Design and Learning of Output Representations for Speech Recognition

Li Deng
Microsoft Research,
Redmond, WA, USA
deng@microsoft.com

Abstract

In deep learning research, often the focus has been on the input feature representation while the output representation tends to receive much less attention. In this paper, three largely separate case studies are provided to argue for the importance of learning output representations. In these studies, three ways of designing and/or learning output representations for the deep-learning approach to speech recognition are discussed and analyzed. First, the very large number of output units in the current context-dependent (CD) deep neural net (DNN) based speech recognizers can be effectively reduced, without lowering recognition accuracy while improving decoding efficiency, by performing dimensionality reduction using low-rank approximation to large DNN output matrices. Second, the currently popular CD-DNN that uses "beads-on-a-string" or linear-sequence representations for linguistic speech units in the DNN output layer can be generalized to structured multi-linear or graph representations. Temporally overlapping linguistic "features" or symbols are used as a basis for such phonological design. Third, when a special type of deep networks, the deep convex network (DCN), is used as a representational model for speech acoustic patterns, the output units in each of the DCN modules are designed to be linear, enabling drastic simplification in learning the parameters of the entire network.

1. Introduction

In recent years, learning representations using deep models, notably those based on deep neural networks (DNNs), have largely focused on the sensory input data, such as speech and image [20][5][22] with visible successes most notably in industry-scale, real-world speech and image recognition tasks [24][6][7][8][19][23][26][29][31]. Learning output representations, however, has received relatively less attention. Nevertheless, in applications such as speech recognition, the linguistic units (e.g., sentences, phrases, words, syllables, phones etc.), which are the output of speech recognizers, have rich and complex structure and require more principled representations and learning than what are currently in use in most of the present deep learning based systems. In almost all state of the art, DNN-based speech recognition systems, the output representation inherits from the over 20-year-old concept of context-dependent (CD) phonetic states [25][15]. Like the traditional GMM-HMM systems, the new CD-DNN-HMM systems in current popular commercial use [5][19][31] all have the same "beads-on-a-string" or linear-sequence representation for the CD phonetic states. (In fact, the discovery that the use of a large number of CD phonetic states as the output layer of the DNN is highly effective prompted the fast industrial adoption of DNN technology in speech recognition, partly because this would involve minimal change in the run-time decoder algorithms and software [4][5].)

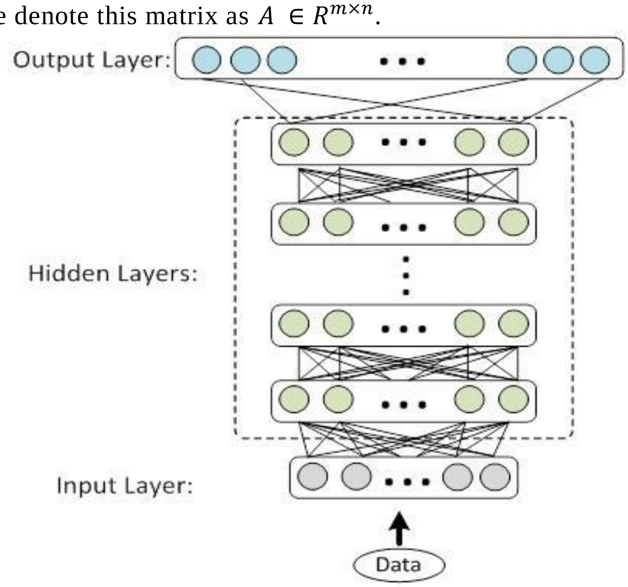
While the use of the CD phonetic states as the output representation for DNN-based acoustic models in speech recognition significantly reduces recognition errors and allows the decoding algorithm to remain to be of the Viterbi beam search style permitting fast pruning of hypotheses, it carries two main disadvantages. First, it introduces a very large number of DNN parameters in the final weight layer. This makes the online computation very costly and would limit the applications of the DNN in various scenarios. Second, the flat representation of speech units based on CD phonetic states coded on the DNN output layer discards the known phonological structure of speech. Putting such structure back to the deep models while improving their output representation holds the promise to further reduce speech recognition errors.

In the remaining part of this paper, some existing and proposed solutions to the two problems above pertaining to the limitations of the current DNN-based speech recognition technology in terms of its output representation are reviewed and discussed. Further, an additional problem arising from a specific type of deep network that has direct bearing on the output representation issue will be discussed.

2. Handling high dimensionality in the DNN's output representation

The first example of the benefit of better output representations in the DNN-based speech recognizers is provided in this section. As discussed in the introduction section, most current DNN systems use a high-dimensional output representation, each component corresponding to one CD phonetic state in the top-level HMM receiving the DNN output as its "features". In brief, by performing SVD-based dimensionality reduction on the DNN's high-dimensional output vectors, the decoding efficiency of running the recognizer in run time can be drastically improved due to the significantly reduced DNN parameters. We now discuss details of this technique based on the work published recently in [28][26].

In Fig. 1, we show the general DNN architecture with high-dimensional (m) output vectors indicated as blue nodes and with the subsequently large weight matrix at the top of the DNN. We denote this matrix as $A \in R^{m\times n}$.



$A \in R^{m\times n}$

Figure 1: The basic architecture of DNN, where output layer represents high-dimensional context-dependent (CD) phonetic states used for HMM decoding; courtesy of authors of [28].

To reduce dimensionality m or n, we perform low-rank matrix factorization via singular value decomposition (SVD) on matrix A:

$$A_{m\times n} = U_{m\times n} \textstyle\sum_{n\times n} V_{n\times n}^{T}$$

Then, the low rank-*k (k<n)* approximation to matrix *A* can be shown by the following steps:

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} = \begin{bmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & \ddots & \vdots \\ u_{m1} & \cdots & u_{mn} \end{bmatrix} \cdot \begin{bmatrix} \epsilon_{11} & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & \epsilon_{kk} & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & \epsilon_{nn} \end{bmatrix} \cdot \begin{bmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \vdots \\ v_{n1} & \cdots & v_{nn} \end{bmatrix}$$

$$\approx \begin{bmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & \ddots & \vdots \\ u_{m1} & \cdots & u_{mn} \end{bmatrix} \cdot \begin{bmatrix} \epsilon_{11} & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & \epsilon_{kk} & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 \end{bmatrix} \cdot \begin{bmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \vdots \\ v_{n1} & \cdots & v_{nn} \end{bmatrix}$$

$$= \begin{bmatrix} u_{11} & \cdots & u_{1k} \\ \vdots & \ddots & \vdots \\ u_{m1} & \cdots & u_{mk} \end{bmatrix} \cdot \begin{bmatrix} \epsilon_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \epsilon_{kk} \end{bmatrix} \cdot \begin{bmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \vdots \\ v_{k1} & \cdots & v_{kn} \end{bmatrix}$$

$$= \begin{bmatrix} u_{11} & \cdots & u_{1k} \\ \vdots & \ddots & \vdots \\ u_{m1} & \cdots & u_{mk} \end{bmatrix} \cdot \begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{k1} & \cdots & w_{kn} \end{bmatrix}$$

Thus the total number of weight parameters in the output matrix is reduced from $m \times n$ to $k \times (m + n)$. This seems to be a more principled way of constructing "bottleneck" layers or features than the earlier work reported in the literature (e.g. [11]).

In the implementation of the dimensionality reduction method reported in [28], the DNN with high-dimensional output layer was trained first. Next, SVD was performed on the large output matrix A, which is then approximated by a product of two much smaller matrices. When applying such approximation back to the DNN, the large single output layer's weight matrix A is converted to two layers both with smaller weight matrices $U \in R^{m\times k}$ and $W \in R^{k\times n}$. The lower layer is made linear and the upper one is nonlinear as in the original DNN. Finally, the DNN with reduced dimensionality in the output layer is re-trained. The experimental results reported in [28][26] both show no speech recognition accuracy reduction with the low-rank matrix approximation, while the run-time computation is significantly reduced.

3. Structured output representation for symbolic speech target sequences

The second case study presented here concerns structured design of the output representation for the symbolic or phonological units of speech. The rich phonological structure of symbolic nature in human speech has been well known for many years [2][3][16][18][13]. Likewise, it has also been well understood for a long time that the use of phonetic or its finer state sequences, even with contextual dependency, in engineering speech recognition systems is inadequate in representing such rich structure [5][12][14][27] and thus leaving a promising open direction to improve the speech recognition systems' performance. In this section, I survey the basic theories about the internal structure of speech sounds and their relevance to speech recognition technology in terms of the specification, design, and learning of possible output representations of the underlying speech model for speech target sequences that may be used in training speech recognizers.

3.1 From linear to nonlinear phonological representations

In the traditional phonology [2], a phoneme is represented as an unstructured set of phonological features, or feature bundles. Likewise, a sequence of phonemes is characterized by a sequence of feature bundles, resulting in a feature matrix which arranges the feature bundles into columns. The feature matrix does not concern how features might be organized or structured. Because phonemes as feature bundles in a word or in word sequences follow each other in strict succession, this feature-matrix approach is called the sequential linear model of phonology. In this regard, the speech units in the “linear” order are often likened to “beads on a string”, as commonly used in the pronunciation model component of modern speech recognizers including the most advanced DNN-based ones (see the discussion section in [5]).

While this sequential model of phonological representation is conceptually simple and analytically tractable, it has a number of serious inadequacies. All features are assumed to have a positive or negative value for every segment, regardless of whether such features could conceivably apply to that segment or not. Further, the phonological rules in linear phonology do not explain why some phonological processes are more natural than others.

The most important inadequacy of the linear phonological model is that it prevents features from extending over domains greater or lesser than one single phoneme, because each feature value can characterize only one phoneme and vice versa. This is contrary to ample phonological evidence that demonstrates “nonlinear” behavior, where strict sequential order is broken and one feature can occupy a domain significantly greater than a phoneme, or a domain less than a full phoneme. For example, the [+nasal] feature in some languages including English may occupy only a fraction of a segment, or it can spread across more than one segment or syllable. This type of inadequacy of the linear phonology model has been overcome by the theory of autosegmental phonology where the features that go beyond the segmental limits set by the linear model are extracted from feature matrices and are placed in separate, independent tiers of their own, hence the term. Autosegmental phonology establishes a “nonlinear” model of phonological representation, where the strict ``linear" order is replaced by a multi-tiered representation where feature elements in different tiers often do not follow the linear order but overlap with each other temporally.

The second inadequacy of the linear sequential model of phonological representation is its implicit assumption that feature bundles in the feature matrix have no internal structure; i.e. each feature is equally related to any other feature. This, again, is against a considerable amount of evidence that suggests that features are grouped into higher-level functional units. In many languages including English, all place features function together as a unit. To overcome this challenge, a tree-like model of feature organization is developed, where segments are represented in terms of hierarchically-organized node configurations with terminal nodes being the feature values and non-terminal nodes being the feature classes resulting from functional feature groupings. This tree-like feature organization, together with a set of general properties associated with the organization and related to phonological rules, is also called feature geometry [3]. Now, rather than putting features in matrices as in the traditional theory, the features are placed as terminal nodes in a tree-like diagram, where these terminal nodes are unordered and are on separate tiers depending on their parent feature classes. This organization permits nonlinear behavior of feature overlap, as in autosegmental phonology. It also permits strong constraints on the form and functioning of phonological rules. Feature geometry is a substantial extension of autosegmental phonology which we discuss next.

3.2 Phonological feature hierarchy

Phonological features are atomic, symbolic specification of the constituent units that make up of all phonemes of the world languages. They are hierarchically organized into a tree-like structure in feature geometry theory [3], forming the basis of internal organization of speech sounds. Compared with the traditional feature theory [2], feature geometry is more heavily grounded on articulators and their functional roles in producing speech sounds. Fig.2 is an illustration of the tree structure that is associated with each phone or segment in the phonological representation of speech.

In Fig. 2, the root node specifies the coherence of the global segmental properties. Popular proposals assign features [cons] and [son] in the root node. The tier below the root node contains a non-terminal node called the Laryngeal node. Three laryngeal features [spread] (i.e., spread glottis, SG), [constr] (or constricted glottis, CG), and [voice] (vc) are grouped under this node. Under the Place node, we have the place features [labial], [coronal] (together with its dependent features [anterior], [distributed], [strident]), and [dorsal] (together with its dependent features [high], [low], and [back]). These phonological features often spread (i.e. temporally overlap) as a unit. This spread is typically independent of other non-place features such as [cont], [voice], [nasal], etc. This regularity is naturally captured by grouping all these place features under a single place node as shown in the lower part of Fig. 2. The remaining features [lateral], [nasal], and [cont] do not form sub-groups within themselves or with other features. They are listed separately under the root node.

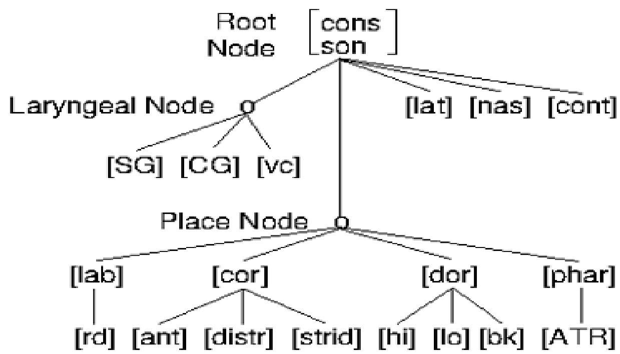


Figure 2: Illustration of feature geometry expressed as a tree-like structure.

When several segments form a sequence, each of which has its feature hierarchy as shown in Fig. 2, we obtain a three-dimensional picture where the feature hierarchy unfolds in time. While the root node dominates all features for each segment for a sequence of segments, all the individual root nodes are linked in a sequence as well. One example of the expanded feature geometry for a three-segment sequence, consisting of /p/, /i/, and /n/, is crafted in Fig. 3.

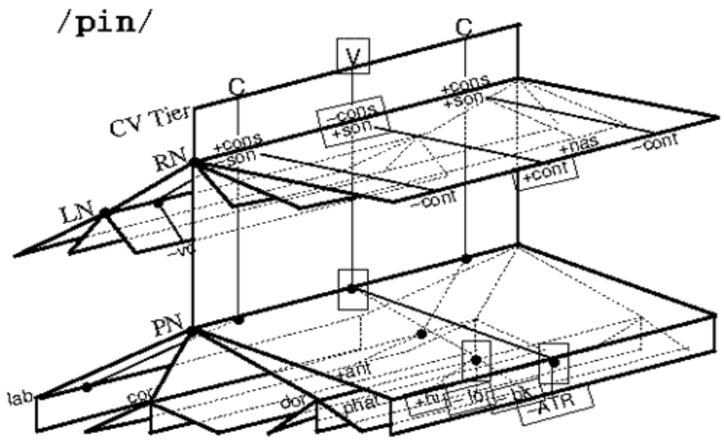


Figure 3: An example of expanded feature geometry for egment sequence /pin/.

3.3 A computational model for designing output representations of speech

Here we describe a computational model that makes use of the expanded feature geometry discussed above to construct structure output representations of symbolic speech target sequences. This model fixes all aspects of inadequacy of the linear "beads-in-a-string" model for target specification of speech units in a sequence that underlies all current speech recognition system including the DNN-HMM systems.

A series of "Computational Phonology" models detailed in Chapter 9 of [12] and sketched in [12] provided a basic framework for designing and learning the structured output representations of speech. The underlying theory of this framework follows "articulatory phonology," which links the expanded feature geometry to its phonetic "implementation" thereby providing a solid "interface" between symbolic phonology continuous-valued, measurable phonetic variables (e.g., articulatory movements and associated acoustic parameters).

Central to this framework is a set of empirically designed symbolic articulatory features with their respective temporal domains specified, permitting their asynchronous overlapping over time with constraints derived from phonetic knowledge as part of articulatory phonology. The articulatory features and their overlapping and constraining rules for complete American English are detailed in [14], [27], and [12].

The overlapping articulatory features are designed based on speech recognition considerations and on theories of phonology. From the theoretical side, they are a mix of, but different from, the (multi-valued) distinctive features in feature-geometry theory and the gestures in articulatory theory. Compared with the gestures, the articulatory features share the same key property of overlapping or blending across tiers. In fact, the tiers are essentially the same between the two representations. However, one very crucial difference is that unlike the gestures which are defined in terms of the parameters of the abstract dynamics in the "tract variables," the articulatory features are entirely symbolic with no specific reference and association to any continuous variables. A separate module, which is called the interface between phonology and phonetics and which can take many different forms, is responsible for the mapping from a symbolic articulatory feature to some continuous, phonetic variables (articulatory or vocal tract resonance variables in some practical implementations). The notion of this type of interface simply does not exist in articulatory phonology and in the gesture.

The second important difference between the gesture and the articulatory feature is that the former is associated with intrinsic duration due to its connection to an underlying, abstract task-dynamic system, while the latter is not. The temporal characterization of the articulatory features is only by way of relative timing among the tiers.

Further, the articulatory features are designed to be phonological units. That is, when serving as underlying units for describing word pronunciation, they play contrastive roles in distinguishing meanings of the words. To ensure this property, in the design of the articulatory features, phoneme-like units are used as the basis and it is made explicit that the different phonemes have distinctive values of the articulatory features associated with them [14]. From speech recognition considerations when that computational model was implemented many years ago using statistical generative models as the theoretical basis [14][27], the articulatory features were designed with the additional requirement of economy. The articulatory features, with their spatial and temporal structures in place, were used as the nonlinear atomic speech units for lexical representation. This was aimed at provide a superior alternative to the popular linear phonetic representation. An efficient lexical representation of this sort requires the choice of a small set of feature units so that in terms of these symbols each lexical item can be compactly specified while being made distinguishable from each other at the same time. In the modern days of big data and big compute, especially with the clear demonstration of superior performance of the big, discriminative DNN over the generative GMM [5][20], the requirement of economy in the construction of the articulatory features and imposition of their overlapping constraints may be reduced or eliminated. The entire framework deserves serious re-thinking and re-design.

3.4 Implementation detail and examples

To illustrate the computational model for designing structured output representations for potential use in speech recognition, we provide selected examples here. We use the articulatory feature design described in [14], where five multi-valued features, Lips, Tongue Blade (TB), Tongue Dorsum (TD), Velum, and Larynx, are assigned uniquely to each phonemic unit, with intended "minimal" redundancy and "maximal" separability. Then the major contextual variations in speech are modeled as a natural result of overlap or asynchronous spread of the "intrinsic" values of one or more of these features across adjacent phonetic units. Given a fixed feature-overlap pattern,

a one-to-one mapping is made from such a pattern to a state-transition graph which forms the topology of the underlying Markov chain of the HMM accomplishing such a mapping. The graph is constructed in such a way that each node in the graph (or the state in the HMM) represents a unique composition of the five features. Each individual lexical item is represented by a distinct state-transition graph, formed by concatenating a sequence of sub-graphs associated with the phone sequence in the phonetic transcription according to the feature-overlap patterns constructed specifically from these phones-in-context. Since each node in this global graph contains a distinct composition of the features, we can also view the representation of a lexical item described here as an organized set of feature-bundle collections.

In Fig. 4, the construction of overlapping patterns across five articulatory features is illustrated for English word *"strong",* together with its spectrogram. The spectrogram is time aligned with the overlapping patterns. Note the lip-rounding and nasalization features have variable (relative) durations, and they are represented by two dashed boxes. This type of variability in the duration of feature overlapping gives rise to alternative feature bundle sequences.

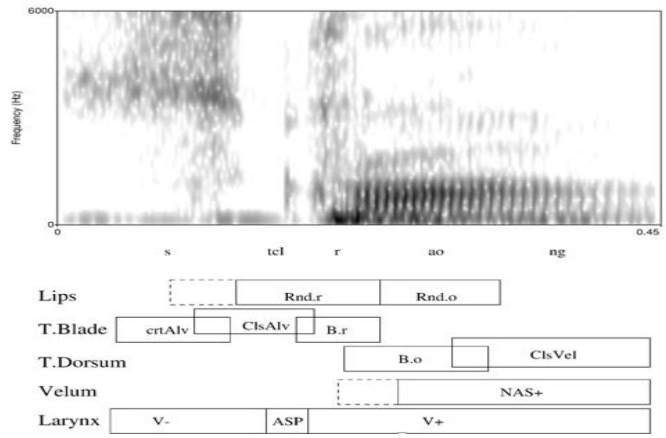


Figure 4: Phonologically defined articulatory feature overlaps for English word *"strong".*

By merging identical feature bundles, a "state transition" network can be constructed using a technique described on pages 314-315 of [13]. Each state in the network corresponds to a unique feature bundle. The network constructed by the overlapping feature bundle generator for the word "*strong*" in Fig. 4 is shown in Fig. 5, where each state is associated with a set of symbolic features. The branches in the network result from alternative overlapping durations specified in the feature overlapping rules. Note that the graph representation of the pronunciation network of English word *"strong"* is very different from the left-to-right linear-chain representation used on virtually all speech recognition systems (e.g. [5][20][25][15]). While both type of representations capture context dependency, the mechanism and capability of embedding phonetic context are very different.

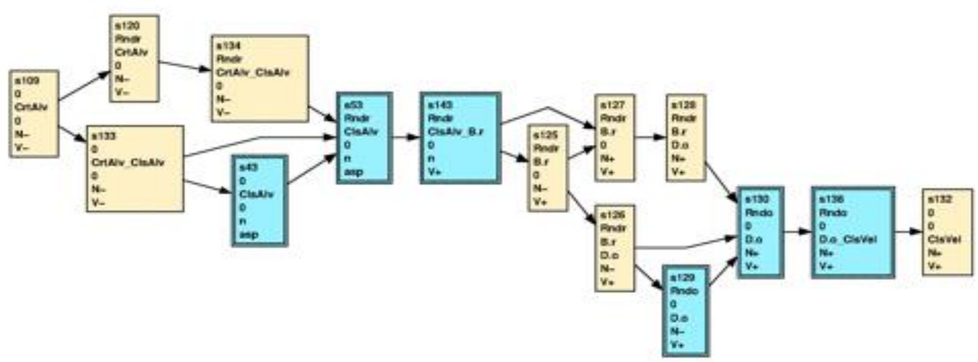


Figure 5: Example of state-transition graph representation for the English word *"strong",* derived from the feature overlapping pattern of Fig. 4.

4. Output representations used in the deep convex network

In this last case study, we show an example of the network output representation in the deep convex network (DCN), with the benefit of drastic simplification of learning the parameters of the full DCN via the use linear-transformation units in the network's output layer [9][10].

Here we show how the use of linear output units in DCN facilitates the learning of the DCN weights with a single module of DCN. First, it is clear that the upper layer weight matrix ***U*** can be efficiently learned once the activity matrix ***H*** over all training samples in the hidden layer is known. Denote the training vectors by $\mathbf{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_i, \cdots, \boldsymbol{x}_N]$. The output of a DCN block is $\boldsymbol{y}_i = \boldsymbol{U}^T \boldsymbol{h}_i$, where $\boldsymbol{h}_i = \sigma(\boldsymbol{W}^T \boldsymbol{x}_i)$ is the hidden-layer vector for sample *i*, $\boldsymbol{U}$ is the weight matrix at the upper layer of a block. $\boldsymbol{W}$ is the weight matrix at the lower layer of a block.

Given target vectors in the full training set with a total of N samples, $\boldsymbol{T} = [\boldsymbol{t}_1, \cdots, \boldsymbol{t}_i, \cdots, \boldsymbol{t}_N]$, where each vector is $\boldsymbol{t}_i = \left[t_{1i}, \cdots, t_{ji}, \cdots, t_{Ci}\right]^T$, the parameters $\boldsymbol{U}$ and $\boldsymbol{W}$ are learned so as to minimize the average of the total square error: $\mathrm{E} = \frac{1}{2}\sum_n ||\boldsymbol{y}_n - \boldsymbol{t}_n||^2 = \frac{1}{2}\mathrm{Tr}[(\mathbf{Y} - \mathbf{T})(\mathbf{Y} - \mathbf{T})^{\mathrm{T}}]$, where the output of the network is: $\boldsymbol{y}_n = \boldsymbol{U}^T \boldsymbol{h}_n = \boldsymbol{U}^T \sigma(\boldsymbol{W}^T \boldsymbol{x}_n) = G_n(\boldsymbol{U}, \boldsymbol{W})$, which depends on both weight matrices, as in the standard neural net. Assuming $\boldsymbol{H} = [\boldsymbol{h}_1, \cdots, \boldsymbol{h}_i, \cdots, \boldsymbol{h}_N]$ is known, or equivalently, $\boldsymbol{W}$ is known. Then, setting the error derivative with respective to $\boldsymbol{U}$ to zero gives: $\boldsymbol{U} = (\boldsymbol{H}\boldsymbol{H}^{\boldsymbol{T}})^{-1}\boldsymbol{H}\boldsymbol{T}^T = \mathrm{F}(\boldsymbol{W})$, where $\boldsymbol{h}_n = \sigma(\boldsymbol{W}^T \boldsymbol{x}_n)$. This provides an explicit constraint between $\boldsymbol{U},$ and $\boldsymbol{W}$, which would be treated independently in the popular backprop algorithm.

Now, given the equality constraint $\boldsymbol{U} = \mathrm{F}(\boldsymbol{W})$, let's use Lagrangian multiplier method to solve the optimization problem in learning $\boldsymbol{W}$. Optimizing the Lagrangian:

$$E = \frac{1}{2}\sum_n ||G_n(\boldsymbol{U}, \boldsymbol{W}) - \boldsymbol{t}_n||^2 + \lambda \, ||\boldsymbol{U} - \mathrm{F}(\boldsymbol{W})||$$

We can then derive batch-mode gradient descent learning algorithm where the gradient takes the following form: $\frac{\partial E}{\partial \boldsymbol{W}} = \boldsymbol{2X}\left[\boldsymbol{H}^T \circ (\boldsymbol{1} - \boldsymbol{H})^T \circ \left[\boldsymbol{H}^\dagger(\boldsymbol{H}\boldsymbol{T}^T)(\boldsymbol{T}\boldsymbol{H}^\dagger) - \boldsymbol{T}^T(\boldsymbol{T}\boldsymbol{H}^\dagger)\right]\right]$,

where $\boldsymbol{H}^\dagger = \boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{H}^T)^{-\boldsymbol{1}}$ is pseudo-inverse of $\boldsymbol{H}$.

Compared with backprop, the above method has less noise in gradient computation due to the exploitation of the explicit constraint $\boldsymbol{U} = \mathrm{F}(\boldsymbol{W})$. As such, it was found experimentally that, unlike backprop, batch training is effective, which aids parallel learning of DCN [10].

5. Summary and conclusions

In this paper, three case studies are presented, all highlighting the importance of designing and learning output representations in machine learning. That is, the machine learning, especially deep learning researcher should turn at least part of their emphasis on input representation learning to the output representation counterpart. Among the three examples, the structured output representation for speech recognition using overlapping articulatory features was elaborated the most (Section 3). Given a drastically different way of capturing contexts in representing sequences of speech classes from the traditional approach, the discussed approach offers a new research direction for improving current speech recognition technology that has been based so far heavily on using DNNs to extract input speech features while paying virtually no attention to designing or learning output representations.

There have been recent advances in output representation learning from the machine learning community [1][30][17] based on latent variable modeling and large scale multi-label learning. And there are more challenging practical applications than speech recognition (e.g. Web search with under-specified output supervision information [21]) which present greater needs for output representation learning in terms of robustness. It is hoped that the three case studies analyzed in this paper can help bring both algorithm-oriented and application-focused machine learning researchers together to advance further the practically useful methods in output representation learning.

Acknowledgments

The author wishes to thank colleagues Jinyu Li, Jian Xue, and J.T. Huang for many discussions on the various topics covered in the case studies included in this paper.

References

[1] Bi, W. and Kwok, J. "Efficient multi-label classification with many labels," Proc. ICML, 2013.
[2] Chomsky, N. and Halle, M. *The Sound Pattern of English*. New York: Harper & Row, 1968.

[3] Clements, G. “The geometry of phonological features,” *Phonology Yearbook*, vol. 2, pp. 225–252, 1985.
[4] Dahl, G. E., Yu, D., Deng, L., and Acero, A. “Large vocabulary continuous speech recognition with context-dependent DBN-HMMs,” Proc. ICASSP, 2011.
[5] Dahl, G. E., Yu, D., Deng, L., and Acero, A. “Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition,” IEEE Trans. on Audio, Speech, and Language Processing, vol. 20, pp. 30–42, 2012.
[6] Dean, J., Corrado, G., R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, Ng, A. “Large Scale Distributed Deep Networks,” Proc. NIPS, 2012.
[7] Deng, L., Hinton, G., and Kingsbury, B. “New types of deep neural network learning for speech recognition and related applications: An overview,” Proc. ICASSP, 2013.
[8] Deng, L., Li, J., Huang, J., Yao, K., Yu, D., Seide, F., Seltzer, M., Zweig, G., He, X., Williams, J., gong, Y., and Acero, A. “Recent advances in deep learning for speech research at Microsoft,” Proc. ICASSP, *2013a.*
[9] Deng, L., Yu, D., and Platt, J. “Scalable stacking and learning for building deep architectures,” Proc. ICASSP, 2012.
[10] Deng, L. and Yu, D. “Deep Convex Network: A scalable architecture for deep learning,” Proc. Interspeech, 2011.
[11] Deng, L., Seltzer, M., Yu, D., Acero, A., Mohamed, A., and Hinton, G. “Binary coding of speech spectrograms using a deep auto-encoder,” Proc. Interspeech, 2010.
[12] Deng, L. and Huang, X.D. “Challenges in Adopting Speech Recognition, Communications of the ACM, vol. 47, no. 1, pp. 11-13, January 2004.
[13] Deng, L. and O'Shaughnessy, D. SPEECH PROCESSING --- A Dynamic and Optimization-Oriented Approach, Marcel Dekker, 2003.
[14] Deng L. and Sun, D. “A statistical approach to automatic speech recognition using the atomic speech units constructed from overlapping articulatory features,” J. Acoust. Society of America, vol. 85, no. 5, pp. 2702-2719, 1994.
[15] Deng, L. Lennig, M., Seitz, F., and Mermelstein, P. “Large vocabulary word recognition using context-dependent allophonic hidden Markov models,” *Computer Speech and Lan*guage, vol. 4, no. 4, pp. 345-357, 1990.
[16] Goldsmith, J. “Phonological Theory,” In John A. Goldsmith. *The Handbook of Phonological Theory*. Blackwell, 1995.
[17] Guo Y. and Schuurmans, D. “Multi-label classification with output kernels,” Proc. ECML, 2013.
[18] Hale, M. and Reiss, C. *The Phonological Enterprise*. Oxford University Press, 2008.
[19] Heigold, G., Vanhoucke, V., Senior, A., Nguyen, P., Ranzato, M., Devin, M., and Dean, J. “Multilingual acoustic models using distributed deep neural networks,” Proc. ICASSP, *2013.*
[20] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. “Deep neural networks for acoustic modeling in speech recognition,” IEEE Signal Processing Magazine, vol. 29, November 2012, pp. 82–97.
[21] Huang, P., He, X., Gao, J., Deng, L., Acero, A., and Heck, L. “Learning deep structured semantic models for web search using clickthrough data,” Proc. CIKM, 2013.
[22] Krizhevsky, A., Sutskever, I. and Hinton, G. “ImageNet classification with deep convolutional neural Networks,” Proc. NIPS 2012.
[23] Le, Q., Ranzato, M., Monga, R., Devin, M., Corrado, G., Chen, K., Dean, J., Ng, A. “Building High-Level Features Using Large Scale Unsupervised Learning,” Proc. ICML 2012.
[24] Markoff. J. “Scientists See Promise in Deep-Learning Programs,” New York Times, Nov 24, 2012.
[25] Rabiner, L. “A tutorial on hidden Markov models and selected applications in speech recognition,” Proceedings of the IEEE, vol.77, no.2, 1989, pp. 257,286.
[26] Sainath, T., Kingsbury, B. V. Sindhwani, E. Arisoy and B. Ramabhadran. “Low-rank matrix factorization for deep neural network training with high-dimensional output targets,” ICASSP, 2013.
[27] Sun J. and Deng, L. “An overlapping-feature based phonological model incorporating linguistic constraints: Applications to speech recognition,” J. Acoust. Society of America, vol. 111, no. 2, pp. 1086-1101, 2002.
[28] Xue, J., Li, J., and Gong, Y. “Restructuring of deep neural network acoustic models with singular value decomposition,” Proc. Interspeech, 2013.
[29] Yan, Z., Huo, Q, and Xu, J. “A scalable approach to using DNN-derived features in GMM-HMM based acoustic modeling for LVCSR,” Proc. Interspeech, 2013.
[30] Yu, C. and Joachims, T. “Learning structural SVMs with latent variables. Proc. ICML, 2009.
[31] Zhou, P., Liu, C., Liu, Q., Dai, L., and Jiang, H. “A cluster-based multiple deep neural networks method for large vocabulary continuous speech recognition,” Proc. ICASSP, 2013.