Training of Reduced-Rank Linear Transformations for Multi-layer Polynomial Acoustic Features for Speech Recognition

Muhammad Ali Tahir¹, Heyun Huang², Albert Zeyer¹, Ralf Schlüter¹, Hermann Ney¹,³

¹ Human Language Technology and Pattern Recognition, Computer Science Department, RWTH Aachen University, Germany
² Centre for Language and Speech Technology, Radboud University Nijmegen, the Netherlands
³ Spoken Language Processing Group, LIMSI CNRS, Paris, France
{tahir,zeyer,schueter,ney}@cs.rwth-aachen.de, h.huang@let.ru.nl

Abstract

The use of higher-order polynomial acoustic features can improve the performance of automatic speech recognition (ASR). However, dimensionality of polynomial representation can be prohibitively large, making acoustic model training using polynomial features infeasible for large vocabulary ASR systems. This paper presents a multi-layer polynomial training framework for acoustic modeling, which recursively expands the acoustic features into their second-order polynomial feature space. After each expansion the dimensionality of resultant features is reduced by a linear transformation. Experimental results obtained for two large-vocabulary continuous speech recognition tasks show that the proposed method outperforms conventional mixture models. More recently the acoustic modelling community has shifted its focus to deep neural networks. We also train multi-layer polynomial features in a similar way: allowing backpropagation and using mean-normalized stochastic gradient descent algorithm. This has led to encouraging results. Specifically, appending a sigmoid-based feed-forward deep neural network with a final polynomial layer has resulted in significant word error rate improvement.

Keywords:
polynomial features, log-linear model, linear transformation, deep neural network
1. Introduction

Automatic Speech recognition (ASR) has come a long way since the early days of small numerical digit and isolated word recognition tasks with small vocabulary size. Recent systems can recognize spontaneous speech with noisy conditions and speaker variability with low error rates [Hinton & Deng+ 12, Zeyer & Doetsch+ 17, Sak & Senior+ 15]. Acoustic modeling is an integral part of such a system, which provides the matching of spoken audio to language phonemes. The classical paradigm for acoustic modeling has been Gaussian mixture models, with Hidden Markov Models [Baker 75] to model the time-scale non-linearity of input speech vectors. Input to a speech recognition system is a stream of acoustic feature vectors, e.g. Mel-frequency cepstral coefficients (MFCC) [Mermelstein 76]. To take acoustic context into account, a fixed window of consecutive feature vectors are concatenated into a larger vector. Such a vector naturally contains a lot of redundant information, due to similar vectors being appended together. A linear transformation such as linear discriminant analysis (LDA) [Häb-Umbach & Ney 92] is applied to this vector, for dimension reduction. LDA makes a maximum likelihood (ML) assumption, similar to what is used for an ML-trained GMM acoustic model [Rabiner 89]. Discriminative training methods attempt to maximize the probability of the correct phonetic classes with respect to competing classes. Experimentally, discriminative training has consistently outperformed ML training for speech recognition tasks [Bahl & Brown+ 86, Valtchev & Odell+ 97]. This reasoning is a motivating factor to consider training linear feature transformations using the discriminative training approach. There have been several works where a linear transformation has been discriminatively trained [Macherey 98, Omer & Hasegawa-Johnson 03, Povey & Kingsbury+ 05].

The motivation for using polynomial features for speech recognition stems from support vector machines (SVM) where a polynomial kernel may be used to project feature vectors to a higher space [Burges 98]. Classes which are not linearly separable may become separable in that higher dimensional space, by fitting a hyperplane between them. High-dimensional polynomial features are a promising application for training of linear transformations. Polynomial features are computationally expensive because the number of dimensions grows exponentially with polynomial order. Dimension reduction after each squaring of features is a way to use polynomial features while keeping them computationally tractable. This work investigates discriminatively trained
dimension reducing transforms for polynomial features.

There is a recent surge in interest related to deep learning and deep neural network architectures; as they have consistently outperformed traditional maximum likelihood or discriminative training [Schmidhuber 11, Hinton 07]. This is true for speech recognition [Hinton & Deng+ 12] as well as other pattern recognition application areas. The first improvements came from feed-forward deep neural networks [Hinton & Deng+ 12]. Currently, neural network based acoustic model training is a very dynamic area of research, with new architectures and approaches being presented constantly. Some of these approaches are recurrent neural networks, specifically long short-term memory (LSTM) architectures [Hochreiter & Schmidhuber 97]. Different variations have been proposed in literature such as peephole connections [Gers & Schmidhuber 01], highway networks [Srivastava & Greff+ 15] and sum-product networks [Gens & Domingos 12]. Based on this motivation, we have also explored the use of polynomial features in a deep multi-hidden-layer fashion. Our polynomial features’ network topology is related to highway networks and peephole connections; as seen in Equation (5) each layer gets a combination of linear and non-linearly transformed versions of last layer’s output. Furthermore, the polynomial function is same as the one used in sum-product networks. Therefore our work can be considered as one of the early endeavours in these directions [Tahir & Schlüter+ 11b, Tahir & Huang+ 13].

The rest of the paper is organized as follows: Section 2 gives an overview of log-linear acoustic models and use of polynomial features with log-linear models. Results and comparison are presented with GMM based discriminatively trained system. Section 3 provides a brief introduction of deep neural networks and the proposed multi-layer deep polynomial networks. Results are presented which show that a combination of polynomial and sigmoid based DNN’s can bring word error rate (WER) improvement. Finally, in Section 4 some conclusions are discussed, along with possible future work directions.

2. Training of Polynomial Features with Log-linear Acoustic Model

2.1. Log-linear Acoustic Model

Log-linear models [Hifny & Renals+ 05, Macherey & Ney 03] are an alternative to the Gaussian distributions, for representing emission probabili-
ties of HMM states. Under assumption of a pooled covariance matrix, posterior probabilities of Gaussian single density HMMs are equivalent to corresponding log-linear models’ posterior probabilities [Heigold & Schlüter + 07]. For speech input features \( x \) and classes \( s \) (HMM-states), the posterior probabilities of a log-linear acoustic model are given as

\[
p_{\theta}(s|x) = \frac{\exp(\lambda_s^T x + \alpha_s)}{\sum_{s'} \exp(\lambda_{s'}^T x + \alpha_{s'})}
\]  

(1)

in which \( \theta = \{\theta_s\} = \{\lambda_s, \alpha_s\} \) are parameters of log-linear model. The equivalence of Gaussian and log-linear posterior probabilities can also be extended to Gaussian mixture models. Due to hidden variables, this mixture model’s optimization is not a convex problem, but in principle local convergence can be guaranteed [Heigold & Ney + 13]. The log-linear acoustic model can be discriminatively split into a log-linear mixture model [Tahir & Schlüter + 11].

Log-linear model parameters are estimated by optimizing the frame-level maximum mutual information (frame-MMI) objective function

\[
F^{(frame)}(\theta) = -\tau_\theta ||\theta||^2 + \sum_{r=1}^{R} \sum_{t=1}^{T_r} w_{s_t} \log p_{\theta}(s_t|x_t)
\]

(2)

for a fixed alignment \( s_t^T \). \( \tau_\theta \) is a regularization parameter. \( w_s \) are state weights which could be tuned to give less weight to accumulations of e.g. noise and silence states. \( \hat{\alpha}_s = \alpha_s + \log p(s) \), \( p(s) \) is the prior probability of state \( s \) and \( R \) is the total number of training segments.

2.2. Log-linear Training of Linear Feature Transformation

In Section 2.1, the log-linear acoustic model was defined. Let the input to this acoustic model be \( Ax \) where \( x \) are the input features as before and \( A \) is a transformation matrix.

\[
p_{\theta,A}(s|x) = \frac{\exp(\lambda_s^T Ax + \alpha_s)}{\sum_{s'} \exp(\lambda_{s'}^T Ax + \alpha_{s'})}
\]

(3)

If the state-specific parameters \( \lambda_s \) are held constant, then the objective function of Equation (2) becomes convex with respect to optimization of the linear transform. Therefore, the dimension reducing linear feature transform can be reliably estimated by frame-level discriminative training of a log-linear
It has been shown [Tahir & Heigold 09] that if the feature transformation matrix is trained via log-linear discriminative training, it results in slightly better WER than using LDA-based transform. Furthermore, it has also been shown that speaker specific transformation matrices can be trained in the same way [Lööf & Schlüter 07], and result in improvement over CMLLR-based transformation matrices.

In this work, log-linear training of linear transformations for high dimensional polynomial features is investigated. This technique can well be applied to other approaches to generate high dimensional features.

### 2.3. Dimension-reduced Higher-order Polynomial Features

The input features (or LDA-transformed features) can be cross-multiplied with themselves and then vectorized to create second-order polynomial features. Thus an $n$-dimensional feature vector becomes $n \times n$ feature vector after this squaring. However, almost half of the elements in this squared vector are duplicate values, and after removing these duplicate elements the number of elements is $\frac{n(n+1)}{2}$. As we have seen in Section 2.1, a Gaussian acoustic model with a pooled covariance matrix can be simplified by cancelling the squared feature terms. The same is true for an equivalent log-linear model.
However, by using squared features we can implicitly represent the same type of information as a class-specific covariance based Gaussian/log-linear model. This squaring of features represents a non-linear transformation of input features into a much higher dimensional space, where the classes are expected to be more easily separable.

In previous work, second order polynomial features have resulted in WER improvements over the original MFCC features [Wiesler & Nußbaum+09]. These features create a large feature vector and hence greatly increase the number of acoustic model parameters; but their WER is as good as a full mixture system with even larger number of parameters. Therefore these squared features are more parameter efficient than the corresponding mixture density system.

Polynomial features are feasible for second order e.g. for 45 dimensional input features the size of squared features is 1035. After appending the original 45 dimensional vector its size becomes 1080. However, for higher orders like fourth or eighth order, their practicality is limited by the fact that going to higher orders increases the number of dimensions exponentially. Therefore, for second order features of 1080 dimensions, fourth order polynomial features would have 585K dimensions. This is about 50 times as much parameters as the full mixture acoustic model. To reduce the number of parameters to a computationally tractable size, dimension-reducing log-linear transformations can be applied to polynomial features after each squaring, thus allowing us to go to fourth-order, eighth-order polynomials and so on. At each layer \( k \) with input vector \( x^{(k-1)} \in \mathbb{R}^{r_{k-1}} \) and output vector \( x^k \in \mathbb{R}^{r_k} \), a transformation \( A^{(k)} \in \mathbb{R}^{r_k \times d_k} \) is trained:

\[
d_k = r_{k-1} + \frac{r_{k-1}(r_{k-1} + 1)}{2}
\]

\[
x^{(k)} = A^{(k)} \begin{bmatrix} \text{vech}(x^{(k-1)}x^{(k-1)\top}) \\ \vdots \\ x^{(k-1)} \end{bmatrix}
\]

where \( A^{(k)} \) is the dimension-reducing matrix to be trained in the current layer. For a symmetric matrix, the \( \text{vech}(\cdot) \) operator denotes half vectorization. It is a vector of length \( n(n+1)/2 \) obtained by vectorizing only the lower triangular part of \( x^{(k-1)}x^{(k-1)\top} \). Vectorizing is defined as a column vector obtained by stacking the columns of a matrix on top of one another.
As seen in Equation (5), at each layer $k$ the input features are also concatenated with squared features to retain the information of original features [Tahir & Huang 13]. The posterior probabilities are:

$$ p_{\theta}(s | x^{(k)}) = \frac{\exp\left(\lambda^{(k)}_s x^{(k)} + \alpha_s\right)}{\sum_{s'} \exp\left(\lambda^{(k)}_{s'} x^{(k)} + \alpha_{s'}\right)} $$

A pictorial representation of multilayer polynomial features can be seen in Figure 1. It is noteworthy that this is like a highway neural network [Srivastava & Greff 15]; albeit with polynomial non-linearity instead of sigmoid. The upper part of RHS in Equation (5) can be seen as the transform gate $T$ and lower part as carry gate $C$ of highway networks. Like peep-hole connections of LSTM, this uses a linear activation function but in a feed-forward setting.

**Training Procedure**

Training for polynomial features is done in a layer-by-layer fashion. First the original LDA transformed input features are appended with its squared features and a linear dimension reduction is trained log-linearly. Using the output of this linear transformation as input features for the next layer, the same process is repeated and another dimension reduction is trained. This process is repeated until the desired number of layers is achieved, or derivatives become so small that no improvement in objective function is observed. For optimization of the frame-MMI objective function, the RPROP algorithm [Riedmiller & Braun 93] is used.

Log-linear training of transformation depends crucially on reasonable initial values of the projection matrix and log-linear weights $A^{(k)}$ and $\lambda^{(k)}_s$ respectively. $k0$ denotes initial values of parameters for layer $k$. One reason why good initial values are crucial is the fact that the large scale of ASR training data makes it likely that the algorithm will suffer from a slow convergence to a (local) maximum. Assuming $r_k = r_{k-1}$, the parameters $A^{(k)} \in \mathbb{R}^{r_k \times d_k}$, $\lambda^{(k)}_s \in \mathbb{R}^{r_k}$ and $\alpha^{(k)}_s \in \mathbb{R}$ are initialized by setting $A^{(k)} = \left[0_{r_k \times (d_k - r_k)}; \mathbf{I}_{r_k \times r_k}\right]$, $\lambda^{(k)}_s = \lambda^{(k-1)}_s$ and $\alpha^{(k)}_s = \alpha^{(k-1)}_s$ respectively. This initialization guarantees that the MMI objective function value of at the beginning of new layer’s training is exactly identical to that at the end of previous layer’s training:
\[ \lambda_s^{(k0)} \left[ \begin{array}{c} 0_{r_k \times (d_k - r_k)} \\ I_{r_k \times r_k} \end{array} \right] \left[ \begin{array}{c} \text{vech}(x^{(k-1)}x^{(k-1)\top}) \\ x^{(k-1)} \end{array} \right] = \lambda_s^{(k-1)\top} x^{(k-1)} \quad (7) \]

Since the iterations continuously absorb more discriminative information from higher-order polynomials, the low-dimensional reduced vector might not be able to capture such additional information and need to be enlarged. Below the initialization formulas are shown for the case where \( r_k = 2r_{k-1} \)

\[ A^{(k0)} = \left[ \begin{array}{c} 0_{r_k \times (d_k - r_k)} \\ I_{r_k \times r_k} \\ 0_{r_k \times (d_k - r_k)} \\ 0_{r_k \times r_k} \end{array} \right] \quad (8) \]

Accordingly, \( \lambda_s^{(k0)} \) is also augmented by its copy, which results in

\[ \lambda_s^{(k0)} = \left[ \begin{array}{c} \lambda_s^{(k0)} \\ \lambda_s^{(k0)} \end{array} \right] \quad (9) \]

It can be seen that \( \lambda_s^{(k0)\top} A^{(k0)} = \lambda_s^{(k0)\top} A^{(k0)} \). Therefore, the additional rows can be expected to learn additional information when higher-order polynomials are considered.

2.4. Experiments and Results

Experiments have been performed on two large vocabulary ASR tasks. The first corpus, **European Parliament Plenary Sessions (EPPS) English** consists of mostly planned (non-spontaneous) speeches of European Parliament under clean conditions. It is part of the TC-STAR project [Lööf & Gollan + 07]. One source of variability is the presence of a number of non-native speakers. The input audio is sampled at 16 KHz and initial input features are 16 MFCC features with VTLN warping plus an energy and a voiced feature. These features are concatenated together in a window of 9 frames (−4 to +4) then transformed by LDA to a 45 dimensional vector. The lexicon consists of 60K words and the language model is a 4-gram true-case model trained on 400M running words. A triphone-based classification and regression tree is computed which clusters allophones into 4501 classes including silence. The baseline acoustic model is a Gaussian mixture density model with 256 densities per CART state with a WER of 16.4%. Apart from silence, the phoneme set also includes some non-voice phonemes like hesitation, noise, breath. The acoustic training data is 90 hours of audio (with
about 40% silence ratio). Development and evaluation corpora are about 3 hours each.

The second speech recognition task for these experiments is **QUAERO English 50 hours corpus**. The British English task consists of news broadcasts, debates, interviews etc. [Sundermeyer & Nußbaum-Thom+ 11]. Speaking style ranges from planned for broadcast news to conversational for interviews. Audio is sampled at 16 KHz but some recordings contain telephone calls which are band-limited to 4 KHz. There are also some instances of multiple simultaneous speakers and background music/noise. For the experiments in this paper, a 50 hour subset of the total QUAERO training data is used. The input features for the baseline setup are 16 MFCC features with VTLN warping. 9 of such consecutive frames are concatenated together and LDA-transformed to 45 dimensions. The lexicon contains 325K words and the language model is trained on 2G running words. There are 4501 CART states. For the MLP experiments in Section 3.3 the input features are slightly different from the baseline setup. Instead of 9 MFCC frames, 17 consecutive MFCC frames are concatenated together and there is no LDA transformation after that. This feature arrangement was found to be better in terms of WER. The development and evaluation corpora (eval10 and eval11 respectively) are 3 hours each.

Table 1: QUAERO English 50h; WER for mixture densities, polynomial features, and a combination of both

<table>
<thead>
<tr>
<th>feature polynomial order</th>
<th>dim. reduction</th>
<th>densities per state</th>
<th>final feature dimension</th>
<th>no. of params. × 1000</th>
<th>WER (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>no</td>
<td>1</td>
<td>45</td>
<td>214</td>
<td>35.4 35.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>128</td>
<td></td>
<td>26508</td>
<td>23.9 30.9</td>
</tr>
<tr>
<td>2nd</td>
<td>no</td>
<td>1</td>
<td>1080</td>
<td>4872</td>
<td>24.2 31.8</td>
</tr>
<tr>
<td>2nd</td>
<td>yes</td>
<td>1</td>
<td>90</td>
<td>467</td>
<td>28.5 35.9</td>
</tr>
<tr>
<td>4th</td>
<td></td>
<td>1</td>
<td>135</td>
<td>721</td>
<td>27.2 34.4</td>
</tr>
<tr>
<td>8th</td>
<td></td>
<td>1</td>
<td>180</td>
<td>967</td>
<td>27.0 34.1</td>
</tr>
<tr>
<td>4th</td>
<td>yes</td>
<td>32</td>
<td>135</td>
<td>19688</td>
<td>23.1 30.2</td>
</tr>
</tbody>
</table>

Figure 2 shows some results for dimension-reduced higher-order polynomial features on EPPS English task. The input features are $16 \times 9$ MFCC features which have been LDA transformed to 45 dimensions. The results
Figure 2: EPPS dev2007: WER(\%) vs. No. of parameters for higher-order polynomial features in comparison with mixture densities
Figure 3: QUAERO English 50h; WER for eval set; for mixtures, higher order features, and a combination of both

compare the full second-order polynomial features with dimension reduced second, fourth, eighth and sixteenth-order polynomial features. It is also compared with the case where non-linearity is modelled by conventional mixtures. Note that the features/models for all rows in the graph (except the first line) have been trained using the same training criterion: frame-level MMI. It can be seen that the number of parameters required for 8th order polynomial features is less than those for mixtures with 4 densities per state; but the WER for the former case is lower than the latter. This gives the indication that modelling the feature non-linearity by polynomial representation required less parameters than doing it through the use of mixtures. For larger number of parameters, the WER of polynomial features starts to converge. Nevertheless, as we shall see in the following experiments on QUAERO task; a combination of polynomial features with mixture densities brings lesser WER than using either of them alone.

Another point to note from Figure 2 is the effect of number of output dimensions of dimension reducing transform. For 8th-order features, using 45 output dimensions gives a WER of 19.0%; while increasing dimensions to 90
decreases the WER to 18.2%. This shows that as the order of the polynomial features gets larger, it is beneficial to use projective transformation matrices with larger number of rows.

Similar experiments have been performed for the acoustically more difficult QUAERO English 50 hours task. These results have been summarized in Table 1 and Figure 3. One additional aspect here is the result in the last row, which combines polynomial features with mixture-based acoustic model. For the original mixture density model with MFCC features (4th row), the best WER is 30.9% with 128 densities per CART state. Going to higher number of densities per state causes the model to overtrain, because the training corpus is relatively small (50 hours). If mixture densities are trained on top of 4th order polynomial features as in the last row, WER drops to 30.2% with just 32 densities per state, because of better available input features. This result shows that the results obtained by dimension-reduced polynomial features with log-linear mixtures are significantly better than either full-rank polynomial features, reduced-rank polynomial features or log-linear mixtures alone.

3. Training of Polynomial Features within a Deep Neural Network

3.1. Deep Neural Networks

Deep neural networks (DNNs) have become an important tool for creating probabilistic features for speech recognition. A neural network for speech recognition consists of a multilayer-perceptron (MLP), having non-linear activation functions in the hidden and output layers. Some earlier works exploring the use of MLPs for speech recognition are [Peeling & Moore 86, Bourlard & Wellekens 87, Waibel & Hanazawa 89]. These were complex systems aiming to model the whole speech recognition process by neural networks, but were not able to outperform the GMM-HMM based approaches. More recently, there are two ways of applying neural networks for acoustic modelling: hybrid and tandem MLPs.

- A **hybrid** MLP system [Bourlard & Morgan 93, Seide & Gang 11, Hinton & Deng 12] directly uses posterior probabilities of MLP network as acoustic model probabilities. The probabilities correspond to clustered allophone (CART) states.

- A **tandem** MLP system [Hermansky & Ellis 00] has a bottleneck layer as the last output layer of the network and then a regular GMM
based maximum likelihood acoustic model is trained on top of it. This makes it easier to use GMM based concepts for optimization such as linear discriminant analysis (LDA) and speaker adaptation.

A neural network is a set of neurons linked together by weighted connections. A neuron’s input activation $z_j$ is a weighted combination of outputs of nodes in the previous layer $x_i$, plus a bias constant $\alpha_j$. The output activation $y_j$ of the node $j$ is a non-linear transformation applied to the node input.

$$z_j = \sum_i \lambda_{i,j} \cdot x_i + \alpha_j \quad ; \quad y_j = \sigma(z_j)$$ (10)

where $\{\lambda_{i,j}\}$ are parameters of neural network. The sigmoid activation function is used (among others) for hidden layers of ANNs

$$y_j = \frac{1}{1 + e^{-z_j}}$$ (11)

For the last output layer of the network, a normalized softmax activation function is used because the outputs are to be interpreted as probabilities

$$y_j = \frac{e^{z_j}}{\sum_i e^{z_i}}$$ (12)

In case a layer has identity activation i.e. $y_j = z_j$, it is called a linear layer. For all the MLP experiments in this section, the sigmoid activation is used for hidden layers and softmax for the output layer neurons. For the linear (bottleneck) layers, the activation is identity. Figure 4 shows an example of an MLP with an input, one hidden and an output layer. For MLP training for classification task of inputs $x_n$ belonging to classes $c_n$, the cross-entropy criterion can be optimized.

$$E_n = -\sum_{k=1}^{\kappa} \delta(k, c_n) \log(y_k)$$ (13)

The error of the last layer is back-propagated through the network, based on connections of each current node to each node in the previous layer. When the errors of all the nodes in all layers are known, the weights of connections can be updated based on error gradients. Details of this process and derivations can be found in [Haykin 98, Plahl 14].
A deep neural network refers to an MLP with several non-linear hidden layers, as many as six or more. Use of deep neural network has become state-of-the-art for acoustic modelling in the last few years [Hinton & Deng+ 12, Dahl & Deng+ 12, Sainath & Kingsbury+ 11].

Transition from Log-linear Models to Deep Neural Networks
We can see that Equation (11) for softmax layer is same as Equation (1) for posterior probability of log-linear acoustic model. This is the path that we have followed in the course of this work, and it explains the corresponding results while going on the equivalence path from GMM $\rightarrow$ Log-Linear Models $\rightarrow$ Softmax layer. As we add hidden layers to the DNN, its classification performance extends beyond that of a softmax layer only. Furthermore, the cross entropy objective function of DNN optimization in Equation (13) is same as that of frame-MMI in Equation (2). Tandem deep neural networks have explored the usefulness of log-linear mixture models with deep neural network based features [Tüske & Michel+ 17, Tüske & Tahir+ 15, Tüske & Golik+ 15].

3.2. Linear bottlenecks for DNNs

Two consecutive layers of an MLP network that are fully connected may have redundancy in the structure. Many of the elements in a layer may have a negligibly small effect on the output of that layer. If the number of elements in a layer can be reduced by removing those redundancies while not compromising classification performance, it can provide large decreases in
time and memory requirements of MLP training. Several methods have been proposed to achieve this compression. [Yu & Seide+ 12] have reduced the number of elements in the layers by removing the close to zero elements and converting the matrices to an index-based representation. [Xue & Li+ 13] have factored the weight matrix into a product of two smaller matrices, providing parameter compression. They have reported encouraging results by doing a singular value decomposition (SVD) based factorization between the hidden layers. The error rate degrades at first but after doing a full network training with back-propagation, the classification performance of the MLP network is restored. [Wiesler & Richard+ 14] have proposed a training mechanism whereby a hidden layer and its low-rank factorization can be simultaneously trained from scratch. Apart from model parameter reduction, they report an added benefit of regularization from this factorization. Thus linear bottlenecks can reduce over-training of MLP network parameters.

3.3. Deep Polynomial Network

In this section the implementation of multilayer polynomial features as a deep network is explored. To keep in sync with our DNN baseline, there are some differences to the log-linear training method of Section 2.3, as listed below.

- The previous section’s polynomial layers were trained in forward direction only. Here we shall allow backpropagation of error.
- Previous training was performed by iterating over the full corpus in each iteration (full batch). Here a stochastic algorithm is used (mini-batch).
- For input features, previously 9 MFCC vectors were windowed together and then transformed by linear discriminant analysis (LDA). Here, 17 MFCC vectors are windowed together without any LDA.

The speech corpus is the QUAERO English 50h corpus (Section 2.4). For the input MFCC features the feature vector length is 29, and 17 consecutive frames are appended along with first and second derivatives. The MLP network has 493 input features. The number of nodes in the output softmax layer is 4501 (no. of CART states). The training objective function is cross-entropy (which is equal to frame-MMI of Section 2.1). The algorithm used for training is mean-normalized stochastic gradient descent. There are six hidden
layers and the hidden layers have a sigmoid activation function. Detailed description of this system can be found in [Wiesler & Richard+ 14].

Training Procedure
Our deep polynomial network takes the same windowed MFCC features as mentioned in last paragraph. A linear dimension reduction reduces these 493 dimensional features to 128. These 128 dimensional features are then expanded to second-order polynomial and concatenated with itself, giving an 8,384 dimensional vector. This large vector is again reduced to 128 dimensions. This expansion and reduction is repeated two more times, so that a three hidden layer deep polynomial network is created. This represents 8th-order polynomial features. Going beyond 8th-order features was tried but the derivatives become so small that there is no tangible effect of an extra layer on objective function and WER.

Before training, the neural network is discriminatively pre-trained layer-by-layer. First, one hidden layer network is trained for a few epochs. Then keeping the first layer’s parameters constant, a second hidden layer is trained using the first hidden layer’s outputs as features, as so on. After pre-training, the full networked is trained with backpropagation. This is also described in [Wiesler & Richard+ 14].

Table 2 shows the results of these polynomial features in comparison to the sigmoid-based DNN baseline. The 3-layer polynomial network achieves a WER of 20.7%(dev). If the last softmax layer is split into a log-linear mixture model, then the WER is reduced to 19.9%. This is a large improvement of 4.0% absolute on the result for GMM trained by the same objective function. This result is close to WER of sigmoid-based DNN, although still worse. This nevertheless shows that polynomial expansion of features is a candidate to be considered for modelling non-linearity of features, just like a sigmoid activation function. It was also noticed that WER for different layers with and without backpropagation did not differ by more than 0.2% absolute. This shows that after layer-by-layer training of polynomial network to convergence, backpropagation is not as crucial as it is for e.g. sigmoid activation function.

3.4 Combination of polynomial with sigmoid-based DNN
Table 3 shows some results of combining a state-of-the-art deep neural network with polynomial features. The baseline DNN system is the same as in [Wiesler & Richard+ 14]. It has 6 sigmoid layers with a linear bottleneck of
Table 2: Training corpus: QUAERO 50h. Comparison of a deep polynomial network with a state-of-the-art DNN (sigmoid)

<table>
<thead>
<tr>
<th>network type</th>
<th>mixtures</th>
<th>no. of layers</th>
<th>no. of params.</th>
<th>WER (%)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>frame-MMI GMM</td>
<td>yes</td>
<td>-</td>
<td>26M</td>
<td>23.9</td>
<td>30.9</td>
</tr>
<tr>
<td>sigmoid</td>
<td>no</td>
<td>6</td>
<td>7.9M</td>
<td>18.6</td>
<td>24.9</td>
</tr>
<tr>
<td>polynomial</td>
<td>no</td>
<td>1</td>
<td>1.7M</td>
<td>24.3</td>
<td>31.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>2.8M</td>
<td>21.2</td>
<td>27.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>3.9M</td>
<td>20.7</td>
<td>27.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>5.0M</td>
<td>20.6</td>
<td>27.0</td>
</tr>
<tr>
<td>polynomial</td>
<td>yes</td>
<td>3</td>
<td>22M</td>
<td>19.9</td>
<td>26.3</td>
</tr>
</tbody>
</table>

Table 3: Training corpus: QUAERO English 50h. Appending a state-of-the-art DNN (sigmoid with cross-entropy) with a last polynomial layer

<table>
<thead>
<tr>
<th>model type</th>
<th>polynomial features</th>
<th>additional MPE training</th>
<th>WER (%)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN</td>
<td>no</td>
<td>no</td>
<td>18.6</td>
<td>24.9</td>
</tr>
<tr>
<td>DNN</td>
<td>yes</td>
<td>no</td>
<td>18.2</td>
<td>24.4</td>
</tr>
<tr>
<td>DNN</td>
<td>yes</td>
<td>yes</td>
<td>18.0</td>
<td>24.1</td>
</tr>
</tbody>
</table>

256 nodes after each sigmoid layer. The network is trained with cross-entropy criterion, which is the same as frame-MMI for log-linear training. The feature input for the polynomial layer is the output of the last linear bottleneck. It can be seen that a combination of DNN features and a polynomial layer (3rd row in table) improves the WER by 0.5% absolute (eval). If the last layer is trained by MPE criterion (4th row in table), then with polynomial features it achieves further WER reduction of 0.2%. Some internal experiments on this corpus at our institute show that adding a 7th or 8th sigmoid layer to the baseline DNN does not bring any WER improvement. Thus the WER improvement by adding the polynomial layer can be genuinely attributed to its difference to the sigmoid layers.

For an explanation of WER improvement due to polynomial features, we can consider polynomial features’ equivalence to class-specific covariance Gaussian acoustic model. It can be shown that a log-linear model with a single polynomial layer as in Equation (5) is equivalent to a class-specific
covariance based Gaussian model; due to squaring of input features. This is in contrary to a regular log-linear model or softmax, which is equivalent to a pooled covariance based Gaussian model. Therefore, if a polynomial hidden layer is used before the last softmax, it allows an extra degree of freedom due to its equivalence with class-specific covariance model. This in turn allows the DNN to achieve better classification performance and hence lower WER.

4. Conclusion and Outlook

In this paper, we presented a multi-layer linear transformation training framework, which can harness crucial information from higher-order ($\geq 3$) polynomial feature space. The framework allows us to train log-linear models in the otherwise prohibitively high-dimensional feature space spanned by higher-order polynomials. By repeating second-order polynomial expansion $n$ times, $2^n$-order polynomial features can be obtained. Following each squaring of polynomial features, the subsequent linear projection limits the dimensionality of features. Experiments on EPPS and QUAERO corpora revealed that polynomial features allow us to significantly reduce the number of parameters. Due to recent popularity and good performance of deep neural networks for speech recognition, deep polynomial features were trained by using similar optimization algorithms and backpropagation. This work proposes a multi-layer polynomial network with highway/peephole connections. Experiments with DNN on QUAERO corpus showed that a combination of sigmoid and polynomial layers leads to WER improvement over the DNN baseline.

Future work in this direction could be further exploration of combinations of polynomial features with sigmoid or rectified linear unit (ReLU) activation functions. For example a DNN with alternating polynomial and sigmoid based layers. Furthermore, polynomial non-linearity can also be used in context of recurrent neural networks especially LSTM architecture.

5. Acknowledgements

The research of H. Huang was funded by the European Community’s Seventh Framework Programme [FP7/2007-2013] under grant agreement no. 213850 SCALE. The research leading to these results has received funding from the European Union Seventh Framework Programme EU-Bridge
(FP7/2007-2013) under grant agreement No. 287658. H. Ney was partially supported by a senior chair award from DIGITEO, a French research cluster in Ile-de-France.
References


[Häb-Umbach & Ney 92] R. Häb-Umbach, H. Ney: Linear discriminant analysis for improved large vocabulary continuous speech recognition, In


evaluation system for European English and Spanish, In Interspeech, pp. 2145-2148, Antwerp, Belgium. 2007.


S. Wiesler, M. Nußbaum, G. Heigold, R. Schlüter, H. Ney: Investigations on features for log-Linear acoustic models in


