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 technical report we describe a linear transform that we call an Exponential Transform (ET), which integrates aspects of Constrained MLLR, 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. 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

This technical report is an extended version of our paper [1]. This introduction describes the exponential transform and our motivation for it; Section II describes in general terms the estimation processes involved in implementing the exponential transform, and Section III provides the detailed equations. Section IV explains our baseline VTLN implementation for the experiments, Section V gives experimental results, and we conclude in Section VI.

A. Vocal Tract Length Normalization (VTLN)

Vocal Tract Length Normalization (VTLN) [2], [3], [4], [5] 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 [6] because gender and other factors affect the length of the vocal tract. The "standard" approach to VTLN requires repeating feature extraction a number of times (e.g. 20 times), for a discrete set of warping factors. This is not very efficient, and can sometimes be inconvenient to implement due to the need to access the original waveform data.

B. Linear VTLN (LVTLN)

Linear-transform based implementations of VTLN have been investigated by various authors [2], [7], [8], [9], [10], [11]. The basic idea is to approximate the VTLN frequency warping by a linear transformation of the MFCC or PLP features. In some cases [2], [7], [9] this is based on an analysis that leads to a formula; in other cases [8], [10] 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) [12] 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 [11] 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 [10].

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. Linear VTLN is in effect a specially constrained form of CMLLR.

C. Linear VTLN via Maximum Likelihood

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 trivial. It is fairly obvious how one could do this in a K-means type of framework, iteratively estimating transforms and reassigning speakers to transforms. One can even envision initializing these clusters from the VTLN warp factors, thereby nudging the system towards normal VTLN, and if necessary one could enforce this relationship by disallowing the reassignment of clusters. Our experiments along these lines (not described here) were not successful, and these ideas are so obvious that most likely other researchers have tried them and had similar experiences.

D. Basic idea of the Exponential Transform

Our thought at this point was that perhaps the essence of "VTLN-ness" is that the transforms should forced to form a continuous sequence. The form that we felt was most natural was as follows:

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

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 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 a Taylor series expansion:

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

with $\mathbf{M}^n$ defined in the obvious way as a product of $\mathbf{M}$ with itself n times ($\mathbf{M}^0$ is just the identity matrix $\mathbf{I}$). The nice property of (2) is that if we multiply two of these transforms together (say with two values of t: t_1 and t_2), we get the corresponding transform for their sum $t_1 + t_2$. This matches up with certain intuitions about what it means to warp something (viewing t as analogous to the log of the warping factor).

The estimation framework we have in mind is Maximum likelihood, where in training time, we jointly optimize $t^{(s)}$ for training speakers and $\mathbf{A}$, to maximize data likelihood. At test time $\mathbf{A}$ would be fixed and the single parameter $t^{(s)}$ would be estimated for each speaker.

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

E. Problems with the basic version of the idea

There is a problem with this approach that dissuaded us from implementing it in this simple form. The issue is that there might be ways to increase the data likelihood which could take the functional form of (2) but would not be what we think of as VTLN. For example, t could correspond roughly to an energy normalization parameter (modulo certain discussions about whether the transform is linear or affine, and whether it matters). Alternatively, if the likelihood improvement from Semi-Tied Covariance [13] (a.k.a. Maximum Likelihood Linear Transform, or MLLT) were greater than that from a VTLN-like transform, we could get greater training likelihood by having $\exp(\mathbf{A})$ correspond to the STC/MLLT transform, and $t^{(s)}$ always being one.

F. Full version of the Exponential Transform

We fix these potential problems by adding new elements to the formulation. The end result can be thought of as roughly equivalent to "mean-offset MLLR plus VTLN plus MLLT/STC". The basic intuition is that if we want to capture a relatively subtle effect, we need to normalize for the big effects first. Although this increases the complexity of the method, in some sense we are only moving the complexity in the system around, because a system with the Exponential Transform (ET) will now not need MLLT/STC or clever approaches to mean normalization (we might still normalize the mean to help the first-pass decoding).

Let the feature dimension be d. We are using notation for affine transforms where $\mathbf{x}^+$ represents $\mathbf{x}$ with a one appended, and an affine transform $\mathbf{W}$ is represented as

$$\mathbf{W} = [\mathbf{A} \;\; \mathbf{b}], \tag{4}$$

where $\mathbf{A}$ is the linear part and $\mathbf{b}$ is the offset term (this transforms $\mathbf{x}$ to $\mathbf{W}\mathbf{x}^+$). The "complete" exponential transform (ET) is:

$$\mathbf{W}^{(s)} = \mathbf{D}^{(s)} \exp(t^{(s)}\mathbf{A})\mathbf{B}, \tag{5}$$

where $\mathbf{D}^{(s)}$ is a mean-offset-only CMLLR/fMLLR transform, the exponential term is the core "exponential transform" part, and $\mathbf{B}$ corresponds to MLLT/STC. 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 $\mathbf{B} \in \mathbb{R}^{(d+1)\times(d+1)}$. We will explain the significance of the last rows and columns of $\mathbf{A}$ and $\mathbf{B}$ below. This transform "acts on" the extended feature $\mathbf{x}^+$, so it transforms $\mathbf{x}$ with:

$$\mathbf{x} \rightarrow \mathbf{W}^{(s)}\mathbf{x}^+. \tag{6}$$

At test time we can optionally make $\mathbf{D}^{(s)}$ a diagonal-only CMLLR transform rather than mean only, but at training time this would significantly complicate the estimation formulae. Note that if we allowed $\mathbf{D}^{(s)}$ to be a generic CMLLR transform, the technique would be completely equivalent to CMLLR so there would be no point in our method, which is just a specially constrained form of CMLLR.

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.

G. Incorporating linear offsets

We mentioned above that $\mathbf{A}$ and $\mathbf{B}$ are both square matrices of dimension $d+1$, so as to encompass affine transforms. This doesn't add any power to the transform since $\mathbf{D}^{(s)}$ already has an offset term; it just makes the estimation of $\mathbf{A}$ converge slightly faster. The last rows of $\mathbf{A}$ and $\mathbf{B}$ are constrained to take on particular values: the last row of $\mathbf{B}$ is $0 \; 0 \; \ldots \; 1$, and the last row of $\mathbf{A}$ is zero, which means that the last row of $\exp(t^{(s)}\mathbf{A})$ is $0 \; 0 \; \ldots \; 1$. Since the transform gets applied to $\mathbf{x}^+$, the purpose of this last row is to "pass through" the last element (i.e., 1) unaltered and make it available for later transforms in the series (noting that the transforms in (5) can be viewed as being applied in order from right to left). The last columns of $\mathbf{A}$ and $\mathbf{B}$ correspond to offset terms.

II. Discussion of the estimation process

In this section we give an overview of the optimization processes needed to implement the exponential transform. We write the detailed equations in Section III. We have organized it this way because the details of Section III are mostly of interest to someone intending to implement the method, but this section is of more general interest.

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), although the estimation of $\mathbf{A}$ is not strictly E-M (it is not guaranteed to increase the likelihood, although it does in practice). 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 and features. It is used at test time for the first-pass decoding and to obtain Gaussian-level alignments for estimating the transform.

B. Notation

We now explain some aspects of our notation:

- The feature dimension is d.
- We assume zero-based indexing of vectors and matrices throughout this document.
- 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.
- $\mathbf{A}^+$, where $\mathbf{A}$ is a matrix, means $\mathbf{A}$ with a row with value $0\ 0\ \ldots\ 1$ appended.
- $\mathbf{A}^{(+0)}$, where $\mathbf{A}$ is a matrix, 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 $\gamma_{jm}(t)$.
- $\mathbf{e}_i$ is a unit vector in the i'th dimension; the dimension of $\mathbf{e}_i$ is implied by the context.
- 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. Initialization

Inputs to the training process include some baseline features (e.g. MFCCs with delta and acceleration) and a model trained on those features. In our training recipes, we started from a model with a single Gaussian per state because we intersperse the training of the exponential transform with the normal model training and mixing-up procedure; also, our belief based on prior experience is that these types of estimation processes make better progress on a model with a smaller number of Gaussians. We initialize $\mathbf{B}$ to the unit matrix and $\mathbf{A}$ to a random matrix with zero on the last row (we draw the other elements of $\mathbf{A}$ from a normal distribution).

D. Computing the speaker-specific transform

Computing the speaker-specific transform $\mathbf{W}^{(s)}$ is something that needs to be done both at training and test time. The problem is this: given fixed values of $\mathbf{A}$ and $\mathbf{B}$, we need to compute the speaker specific transform $\mathbf{W}^{(s)}$ which will be of the form (5). The first step is to compute the standard statistics as used to compute CMLLR/fMLLR (e.g. see [12]). Then we iteratively estimate the scalar $t^{(s)}$ and the CMLLR matrix $\mathbf{D}^{(s)}$. At training time, $\mathbf{D}^{(s)}$ is constrained to be an offset-only CMLLR matrix with d free parameters, but at test time it is typically a diagonal CMLLR matrix with $2d$ free parameters (we will discuss the reason for this below). The estimation of the scalar $t^{(s)}$ is done via Newton's method. The estimation of $\mathbf{D}^{(s)}$ is a special case of the standard CMLLR update formulae. We always use the Gaussian-level alignments obtained using the previously computed transform $\mathbf{W}^{(s)}$, if applicable.

E. Updating $\mathbf{B}$

Updating $\mathbf{B}$ is more straightforward than updating $\mathbf{A}$, so we will cover it first. Suppose we have computed speaker-specific exponential transforms $\mathbf{W}^{(s)}$ for the training speakers. We compute an STC/MLLT transform on top of the adapted features. This is done in the normal way [13], with statistics consisting of d matrices of size $d \times d$. Let the STC/MLLT matrix we compute be $\mathbf{C} \in \mathbb{R}^{d \times d}$, and let $\mathbf{C}_f$ be as $\mathbf{C}$ but extended with an extra row and column, consisting of zeros except for a one in the diagonal element. As the update method requires, we would at this point update the model means by pre-multiplying by $\mathbf{C}$. The speaker-specific feature transforms are now $\mathbf{C}\mathbf{W}^{(s)}$, and it not immediately obvious that these would be valid "exponential transforms", but they are. The new transform can be written as:

$$\tilde{\mathbf{W}}^{(s)} = \mathbf{C}\mathbf{D}^{(s)} \exp(t^{(s)}\mathbf{A})\mathbf{B} \quad (7)$$

$$= (\mathbf{C}\mathbf{D}^{(s)}\mathbf{C}_f^{-1})(\mathbf{C}_f \exp(t^{(s)}\mathbf{A})\mathbf{C}_f^{-1})(\mathbf{C}_f\mathbf{B}) \quad (8)$$

$$= (\mathbf{C}\mathbf{D}^{(s)}\mathbf{C}_f^{-1}) \exp(t^{(s)}\mathbf{C}_f\mathbf{A}\mathbf{C}_f^{-1})(\mathbf{C}_f\mathbf{B}), \quad (9)$$

and we can easily show that if $\mathbf{D}^{(s)}$ is an offset-only transform, then so is $\mathbf{C}\mathbf{D}^{(s)}\mathbf{C}_f^{-1}$ (we cannot show this if $\mathbf{D}^{(s)}$ is a diagonal transform, and this is why we can't let $\mathbf{D}^{(s)}$ be a diagonal transform at training time). We need to set $\mathbf{A} \leftarrow \mathbf{C}_f\mathbf{A}\mathbf{C}_f^{-1}$ and $\mathbf{B} \leftarrow \mathbf{C}_f\mathbf{B}$, and for all the training speakers' transforms, to keep them up to date, we should set $\mathbf{W}^{(s)} \leftarrow \mathbf{C}\mathbf{W}^{(s)}$. At this point we can "forget about" $\mathbf{C}$. We refer to this as the update phase for $\mathbf{B}$ even though $\mathbf{A}$ is also changed, because it updates the STC-like part of the transform.

F. Updating $\mathbf{A}$

The update formula for $\mathbf{A}$ is slightly less straightforward than the one for $\mathbf{B}$. We can without too much trouble compute the derivative of the auxiliary function w.r.t. $\mathbf{A}$. The basic approach we use to update $\mathbf{A}$ is a quasi-Newton method where we use the derivative w.r.t. $\mathbf{A}$ and a reasonably close approximation to the second derivative (the Hessian). Since evaluating the objective function after updating $\mathbf{A}$ would

involve revisiting the data (or at least, the per-speaker CMLLR statistics), we do not use any kind of line search after the quasi-Newton update of $\mathbf{A}$. This means that the update is not guaranteed to converge. Divergence is possible, but we have not seen it happen in practice.

In the rest of this section we describe how we approximate the Hessian. What we will accumulate is a matrix $\mathbf{G}_i$ for each $0 \leq i < d$, which is a positive semi-definite approximation to the negated matrix of second derivatives w.r.t. the i'th row of $\mathbf{A}$. We emphasize that approximations of the Hessian do not affect the fixed point of the update, only the convergence behavior. It would be possible (but quite tedious) to compute the exact second derivative, but since this is not even guaranteed negative definite, it would not guarantee good convergence behavior.

Define $\mathbf{X}^{(s)} = \exp(t^{(s)}\mathbf{A})$. Let us write $\mathcal{Q}'^{(s)}$ for the auxiliary function for speaker s, but ignoring the log-determinant term for now. Expressed in terms of $\mathbf{X}^{(s)}$, it is:

$$\mathcal{Q}'^{(s)}(\mathbf{X}^{(s)}) = \mathbf{K}^{(s)\,T}\,\mathbf{X}^{(s)\,-} - \tfrac{1}{2}\sum_{i=0}^{d-1}\mathbf{x}_i^{(s)\,T}\mathbf{G}_i^{(s)}\mathbf{x}_i^{(s)}, \tag{10}$$

where $\mathbf{K}^{(s)}$ and $\mathbf{G}_i^{(s)}$ are CMLLR statistics in the standard form that describe the auxiliary function as a function of $\mathbf{X}^{(s)}$; we define the CMLLR statistics below, in Section III-A, and in Section III-H we will describe how we get the statistics needed to express the auxiliary function in terms of $\mathbf{X}^{(s)}$. We now define $\mathbf{Y}^{(s)} = \mathbf{X}^{(s)} - \mathbf{I}$, which will be convenient later, and we re-express (10) in terms of $\mathbf{Y}^{(s)}$, getting:

$$\begin{aligned}\mathcal{Q}'^{(s)}(\mathbf{Y}^{(s)}) &= \left(\mathbf{K}^{(s)} - \mathbf{S}^{(s)}\right)^T\mathbf{Y}^{(s)\,-} && (11)\\ &\quad -\tfrac{1}{2}\sum_{i=0}^{d-1}\mathbf{y}_i^{(s)\,T}\mathbf{G}_i^{(s)}\mathbf{y}_i^{(s)}, && (12)\end{aligned}$$

where $\mathbf{S}^{(s)}$ is defined by $\mathbf{s}_i^{(s)} = \mathbf{g}_{ii}^{(s)}$, i.e. its i'th row is equal to the i'th row of $\mathbf{G}_i^{(s)}$.

The Taylor series expansion of $\mathbf{Y}$ in terms of $\mathbf{A}$ is:

$$\mathbf{Y}^{(s)} = t^{(s)}\mathbf{A} + \frac{1}{2}\,t^{(s)\,2}\,\mathbf{A}\mathbf{A} + \ldots \tag{13}$$

The ellipsis in (13) involves terms of higher than order than two in $t^{(s)}\mathbf{A}$, which we assume to be small. Expanding (12) using (13), we have

$$\begin{aligned}\mathcal{Q}'^{(s)}(\mathbf{A}) &= \text{linear plus constant terms in } \mathbf{A}\\ &\quad +\tfrac{1}{2}\,t^{(s)\,2}(\mathbf{K}^{(s)} - \mathbf{S}^{(s)})^T(\mathbf{A}\mathbf{A})^-\\ &\quad -\tfrac{1}{2}\,t^{(s)\,2}\sum_{i=0}^{d-1}\mathbf{a}_i^T\mathbf{G}_i^{(s)}\mathbf{a}_i\\ &\quad +\text{higher-order terms} \qquad (14)\end{aligned}$$

It is possible to accumulate sufficient statistics with the same dimensions as the standard CMLLR statistics that capture the behaviour of this approximation of the quadratic term. Unfortunately, the effect of the second line of (14) can sometimes destabilize the estimation process as it does not correspond to a negative semi-definite Hessian. What we found works better is to accumulate statistics that encapsulate an approximate Hessian w.r.t. each row; this amounts to a block-diagonal approximation of the overall Hessian, with a block per row. The second line of (14) connects each row with the corresponding column; the only part of this that operates within a row is the part that affects the i'th element of each i'th row. Even this term is not guaranteed negative semidefinite, so we only use this term (for a particular speaker) if it has the desired sign; this avoids potential instability. The accumulation phase is as follows, for $0 \leq i < d$:

$$\mathbf{G}_i = \sum_s t^{(s)\,2}\left(\mathbf{G}_i^{(s)} + \max(g_{i,i,i}^{(s)} - k_{i,i}^{(s)}, 0)\mathbf{e}_i\mathbf{e}_i^T\right). \tag{15}$$

In the update phase, the change to a row of $\mathbf{A}$ is given by $\mathbf{G}^{-1}$ times the auxiliary function derivative w.r.t that row of $\mathbf{A}$. As mentioned above, we do not include any line search because this would be quite cumbersome and would involve revisiting the training data. There is no guarantee that this update method will converge; failure to converge would be obvious as the reported auxiliary function improvements not becoming small in later iterations of update. In case this ever happens, we included in our software a learning rate parameter that can be set to less than one to slow down the update. If this ever turned out to be necessary in practice, we would insert logic to automatically detect failure of the likelihood to increase, and slow down the learning rate or backtrack in this case.

III. Details of estimation process

In this section we give detailed equations for the estimation processes needed for ET.

A. Definition of CMLLR statistics

The sufficient statistics for CMLLR (for a particular speaker) are as follows, where $0 \leq i < d$:

$$\begin{aligned}\mathbf{K} &= \sum_{t,j,m}\gamma_{jm}(t)\boldsymbol{\Sigma}_{jm}^{-1}\boldsymbol{\mu}_{jm}\mathbf{x}_t^{+\,T} && (16)\\ \mathbf{G}_i &= \sum_{t,j,m}\gamma_{jm}(t)\frac{1}{\sigma_{jmi}^2}\mathbf{x}_t^+\mathbf{x}_t^{+\,T} && (17)\\ \beta &= \sum_{t,j,m}\gamma_{jm}(t). && (18)\end{aligned}$$

When we use these quantities below, we will often put a speaker superscript $\cdot^{(s)}$ on them. The auxiliary function is:

$$\mathcal{Q}(\mathbf{W}) = \mathrm{tr}(\mathbf{K}^T\mathbf{W}) - \tfrac{1}{2}\sum_{i=1}^{D}\mathbf{w}_i^T\mathbf{G}_i\mathbf{w}_i + \log|\det(\mathbf{W}^{T\,-})|. \tag{19}$$

B. Manipulations of CMLLR statistics

There are some manipulations of CMLLR statistics that are needed in our algorithms. Since $\mathbf{W}^{(s)}$ consists of a series of chained transforms, these manipulations are sometimes needed to "normalize away" the effect of other transforms.

Applying a transform in the feature space to some statistics 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 is equivalent to having pre-multiplied $\mathbf{x}^+$ by $\mathbf{M}$ while collecting the statistics:

$$\mathbf{K} \leftarrow \mathbf{K}\mathbf{M}^T \quad (20)$$

$$\mathbf{G}_i \leftarrow \mathbf{M}\mathbf{G}_i\mathbf{M}^T. \quad (21)$$

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:

$$k_{i,j} \leftarrow m_i k_{i,j} - m_i b_i g_{i,d,j}, \quad (22)$$

where the index d is the feature dimension (this assumes zero-based indexing), and then the matrices $\mathbf{G}_i$ are scaled with:

$$\mathbf{G}_i \leftarrow m_i^2 \mathbf{G}_i. \quad (23)$$

C. Computing offset-only and diagonal transforms from CMLLR statistics

Given CMLLR statistics as described above, an offset-only transform of the form $\mathbf{W} = [\mathbf{I}\ \mathbf{b}]$ can be computed using the formula $b_i = \frac{k_{i,d}}{g_{i,d,d}}$.

In the case of a diagonal transform, we now describe the update process for each row of the transform (row i, with $0 \le i < d$). For clearer notation, we will define $s \equiv w_{i,i}$ and $o \equiv w_{i,d}$ (the scale and offset parameters). We limit solutions to $s > 0$; this is more natural and simplifies the estimation, and it makes no difference in practice, in our experience. The auxiliary function written in terms of s and o is:

$$\begin{aligned}\mathcal{Q}(s,o) &= sk_{i,i} + ok_{i,d} \\ &\quad -\frac{1}{2}s^2 g_{i,i,i} - \frac{1}{2}o^2 g_{i,d,d} - sog_{i,d,i} \\ &\quad +\beta \log s. \end{aligned} \quad (24)$$

We can solve for o in terms of s and get

$$o = (k_{i,d} - sg_{i,d,i})/g_{i,d,d} \quad (25)$$

Substituting this expression into (24), differentiating w.r.t. s, equating the derivative to zero and multiplying by s to get a quadratic function, we get the equation:

$$as^2 + bs + c = 0 \quad (26)$$

$$a = \frac{g_{i,d,i}^2}{g_{i,d,d}} - g_{i,i,i} \quad (27)$$

$$b = k_{i,i} - \frac{g_{i,d,i}k_{i,d}}{g_{i,d,d}} \quad (28)$$

$$c = \beta. \quad (29)$$

Getting $s > 0$ requires taking the negative root (since $a < 0$), so the solution is $s = (-b - \sqrt{b^2 - 4ac})/(2a)$. We get the value of o from (25).

D. 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 $N \ge 0$ 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:

$$\mathbf{B}_0 = \sum_{n=1}^{K} \frac{1}{n!}\mathbf{P}^n, \quad (30)$$

where 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 $1 \le n \le N$,

$$\mathbf{B}_n = \mathbf{B}_{n-1}\mathbf{B}_{n-1} + 2\mathbf{B}_{n-1}, \quad (31)$$

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

E. Reverse differentiating through the matrix exponential function

We also need to differentiate a scalar backwards through the matrix exponential function; this is an instance of reverse-mode automatic differentiation (but done "manually"). Suppose $\mathbf{X} \equiv \exp(\mathbf{M})$, and we define $f \equiv \mathrm{tr}(\mathbf{X}^T\hat{\mathbf{X}})$, where $\hat{\mathbf{X}}$ is a separate quantity from $\mathbf{X}$. that represents the derivative of f w.r.t. $\mathbf{X}$.

In general we will use a hat to denote the derivative of the scalar function w.r.t. an arbirary quantity, using a convention where there is no transpose, i.e. $\hat{x}_{ij} \equiv \frac{\partial f}{\partial x_{ij}}$.

In this section we are defining a function exp-backprop, of the form

$$\text{exp-backprop}(\mathbf{M}, \hat{\mathbf{X}}) = \hat{\mathbf{M}}, \quad (32)$$

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 will now describe the exp-backprop procedure. We assume that the intermediate quantities used while computing the matrix exponential function given $\mathbf{M}$ (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:

$$\hat{\mathbf{B}}_n = \hat{\mathbf{B}}_{n+1}\mathbf{B}_n^T + \mathbf{B}_n^T\hat{\mathbf{B}}_{n+1} + 2\hat{\mathbf{B}}_{n+1}. \quad (33)$$

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

$$\hat{\mathbf{P}} = \sum_{n=1}^{K} \hat{\mathbf{P}}_n, \quad (34)$$

where $\hat{\mathbf{P}}_n$ is the part of the derivative arising from the n'th term of the truncated Taylor series (30). We set $\hat{\mathbf{P}}_1 = \hat{\mathbf{B}}_0$, and for $2 \le n \le K$, let

$$\hat{\mathbf{P}}_n = \frac{1}{n}\hat{\mathbf{P}}_{n-1}\mathbf{A}^T + \frac{1}{n!}\,\mathbf{A}^{n-1\,T}\,\hat{\mathbf{B}}_0, \quad (35)$$

where it may be convenient to cache the powers of $\mathbf{A}$ from the forward computation (or just start with $\hat{\mathbf{B}}_0$ and left-multiply by $\mathbf{A}^T$ each time). The final answer is given by $\hat{\mathbf{M}} = \frac{1}{2^N}\hat{\mathbf{P}}$. It is easy to double-check this computation using a small-differences method.

F. Computing the speaker-specific transforms

In this section we describe how to compute the speaker-specific parameters t and $\mathbf{D}$ (we will take the speaker superscript $\cdot^{(s)}$ as given), 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 an "alignment 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-B. 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 II-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-B.

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:

$$\mathbf{J} = \mathbf{K} - \mathbf{S}, \tag{36}$$

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 is 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

$$a = \mathrm{tr}(\mathbf{J}^T\mathbf{A}^-) + \beta\,\mathrm{tr}(\mathbf{A}) \tag{37}$$

$$b = b_1 - b_2 \tag{38}$$

$$b_1 = \left(\sum_{i=0}^{d-1} \mathbf{a}_i^T\mathbf{G}_i\mathbf{a}_i\right) \tag{39}$$

$$b_2 = \mathrm{tr}(\mathbf{J}^T(\mathbf{A}\mathbf{A})^-) \tag{40}$$

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-B.

After iterating the estimation of $\mathbf{D}$ and t, we compute $\mathbf{W} = \mathbf{D}\exp(t\mathbf{A})\mathbf{B}$.

G. 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 $0 \le i < d$),

$$\mathbf{G}_i = \sum_t \frac{\gamma_{jm}(t)}{\sigma^2_{jmi}} (\boldsymbol{\mu}_{jm} - \mathbf{x}')(\boldsymbol{\mu}_{jm} - \mathbf{x}')^T, \tag{41}$$

and $\beta = \sum_t \gamma_{jm}(t)$. Let the result of the MLLT/STC update be the transform $\mathbf{C} \in \mathbb{R}^{d \times d}$, which we optimize starting from $\mathbf{C} = \mathbf{I}$ using the formulas from [13, 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 $0 \le i < d$,

$$\mathbf{F} \leftarrow \mathbf{C}^{-T} \tag{42}$$

$$\mathbf{c}_i \leftarrow \sqrt{\frac{\beta}{\mathbf{f}_i^T\mathbf{G}_i^{-1}\mathbf{f}_i}}\mathbf{G}_i^{-1}\mathbf{f}_i. \tag{43}$$

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 $\mathbf{B} \leftarrow \mathbf{C}_f\mathbf{B}$.

H. 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-A 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-B. Let us write the transformed statistics as $\tilde{\mathbf{K}}^{(s)}$ and $\tilde{\mathbf{G}}_i^{(s)}$. Define

$$\mathbf{X}^{(s)} = \exp(t^{(s)}\mathbf{A}). \tag{44}$$

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

$$\hat{\mathbf{X}}^{(s)} = \left(\tilde{\mathbf{K}}_i^{(s)} - \mathbf{S}^{(s)}\right)^{(+0)} \tag{45}$$

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

$$\mathbf{s}_i^{(s)} = \tilde{\mathbf{G}}_i^{(s)}\mathbf{c}_i^{(s)}. \tag{46}$$

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

$$\hat{\mathbf{A}}^{(s)} = t^{(s)}\,\text{exp-backprop}(t^{(s)}\mathbf{A}, \hat{\mathbf{X}}^{(s)}). \tag{47}$$

The statistics for updating $\mathbf{A}$ are written as follows, where summations over s are over all training speakers. The index i takes values $0 \leq i < d$.

$$\beta = \sum_s \beta^{(s)} \tag{48}$$

$$\beta_t = \sum_s t^{(s)} \beta^{(s)} \tag{49}$$

$$\hat{\mathbf{A}} = \sum_s \hat{\mathbf{A}}^{(s)} \tag{50}$$

$$\mathbf{G}_i = \sum_s {t^{(s)}}^2 \left(\tilde{\mathbf{G}}_i^{(s)} + \max(g_{i,i,i}^{(s)} - k_{i,i}^{(s)}, 0) \mathbf{e}_i \mathbf{e}_i^T \right), \tag{51}$$

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 $0 \leq i < d$,

$$\mathbf{a}_i \leftarrow \mathbf{a}_i + \mathbf{G}_i^{-1} \mathbf{h}_i \tag{52}$$

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 estimation formulas for $\mathbf{A}$ make more sense (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:

$$\mathbf{B} \leftarrow \exp\left(\frac{\beta_t}{\beta} \mathbf{A} \right) \mathbf{B}. \tag{53}$$

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 a fairly standard, nonlinear, feature-level VTLN. This operates by shifting the locations 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 [15], 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 [16], 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.

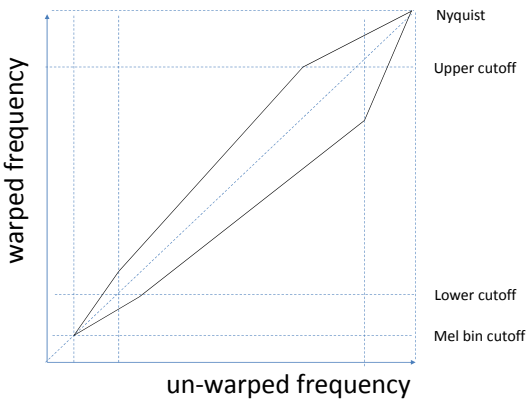


Fig. 1. VTLN warping function

We implemented linear VTLN (LVTLN) in a way fairly similar to [10], 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 using the process 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).

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

[1]Brian Kingsbury, personal communication

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 [17], 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 test results we report 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.

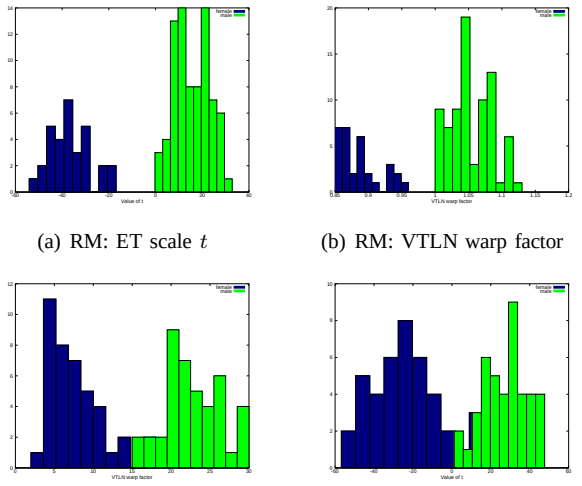
Both systems use decision-tree-clustered triphones and standard HMM-GMM models. In addition, for the WSJ experiments we used an extended phone set with position and stress dependent phones, but decision-tree roots corresponding to "real" phones (questions can be asked about the central phone). As reported in [17], results for this setup are comparable to previously published results on the RM and WSJ corpora. Training is based on Viterbi. The features are based on 13-dimensional MFCCs; we show experiments either with delta and acceleration features, or processing with splicing 9 adjacent frames together and doing LDA to 40 dimensions. 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 time or test, we always exclude the statistics corresponding to silence.

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 [18]. 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.

TABLE I
BASELINE %WERS, UNADAPTED

Features	System ID	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

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

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 statics as used to estimate the warp factor, and it is integrated into the warp factor calculation in

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

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

VTLN type	CMLLR type	System ID	WSJ RM	WSJ Nov'92	WSJ Nov'93
Adapting per speaker					
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

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.

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

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. Something that might be relevant is as follows: we had previously noticed, on another setup, that if we ignored the log-determinant then when using LDA+MLLT features, the linear VTLN training process would fail, with warp factors all going to one end of the scale. The message we take home from this is that one strays from the path dictated by the mathematics at one's own peril: that is, when one optimizes an objective function that does not make sense, even if it seems to work on one setup, one should not be surprised if it fails elsewhere.

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).

D. Results with Constrained MLLR

TABLE IV
APPLYING CONSTRAINED MLLR AFTER TRANSFORMS (PER SPEAKER): %WERS

Base Feats	Training type /system id	Adaptation type First	Adaptation type Second	RM	WSJ Nov'92	WSJ Nov'93
$\Delta + \Delta\Delta$	Unadapted/tri2a	Diag	Full	3.6	11.5	15.7
$\Delta + \Delta\Delta$	Unadapted/tri2a	Full	-	3.6	11.4	15.5
$\Delta + \Delta\Delta$	ET/tri2b	ET/Diag	Full	3.1	10.6	13.9
$\Delta + \Delta\Delta$	LVTLN/tri2g	LVTLN/Diag	Full	3.1	10.3	15.4
LDA	ET/tri2k	ET	Full	2.6	10.0	13.7
LDA+STC	LVTLN/tri2m	LVTLN/Diag	Full	2.8	10.1	13.7
LDA+STC	CMLLR (SAT)/tri2l	Full	-	2.7	9.6	13.7

Table IV concerns the combination of various VTLN methods with CMLLR. The setup is generally that we do a

speaker-independent pass with the "alignment models" we mentioned; then estimate an ET or LVTLN transform which includes a diagonal CMLLR component; then redecode with the ET/LVTLN-trained model; then estimate a full CMLLR transform on top of the ET or LVTLN transform. For comparisons, we also test with models trained in a speaker independent manner ("Unadapted"), and with SAT. For rows with "-" indicated as the second adaptation type, this means we did only two decoding passes and a single iteration of adaptation. It seemed plausible to us that there might be an inherent advantage in first doing a simpler adaptation type and then re-decoding and using this as supervision for more advanced type of adaptation; however, the top two rows of this table to not support this notion. The absolute best results are obtained with Speaker Adapted Training (SAT), but as is clear from Table III, this does not work well if we are only able to adapt per utterance. The general picture is similar to what we saw without CMLLR, i.e. in some conditions LVTLN is better and in some conditions ET is better.

VI. Conclusions and future work

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, rather than using 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, which we find to be the best type of features. For us, the most compelling advantage is that it is a relatively 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 more easily 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). In fact, when we compare these methods with SAT, we get the best results from SAT as long as we are adapting per speaker rather than per utterance (that is, as long as we are adapting on enough data). This does not invalidate the usefulness of ET, because ET still allows us to adapt on smaller amounts of data. However, we do question whether VTLN or its subsitutes such as ET are really necessary as long as there is enough data to adapt on; and even if there is very little data to adapt on, it is possible that other methods such as our previously published basis method for CMLLR adaptation [19] could solve the same problem that VTLN is solving.

Future work which we would like to do includes comparing this method with [19]. Since both are specially constrained forms of CMLLR adaptation, it makes sense to compare them (we have not done so because that method is not yet implemented in Kaldi). Another very natural extension is to consider subspaces of dimension higher than one in the log domain; essentially this becomes a log-domain version of [19], and our intuition is that this type of method would probably perform slightly better in the log domain.

References

[1] D. Povey, G. Zweig, and A. Acero, "Speaker Adaptation with an Exponential Transform," in *Proc. ASRU (submitted)*, 2011.

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

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

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

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

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

[7] J. McDonough, W. Byrne, and X. Luo, "Speaker normalization with all-pass transforms," in *Fifth International Conference on Spoken Language Processing*, 1998.

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

[9] 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.

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

[11] 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.

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

[13] 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.

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

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

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

[18] G. Saon, M. Padmanabhan, R. Gopinath, and S. Chen, "Maximum likelihood discriminant feature spaces," in *Proc. ICASSP*, 2000, pp. 1129–1132.

[19] D. Povey and K. Yao, "A Basis Representation of Constrained MLLR Transforms for Robust Adaptation," *Computer Speech and Language*, 2011.