacobian as required by the math (we found that omitting the Jacobian sometimes helped a little, but sometimes hurt a lot).

In order to implement conventional, feature-level VTLN, we used the final warp factors computed during LVTLN training and did an iteration of single-pass retraining, along with the conventionally warped features, to convert the model. At test time we used the LVTLN approach and LVTLN-trained models to work out the warp factor to use in the feature-level VTLN. In our implementation, we found the use of LVTLN derived warp factors more reliable than conventionally estimated warp factors, even for VTLN itself. As with ET, we did the speaker-independent decoding at test time using a speaker independent model with the same mixture-of-Gaussians structure as the speaker-adapted model. The speaker independent model was obtained using a single iteration of re-estimation using Gaussian alignments from the final adapted model and features, but accumulating speaker-independent statistics.

V. Experimental results

Our experiments are conducted with the recently released, open-source Kaldi toolkit [18], available from http://kaldi.sourceforge.net. We report results on the Resource Management (RM) and Wall Street Journal (WSJ) corpora. Scripts corresponding to the experiments reported here are available in version 1.0 of the toolkit.

The Resource Management corpus has 3.8 hours of training data. The Word Error Rates (WERs) we give are averaged over the Feb'89,

Feb'91, Mar'87, Oct'87, Oct'89 and Sep'92 test sets, 1.3 hours of data in total; we use the standard word-pair bigram language model.

The WSJ test sets are decoded with the 20K open vocabulary with non-verbalized pronunciations, which is the hardest of the test conditions. We used a highly-pruned version of the trigram language model included with the WSJ corpus; this is because Kaldi does not yet have a decoder that works with large language models (the full trigram model has 6.7 million entries/arcs; the pruned one has 1.5 million). We report results on the Nov'92 and Nov'93 evaluation test sets, which have 3439 and 5641 words respectively. For our results here, for fast turnaround of experiments we trained on half the SI-84 data, using randomly sampled utterances.

Both systems use decision-tree-clustered triphones and standard HMM-GMM models. In addition, for the WSJ experiments we used an extended phone set with position and stress dependent phones, but decision-tree roots corresponding to "real" phones (questions can be asked about the central phone). As reported in [18], results for this setup are comparable to previously published results on the RM and WSJ corpora. The features are based on 13-dimensional MFCCs; we show experiments either with delta and acceleration features, or processed by splicing 9 adjacent frames together and doing LDA to 40 dimensions. LDA features are further transformed via Semi-tied Covariance (STC) estimation, except for ET systems in which STC is automatically included. The RM systems had 1473 leaves and 9 000 Gaussians. The WSJ systems had 1583 leaves and 10 000 Gaussians. Whenever we accumulate statistics to estimate any kind of transformation matrix, whether global or speaker-specific, at training or test time time, we always exclude the statistics corresponding to silence.

TABLE I
BASELINE %WERS, UNADAPTED

Features	System ID	WSJ RM	WSJ Nov'92	WSJ Nov'93
Delta+Accel	tri2a	4.0	12.5	18.3
Delta+Accel+STC	tri2d	4.3	13.0	19.4
Splice+LDA	tri2e	4.7	14.3	19.1
Splice+LDA+STC	tri2f	3.9	12.2	17.7

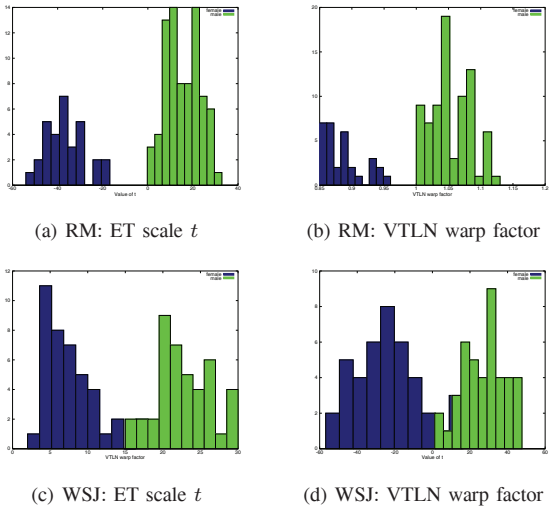
We show the unadapted WERs in Table I. Considered separately, LDA and STC both hurt performance, but together they improve it. Although this is unintuitive, the combination of LDA plus STC/MLLT is known to work well [19]. Bear in mind that ET does STC/MLLT as part of the training process, so it should be at a slight disadvantage versus conventional VTLN when working from the delta plus acceleration features.

Figure 2 shows the distribution of t values and VTLN warp factors, on RM and WSJ. This is for systems based on MFCC plus delta plus acceleration features. Both with (linear) VTLN and ET we have a very reasonable distribution of warp factors, with a good separation between male and female; this is clearer in RM, and we speculate that it has to do with the characteristics of the speakers. The number of speakers is relatively small, which accounts for the noise in the distributions.

A. Integration of CMLLR with ET/LVTLN/VTLN

We should emphasize that our implementations of both ET and LVTLN incorporate an element of Constrained MLLR. When computing the speaker-specific transform $\mathbf{W}^{(s)}$ in ET, we make the factor $\mathbf{D}^{(s)}$ a diagonal CMLLR matrix at test time (i.e. it contains a scale and an offset term for each dimension). In order to make our LVTLN results comparable, we also enabled the estimation of offset-only and diagonal CMLLR matrices after the pure "LVTLN" part of the transform. This uses the same CMLLR statistics that are used to estimate the warp factor, and it is integrated into the warp factor calculation in that we compare the likelihoods after including the effect of the diagonal or offset-only transform. In the case of feature-level VTLN, after extracting the VTLN-warped features using the warp factor obtained from the LVTLN computation, we estimated an offset-only or diagonal CMLLR transform on top of the VTLN-warped features. This was done without an extra pass of decoding, i.e. all results in Tables II and III are done with a single speaker-independent decoding pass and a single adapted decoding pass.

Fig. 2. Distribution of warp factors and t values (female dark blue, male pale green)



(a) RM: ET scale t (b) RM: VTLN warp factor

(c) WSJ: ET scale t (d) WSJ: VTLN warp factor

TABLE II
ET VERSUS LVTLN VERSUS VTLN: ON DELTA PLUS ACCELERATION FEATURES. %WERS

VTLN type	CMLLR type	System id	WSJ RM	WSJ Nov'92	WSJ Nov'93
Adapting per speaker					
None	None	tri2a	4.0	12.5	18.3
None	Diag	tri2a	3.9	12.7	17.2
ET	Diag	tri2b	**3.1**	11.5	**15.0**
LVTLN	Offset	tri2g	3.3	11.1	16.4
LVTLN	Diag	tri2g	**3.1**	**10.7**	16.5
VTLN	None	tri2g	3.7		
VTLN	Offset	tri2g	3.2		
VTLN	Diag	tri2g	**3.1**	10.9	15.9
Adapting per utterance					
None	Diag	tri2a	3.9	12.6	17.3
ET	Diag	tri2b	3.3	11.5	**15.0**
LVTLN	Offset	tri2g	3.3	11.2	16.2
LVTLN	Diag	tri2g	**3.1**	11.1	16.1
VTLN	Diag	tri2g	3.4	**10.9**	16.1

B. Results on delta and acceleration features

Table II compares ET with LVTLN and VTLN, on top of MFCC plus delta and acceleration features. The rows that say "Diag" (meaning, the transforms have a diagonal CMLLR component) are probably the most suitable ones to compare, as this is always the best configuration. We do not see any consistent pattern— none of the three methods is consistently best across all test sets. However,

it is clear that doing some form of VTLN or VTLN substitute is better than doing nothing at all. We should note that ET contains STC/MLLT, and we can see from Table I that STC makes things worse on delta and acceleration features, so in some sense ET is at a disadvantage here.

C. Results on LDA+STC features

TABLE III
ET VERSUS LVTLN VERSUS VTLN: ON SPLICED PLUS LDA PLUS STC FEATURES. %WERS

Adapting per speaker					
VTLN type	CMLLR type	System ID	WSJ RM	WSJ Nov'92	WSJ Nov'93
None	None	tri2f	3.9	12.2	17.7
ET	Diag	tri2k	**3.1**	**10.6**	**14.7**
LVTLN	Offset	tri2m	3.2	10.8	15.0
LVTLN	Diag	tri2m	**3.1**	10.7	16.5
VTLN	Offset	tri2m	4.7		
VTLN	Diag	tri2m	**3.1**	10.7	14.9
SAT	Full	tri2m	2.7	9.6	13.7
Adapting per utterance					
ET	Diag	tri2k	**3.0**	**10.4**	14.6
LVTLN	Offset	tri2m		10.6	**14.4**
LVTLN	Diag	tri2m	3.3	10.8	14.5
VTLN	Diag	tri2m	4.3	10.6	**14.4**
SAT	Full	tri2l	5.1	12.0	16.8

In Table III we show results on top of features based on LDA plus STC/MLLT. In the case of the ET models, the estimation of the STC is part of the ET computation so we just need to provide it with the LDA features. In the case of the LVTLN or VTLN, it would have been too complex to embed the estimation of STC/MLLT into the training procedure, so instead we used the STC transform estimated with the baseline LDA+STC system, and initialized the system build using alignments from the LDA+STC model. This possibly provides an unfair advantage to the LVTLN/VTLN system, as it uses an extra phase of system building and better alignments.

This time, we again do not see perfectly consistent results, but the general advantage seems to be in favor of ET. Note that our implementation of VTLN seems to fail quite badly in some circumstances on RM; we could not find the reason for this.

The bottom row of each section of Table III is with Speaker Adapted Training (SAT), in which we train with CMLLR-adapted features. We felt that this was a relevant comparison for VTLN because both the ET and LVTLN training procedures are special cases of SAT. It can be seen that when adapting per speaker, SAT outperforms all the versions of VTLN, but when adapting per utterance, the SAT trained system performs very badly and in the case of RM, is worse than a completely unadapted system (this could not necessarily be fixed by adjusting the count cutoff for estimating a transform, because the "default" transform may not be well matched to the SAT trained model).

VI. CONCLUSIONS

We have introduced a new form of adaptation which fuses elements of VTLN, CMLLR and STC/MLLT. Our method is a generic feature transformation with parameters learned from data, and does not include any explicit notion of frequency warping. The experimental results show that it generally performs about the same as linear-transform based VTLN (LVTLN) or conventional VTLN, and may have a slight advantage when combined with features based on spliced frames plus LDA plus STC/MLLT. For us, the most compelling advantage is that it is a simple, attractive formulation in which the training consists of optimizing a simple objective function, as opposed to VTLN in which many implementation details are not obvious and have to be tuned (e.g. frequency cutoffs; variance normalization; what to do with the determinant). The exponential transform is also applicable in principle to any kind of feature, which is an advantage if we want to significantly change the features.

We have noted that both ET and the linear version of VTLN are special cases of Constrained MLLR, and the training procedure is just a specially constrained form of Speaker Adapted Training (SAT). We find that SAT-trained models and standard CMLLR is better as long as we are adapting per speaker rather than per utterance (that is, is we have enough adaptation data). When adaptation must be done on a per-utterance level, however, the Exponential Transform can achieve superior results.

A longer version of this paper with more discussion, more derivations and more experimental results can be obtained as [20].

REFERENCES

[1] A. Acero and R. Stern, "Robust speech recognition by normalization of the acoustic space," in *Proc. ICASSP*. IEEE, 1991, pp. 893–896.

[2] A. Andreou, T. Kamm, and J. Cohen, "Experiments in Vocal tract Normalization," in *Proc. CAIP Workshop: Frontiers in Speech Recognition II*, 1994.

[3] L. Lee and R. Rose, "Speaker normalization using efficient frequency warping procedures," in *Proc. ICASSP*. IEEE, 1996, pp. 353–356.

[4] S. Wegmann, D. McAllaster, J. Orloff, and B. Peskin, "Speaker normalisation on conversational telephone speech," in *Proc. ICASSP*, 1996.

[5] E. Eide and H. Gish, "A parametric approach to vocal tract length normalization," in *Proc. ICASSP*. IEEE, 1996, pp. 346–348.

[6] J. McDonough, W. Byrne, and X. Luo, "Speaker normalization with all-pass transforms," in *Proc. ICSLP*, 1998.

[7] L. Uebel and P. Woodland, "An investigation into vocal tract length normalisation," in *Sixth European Conference on Speech Communication and Technology*, 1999.

[8] M. Pitz, S. Molau, R. Schlüter, and H. Ney, "Vocal tract normalization equals linear transformation in cepstral space," in *Proc. EUROSPEECH*, vol. 2001. Citeseer, 2001, pp. 2653–2656.

[9] D. Kim, S. Umesh, M. Gales, T. Hain, and P. Woodland, "Using VTLN for broadcast news transcription," in *Proc. ICSLP*. Citeseer, 2004.

[10] D. Sanand and S. Umesh, "Study of Jacobian Compensation Using Linear Transformation of Conventional MFCC for VTLN," in *Ninth Annual Conference of the International Speech Communication Association*, 2008.

[11] M. Gales, "Maximum Likelihood Linear Transformations for HMM-based Speech Recognition," *Computer Speech and Language*, vol. 12, 1998.

[12] R. A. Gopinath, "Maximum Likelihood Modeling With Gaussian Distributions For Classification," in *Proc. ICASSP*, 1998, pp. 661–664.

[13] M. Gales, "Semi-tied covariance matrices for hidden markov models," *IEEE Transactions on Speech and Audio Processing*, vol. 7, pp. 272–281, 1999.

[14] C. Moler and C. Van Loan, "Nineteen dubious ways to compute the exponential of a matrix," *SIAM review*, vol. 20, no. 4, pp. 801–836, 1978.

[15] M. J. F. Gales, "Semi-Tied Covariance Matrices For Hidden Markov Models," *IEEE Transactions on Speech and Audio Processing*, vol. 7, pp. 272–281, 1999.

[16] H. Soltau, G. Saon, and B. Kingsbury, "The IBM Attila speech recognition toolkit," in *Proc. IEEE Workshop on Spoken Language Technology*, 2010.

[17] S. Young *et al.*, *The HTK Book (for version 3.4)*. Cambridge University Engineering Department, 2009.

[18] D. Povey, A. Ghoshal *et al.*, "The Kaldi Speech Recognition Toolkit," in *Proc. ASRU (submitted)*, 2011.

[19] G. Saon, M. Padmanabhan, R. Gopinath, and S. Chen, "Maximum likelihood discriminant feature spaces," in *Proc. ICASSP*, 2000.

[20] D. Povey, G. Zweig, and A. Acero, "Speaker Adaptation with an Exponential Transform," Microsoft Research, Tech. Rep. MSR-TR-2011-101, 2011.