Automation of System Building for State-of-the-art Large Vocabulary Speech Recognition Using Evolution Strategy

Moriya, T.; Shinozaki, T.; Watanabe, S.; Duh, K.

TR2015-137 December 2015

Abstract

When building a state-of-the-art speech recognition system, a major challenge is the laborious effort required by human experts in tuning numerous parameters. The goal of this paper is to automate the process. We propose to use covariance matrix adaptation evolution strategy (CMA-ES), a meta-heuristic method known to work well on various blackbox optimization problems. Further, we extend CMA-ES to perform multiobjective optimization, giving a high-accuracy speech recognition system with reasonable model size. We apply the proposed automation method to building GMM and DNN HMMbased systems with the Corpus of Spontaneous Japanese (CSJ), a widely used large-scale Japanese speech corpus. Experiments are performed using the TSUBAME 2.5 supercomputer, demonstrating the evolution of a large vocabulary speech recognition system. The optimized training code will be released in the Kaldi speech recognition toolkit as the first publicly available recipe for Japanese large vocabulary speech recognition.

2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)

Automation of System Building for State-of-the-art Large Vocabulary Speech Recognition Using Evolution Strategy

Takafumi Moriya[1], *Takahiro Shinozaki*[1], *Shinji Watanabe*[2], *Kevin Duh*[3]

[1]Tokyo Institute of Technology, Japan
[2]Mitsubishi Electric Research Laboratories, USA
[3]Nara Institute of Science and Technology, Japan

Abstract

When building a state-of-the-art speech recognition system, a major challenge is the laborious effort required by human experts in tuning numerous parameters. The goal of this paper is to automate the process. We propose to use covariance matrix adaptation evolution strategy (CMA-ES), a meta-heuristic method known to work well on various blackbox optimization problems. Further, we extend CMA-ES to perform multi-objective optimization, giving a high-accuracy speech recognition system with reasonable model size. We apply the proposed automation method to building GMM and DNN HMM-based systems with the Corpus of Spontaneous Japanese (CSJ), a widely used large-scale Japanese speech corpus. Experiments are performed using the TSUBAME 2.5 supercomputer, demonstrating the evolution of a large vocabulary speech recognition system. The optimized training code will be released in the Kaldi speech recognition toolkit as the first publicly available recipe for Japanese large vocabulary speech recognition.

Index Terms: speech recognition, evolution strategy, Japanese spontaneous speech, Kaldi toolkit

1. Introduction

Automatic speech recognition (ASR) systems consist of several statistical models that efficiently represent acoustic and linguistic patterns in speech [1]. Parameters of these models, such as the connection weights of a deep neural network (DNN) [2], transition probabilities of a hidden Markov model (HMM) [3], and arc weights of an weighted finite state transducer (WFST) [4], are estimated from large amounts of training data. Additionally, there are various meta-parameters such as model structure (e.g., number of context-dependent HMM states [5], topology of deep networks), training configuration (e.g., learning rate, maximum number of iterations), and system organization (e.g., the choice of sub-systems). Meta-parameter tuning is essential for building state-of-the-art systems, but as a consequence of the increased complexity of recent ASR techniques, this process is becoming increasingly more difficult and time-consuming even for human experts. There is thus a strong demand to automate the tuning process by computers.

If we consider system performance as a function of meta-parameters, then tuning can be formalized as an optimization problem. This function is very complex; analytic solution is infeasible. Further, evaluating the value of the function for an input requires a lot of computation since it involves model training and system evaluation using training and development sets. An approach to this type of problem is to use evolutionary algorithms such as genetic algorithm (GA) [6] and evolution strategy (ES) [7]. The evolutionary algorithm is a fairly broad concept, but the basic idea is to represent possible solutions as genes and search the optimal solution by iteratively evaluating and sampling generations of genes.

Several researchers have applied GA to HMM acoustic modeling with promising results [8, 9, 10]. Previously, we have used covariance matrix adaptation evolution strategy (CMA-ES) [7, 11, 12, 13], a type of ES that represent gene distribution using a multivariate Gaussian, to optimize the meta-parameters of a medium vocabulary size DNN ASR system [14]. We have also applied CMA-ES and GA to optimize DNNs used as feature extractors for a keyword spotting system [15]. The DNN had an extended structure represented by a directed acyclic graph, which were subject for optimization. We found that CMA-ES and GA gave similar final performance given sufficient generations, but CMA-ES was superior when the number of generations is small. We also found that while CMA-ES is successful in optimizing single objectives such as word accuracy or mean average precision [16], there were cases where the model size was extremely large, which is not practical in terms of the memory efficiency and decoding speed.

In this paper, our contributions are:

1. an extension of CMA-ES to multi-objective optimization of both word accuracy and model size,
2. a proof-of-concept demonstration of automation of system building for complex large vocabulary DNN HMM, with state-of-the-art results in Japanese spontaneous speech.

Although typical multi-objective optimization can be often performed by using the weighted combination of individual objective functions, our goal is to automate the building process without tuning, and introducing additional weight parameter must be avoided. Instead, this paper uses the Pareto optimality [17, 18], which can reasonably rank multi-objective scores in an automatic way. The systems are built using a massively parallel computing platform TSUBAME 2.5 supercomputer[1] developed at Tokyo Institute of Technology, which ranked 6th place in the Green 500 supercomputer ranking. The process is highly automated based on the multi-objective CMA-ES and implemented based on the Kaldi toolkit [19]. The optimized training code will be released in the Kaldi toolkit as the first publicly available recipe for Japanese large vocabulary speech recognition.

2. Formulation

This section first describes a single-objective optimization problem, then extends it to a multi-objective one. Let us represent an evaluation function $y = f(\boldsymbol{x})$ as the accuracy (or some

[1]`http://www.gsic.titech.ac.jp/en`

other correctness measure) of an ASR system built from meta-parameters $\boldsymbol{x}$. The process of finding the optimal tuning parameter $\boldsymbol{x}^*$, which maximizes the ASR accuracy, can be formulated as the following optimization problem:

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}} f(\boldsymbol{x}). \tag{1}$$

As ASR systems are extremely complex, there is no analytical form for the solution. It is difficult to include specific knowledge on f in the optimization, so such situations are best handled by considering f as a black box. Moreover, evaluating the function value $f(\boldsymbol{x})$ is very costly, because training a large vocabulary model and computing its development set accuracy can take considerable time. The key point here is thus for the black box optimization to generate appropriate hypotheses $\hat{\boldsymbol{x}}$ to find the best $\boldsymbol{x}^*$ in as few ASR training and evaluation steps ($f(\boldsymbol{x})$) as possible.

2.1. CMA Evolution Strategy

CMA-ES iteratively estimates the parameters of a sample distribution for $\boldsymbol{x}$ such that the distribution is concentrated in a region with high values of $f(\boldsymbol{x})$. Hypotheses are sampled from a multivariate Gaussian distribution:

$$\hat{\boldsymbol{x}} \sim \mathcal{N}(\boldsymbol{x}|\hat{\boldsymbol{\theta}}) \ \text{ s.t. } \hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \underbrace{\int f(\boldsymbol{x})\mathcal{N}(\boldsymbol{x}|\boldsymbol{\theta})d\boldsymbol{x}}_{\triangleq \mathbb{E}[f(\boldsymbol{x})|\boldsymbol{\theta}]}. \tag{2}$$

CMA-ES iteratively re-estimates the mean vector and covariance matrix ($\boldsymbol{\theta}$) so as to optimize the expected value of $f(\boldsymbol{x})$ under the distribution. Since the concrete functional form of f is unknown, it is difficult to deal with Eq. (2) analytically. To solve this problem, we use a natural gradient method [20] by taking a gradient of $\mathbb{E}[f(\boldsymbol{x})|\boldsymbol{\theta}]$ with respect to $\boldsymbol{\theta}$. The expectation in the natural gradient can be approximately computed by using Monte Carlo sampling with the function evaluation $y_k = f(\boldsymbol{x}_k)$:

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}[f(\boldsymbol{x})|\boldsymbol{\theta}] \approx \frac{1}{K}\sum_{k=1}^{K} y_k \nabla_{\boldsymbol{\theta}} \log \mathcal{N}(\boldsymbol{x}_k|\boldsymbol{\theta}), \tag{3}$$

where $\boldsymbol{x}_k$ is sampled from the previously estimated distribution $\mathcal{N}(\boldsymbol{x}|\hat{\boldsymbol{\theta}}_{n-1})$. Since CMA-ES uses a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\theta})$ with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, we can obtain the analytical forms of $\hat{\boldsymbol{\mu}}_n$ and $\hat{\boldsymbol{\Sigma}}_n$ by substituting the concrete Gaussian form into Eq. (3), leading to:

$$\begin{cases} \hat{\boldsymbol{\mu}}_{n-1} + \epsilon_{\boldsymbol{\mu}} \sum_{k=1}^{K} w(y_k)(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_{n-1}) \\ \hat{\boldsymbol{\Sigma}}_{n-1} + \epsilon_{\boldsymbol{\Sigma}} \sum_{k=1}^{K} w(y_k)\big((\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_{n-1})(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_{n-1})^{\intercal} - \hat{\boldsymbol{\Sigma}}_{n-1}\big) \end{cases} \tag{4}$$

where $^{\intercal}$ is the matrix transpose. Note that, as in [7], y_k in Eq. (3) is approximated in Eq. (4) as a weight function $w(y_k)$, defined as:

$$w(y_k) = \frac{\max\{0, \log(K/2+1) - \log(\mathrm{R}(y_k))\}}{\sum_{k'=1}^{K} \max\{0, \log(K/2+1) - \log(\mathrm{R}(y_{k'}))\}} - \frac{1}{K}, \tag{5}$$

where $\mathrm{R}(y_k)$ is a ranking function that returns the descending order of y_k among $y_{1:K}$ (i.e., $\mathrm{R}(y_k) = 1$ for the highest y_k, $\mathrm{R}(y_k) = K$ for the smallest y_k, etc.). This equation only considers the order of y, which makes the updates less sensitive to evaluation measurements (e.g., to prevent from the different results using word accuracies and the negative sign of error counts).

Algorithm 1 Multi-objective CMA-ES

```
1:  Initialization of $\hat{\boldsymbol{\mu}}_0$ and $\hat{\boldsymbol{\Sigma}}_0$
2:  for $n = 1$ to $N$ do
3:      for $k = 1$ to $K$ do
4:          Sample $\boldsymbol{x}_k$ from $\mathcal{N}(\boldsymbol{x}|\hat{\boldsymbol{\mu}}_{n-1}, \hat{\boldsymbol{\Sigma}}_{n-1})$
5:          Evaluate $F(\boldsymbol{x}_k) \triangleq [f_1(\boldsymbol{x}_k), f_2(\boldsymbol{x}_k), \dots, f_J(\boldsymbol{x}_k)]$
6:      end for
7:      Rank $\{F(\boldsymbol{x}_k)\}_{k=1}^{K}$ according to the Pareto optimality
8:      Update $\hat{\boldsymbol{\mu}}_n$ and $\hat{\boldsymbol{\Sigma}}_n$
9:  end for
10: return subset of solutions $\{\boldsymbol{x}, F(\boldsymbol{x})\}$ that lie on the Pareto
    front (rank 1) of all stored $N \times K$ samples
```

2.2. Multi-objective Optimization

Besides high accuracy, objectives such as small model size and fast run-time are also important in practice. Without loss of generalization, assume that we wish to maximize J objectives $F(\boldsymbol{x}) \triangleq [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \dots, f_J(\boldsymbol{x})]$ with respect to $\boldsymbol{x}$ jointly. Since objectives may conflict, we adopt a notion of optimality known as Pareto Optimality [21]: First, if $f_j(\boldsymbol{x}_k) \geq f_j(\boldsymbol{x}_{k'}) \ \forall \ j = 1,..,J$ and $f_j(\boldsymbol{x}_k) > f_j(\boldsymbol{x}_{k'})$ for at least one objective j, then we say that $\boldsymbol{x}_k$ *dominates* $\boldsymbol{x}_{k'}$ and write $F(\boldsymbol{x}_k) \rhd F(\boldsymbol{x}_{k'})$. Given a set of candidate solutions, $\mathbf{x_k}$ is *pareto-optimal* iff there does not exist another $\mathbf{x_{k'}}$ such that $F(\boldsymbol{x}_k) \rhd F(\boldsymbol{x}_{k'})$.

Pareto-optimality formalizes the intuition that a solution is good if no other solution outperforms (dominates) it in all objectives. Given a set of candidates, there are generally multiple Pareto-optimal solutions; this is known as the Pareto frontier. Note that an alternative approach is to combine multiple objectives into a single objective via weighted linear combination: $\sum_j \beta_j f_j(\boldsymbol{x})$, where $\sum_j \beta_j = 1$ and $\beta_j > 0$. The advantage of the Pareto definition is that weights β_j need not be specified and is more general, i.e. the optimal solution obtained by any setting of β_j is guaranteed to be included in the Pareto frontier.

CMA-ES can be extended to optimize multiple objectives by modifying the rank function $\mathrm{R}(y_k)$ used in Eq. (5). Given a set of solutions $\{\boldsymbol{x}_k\}$, we first assign rank = 1 to those on Pareto frontier. Then we exclude these rank 1 solutions and compute the Pareto frontier again on the remaining solutions, assigning them rank 2. This is iterated until no $\{\boldsymbol{x}_k\}$ remain, and in the end we obtain a ranking of all solutions according to multiple objectives. The rest of CMA-ES remains unchanged; by this modification future generations are drawn to optimize not a single, but multiple objectives. With some book-keeping, this ranking can be computed efficiently in $O(J \cdot K^2)$ [17].

In our experiments, we jointly optimize for high accuracy and small model size. To prevent unreasonably low accuracy solutions (that might have good model size) to appear on the Pareto frontier, we need an additional heuristic. Solutions with accuracy below a manually-set threshold are penalized to not appear as rank 1.

Algorithm 1 summarizes the CMA-ES optimization procedure with the Pareto optimality, which is used to rank the multi objectives $F(\boldsymbol{x}_k)$. The obtained rank is used to update the mean vector and covariance matrix of the CMA-ES. The CMA-ES gradually samples neighboring tuning parameters from the initial values, and finally provide a subset of solutions $\{\boldsymbol{x}, F(\boldsymbol{x})\}$ that lie on the Pareto front (rank 1) of all stored $N \times K$ samples. Note that, as CMA-ES is a gradient method, initial values need to be set. As CMA-ES assumes a multivariate Gaussian

Table 1: *Meta-parameters subject for optimization.*

Name	Initial value	Description	Conversion from gene value x
feat_type	MFCC	MFCC, FBANK, or PLP (applied for GMM and DNN)	$\bmod\ (\lfloor ceil(abs(x) * 3) \rfloor, 3)$
splice	5	segment length for DNN	$ceil(10^x)$
nn_depth	6	number of hidden layers	$ceil(10^x)$
hid_dim	2048	units per layer	$ceil(10^x)$
param_stddev_first	0.1	init parameters in 1st RBM	10^x
param_stddev	0.1	init parameters in other RBMs	10^x
rbm_lrate	0.4	RBM learning rate	10^x
rbm_lrate_low	0.01	lower RBM learning rate	10^x
rbm_l2penalty	0.0002	RBM Lasso regularization	10^x
learn_rate	0.008	learning rate for fine tuning	10^x
momentum	0.00001	momentum for fine tuning	10^x

for $\boldsymbol{x}$, it is originally suitable for tuning parameters that take continuous values, and needs some extra discretization for discrete value optimization. Finally, the evaluation of $F(\boldsymbol{x}_k)$ can be performed independently for each k and can thus be easily parallelized. The number of samples K is automatically determined from the number of dimensions of $\boldsymbol{x}$ [7], or we can set it manually by considering computer resources.

3. Experimental setup

Experiments were performed using the Kaldi speech recognition toolkit with speech data from CSJ [22]. We ran two separate experiments with different amounts of training sets: the first set is 240 hours of academic presentations, and the second set is a 100-hour subset. A common development set consisting of 10 academic presentations was used for performance computation in CMA-ES. The official evaluation set defined in CSJ, which had 10 academic presentations amounting to 110 minutes total, is used as the evaluation set.

Acoustic models were trained by first making a GMM-HMM by maximum likelihood estimation, and then building a DNN-HMM by pre-tuning and fine-tuning using alignments generated by the GMM-HMM. For the performance evaluation of the system, the DNN-HMM was used as the final model. Language model was a 3-gram model trained on CSJ with the academic and other types of presentations. Speech recognition was performed using the openfst WFST decoder [23]. These model trainings and system evaluations were performed by borrowing a Kaldi recipe that was originally designed for the Switchboard corpus (i.e. egs/swbd/s5b).

In the evolution experiments, feature types, DNN structures, and learning parameters were optimized. These meta-parameters were implemented as configuration variables for training scripts. Table 1 lists these variables. We specify three base feature types (feat_type) for GMM-HMM and DNN-HMM models: mel-frequency cepstrum coefficients (MFCC) [24], perceptual linear prediction (PLP) [25], and filter bank (FBANK). The GMM-HMMs were trained with the specified features and their delta and delta-delta, and the DNN-HMMs were trained with the features expanded to # splice pre and post context frames. Other settings were the same as those used in the Kaldi recipe. Since CMA-ES uses genes represented as real-valued vectors, some variables need a mapping of a real scalar value to a required type such as integers. The last column in the table shows the mappings.

The system training and evaluation were performed using the TSUBAME 2.5 supercomputer. Population sizes for CMA-ES were 20 for the 100 hour training set and 44 for the 240 hour training set. Same numbers of NVIDIA K20X GPGPUs (20 and 44, one per gene) were used in parallel through the message passing interface (MPI). For the MFCC based baseline system using the 240 hour training data, it took 12 hours for the RBM pretraining and 70 hours for the fine tuning. The minimum accuracy thresholds for multi-objective optimization were set to include top 1/2 and 1/3 populations at each generation respectively for the trainings using the 100 and 240-hour data sets.

Table 2: *Word error rate and DNN size of base systems.*

Training data	Dev set	Eval set	DNN size (M byte)
MFCC 100h	14.4	13.1	161.8
PLP 100h	14.5	13.1	163.2
FBANK 100h	15.1	13.8	163.9
MFCC 240h	13.5	12.5	161.8
PLP 240h	13.6	12.5	163.2
FBANK 240h	14.1	13.0	163.9

Figure 1: *Word accuracy of development set and DNN size when 100 hour training data was used.*

4. Results

Table 2 shows word error rates and DNN sizes of systems with the default configuration using the 100 and 240 hour training sets with one of the three types of features. Among the features, MFCC was the default in the Switchboard recipe, and it gave the lowest word error rates for the development set for both of the training sets. The corresponding word error rates for the evaluation set were 13.1% and 12.5% for the 100 and 240 hour training sets, respectively.

Figure 1 shows a scatter plot of a word accuracy (i.e. 100 minus word error rate) evaluated for the development set and a

Table 3: *Details of systems that were selected according to word error rates on development set.*

Train	item	base	gen 1	gen 2	gen 3	gen 4
100h	WER(Dev)	14.4	14.3	14.1	14.1	14.0
	WER(Eval)	13.1	13.2	12.9	13.0	12.9
	Feature type	MFCC	MFCC	MFCC	MFCC	MFCC
	# layers	6	8	7	6	6
	# units per layer	2048	2203	2124	2622	2198
240h	WER(Dev)	13.5	13.2	13.0	13.0	running
	WER(Eval)	12.5	12.3	12.1	12.1	
	Feature type	MFCC	MFCC	MFCC	MFCC	
	# layers	6	7	6	6	
	# units per layer	2048	1755	1907	2575	
	Training completion rate	-	65.9	59.1	79.5	

Figure 2: *Word accuracy of development set and DNN size when 240 hour training data was used.*

file size of the DNNs. Ideally, we want systems on the lower-right side of the plot. Baseline is the MFCC based system. The initial mean vector of the multivariate Gaussian for CMA-ES was set equal to the baseline settings. Therefore, the genes of the first generation randomly distributed around the baseline settings. Accordingly, their word accuracy and DNN file sizes distributed around the baseline. The Gaussian distribution was then updated based on the results of the first generation. The genes of the second generation were sampled from the updated Gaussian. In the scatter plot, it is seen that the distribution of the second generation shifted toward higher word accuracies and lower DNN file sizes. After the second generation, the accuracies continued improving, though some larger variance of DNN sizes was observed.

Similarly, Figure 2 shows a scatter plot of the recognition systems using the 240 hour training set. Due to the increased data size and a time constraint, a time limit was introduced for the training at each generation. If a system did not finish the training within four days, it was interrupted and the last model in the iterative back-propagation training at that timing was used as the final model. In the figure, it is seen that the distributions shifted toward higher word accuracies and lower DNN file sizes with each successive generation, as desired.

Table 3 describes the details of the best systems selected based on word error rates on the development set. When the 100 hour training set was used, the word error rate of the development set decreased by 0.4% at the fourth generation from the baseline with about 7% increase in the DNN model size. The word error rate reduction for the evaluation set was 0.2%. When the 240 hour training set was used, the model sizes generally became smaller than when the 100-hour set was used. This was mainly due to the introduction of the time limit to the training process where larger models tend to fail to complete the training in time. In spite of the constraint, better improvements were obtained than when the 100 hour training set was used. At the 3rd generation, the word error rate was reduced by 0.5% for the development set and 0.4% for the evaluation set, which corresponded to the relative word error rate reduction of 3.7% and 3.6%, respectively. The differences were statistically significant with the MAPSSWE test [26]. Although the size of the DNN was 45.0% larger than the baseline, if we choose the one that was smaller than the baseline and had the third smallest word error rate in the development set, it gave 3.0% relative word error rate reduction in the evaluation set and 8.3% model size reduction at the same time. The difference of the word error rate of the system from the baseline was also statistically significant.

5. Conclusions

To automate the process of building the state-of-the-art large vocabulary speech recognition systems, we propose to use covariance matrix adaptation evolution strategy (CMA-ES). Further, we extend CMA-ES using Pareto rank to perform multi-objective optimization considering both word accuracy and model size while minimizing tuning factors required for human experts. The proposed automation method was applied to build GMM and DNN HMM-based systems using data from the Corpus of Spontaneous Japanese (CSJ). Experiments were performed using TSUBAME 2.5 supercomputer. It has been demonstrated that distributions of word accuracies move toward better performance generation by generation. So far, four and three generations have been evaluated for the 100 and 240 hour training sets, respectively. For the 240 hour training set, 3.7% relative word error rate reduction from the baseline was obtained by a system selected based on the word error rate on the development set. Future work includes continuing the experiments for more generations, applying the proposed method for more complex systems, and improving the efficiency of the evolution strategy.

6. Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 26280055.

7. References

[1] S. Furui, "Recent progress in corpus-based spontaneous speech recognition," *IEICE Transactions on Information and Systems*,

vol. E88-D, pp. 366–375, 2005.

[2] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[3] M. Gales and S. Young, "The application of hidden Markov models in speech recognition," *Signal Processing*, vol. 1, no. 3, pp. 195–304, 2007.

[4] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech and Language*, vol. 16, pp. 69–88, 2002.

[5] S. Young, J. Odell, and P. Woodland, "Tree-based state tying for high accuracy acoustic modelling," in *Proceedings of the Workshop on Human Language Technology*, 1994, pp. 307–312.

[6] L. Davis *et al.*, *Handbook of genetic algorithms*. Van Nostrand Reinhold New York, 1991, vol. 115.

[7] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, 2003.

[8] C. W. Chau, S. Kwong, C. K. Diu, and W. R. Fahrner, "Optimization of HMM by a genetic algorithm," in *Proc. ICASSP*, vol. 3. IEEE, 1997, pp. 1727–1730.

[9] S. Kwong, C. W. Chau, K. F. Man, and K. S. Tang, "Optimisation of HMM topology and its model parameters by genetic algorithms," *Pattern Recognition*, vol. 34, no. 2, pp. 509–522, 2001.

[10] F. Yang, C. Zhang, and T. Sun, "Comparison of particle swarm optimization and genetic algorithm for HMM training," in *Proc. ICPR*, 2008, pp. 1–4.

[11] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík, "Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009," in *Proc. the 12th annual conference companion on Genetic and evolutionary computation (GECCO)*, 2010, pp. 1689–1696.

[12] Y. Akimoto, Y. Nagata, I. Ono, and S. Kobayashi, "Bidirectional relation between CMA evolution strategies and natural evolution strategies," in *Proc. Parallel Problem Solving from Nature (PPSN)*, 2010, pp. 154–163.

[13] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, and J. Schmidhuber., "Natural evolution strategies," *arXiv preprint arXiv:1106.4487*, 2011.

[14] S. Watanabe and J. Le Roux, "Black box optimization for automatic speech recognition," in *Proc. ICASSP*. IEEE, 2014, pp. 3256–3260.

[15] T. Shinozaki and S. Watanabe, "Structure discovery of deep neural network based on evolutionary algorithms," in *Proc. ICASSP*, 2015, (accepted).

[16] K. Kishida, "Property of average precision and its generalization: an examination of evaluation indicator for information retrieval," National Institute of Informatics, Tech. Rep., 2005.

[17] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, 2002.

[18] B. Sankaran, A. Sarkar, and K. Duh, "Multi-metric optimization using ensemble tuning." in *HLT-NAACL*, 2013, pp. 947–957.

[19] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlıcek, Y. Qian, P. Schwarz, J. Silovskı, G. Stemmer, and K. Veselı, "The Kaldi speech recognition toolkit," in *Proc. ASRU*, 2011.

[20] S. Amari, "Natural gradient works efficiently in learning," *Neural computation*, vol. 10, no. 2, pp. 251–276, 1998.

[21] K. Miettinen, *Nonlinear Multiobjective Optimization*. Springer, 1998.

[22] S. Furui, K. Maekawa, and M. H. Isahara, "A Japanese national project on spontaneous speech corpus and processing technology," in *Proc. ASR'00*, 2000, pp. 244–248.

[23] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "Openfst: A general and efficient weighted finite-state transducer library," *Implementation and Application of Automata*, pp. 11–23, 2007.

[24] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transaction on Acoustic Speech and Singal Processing*, vol. 28, no. 4, pp. 357–366, 1980.

[25] H. Hermansky, "Perceptual linear predictive (plp) analysis of speech," *J. Acoust. Soc. Am*, vol. 87, no. 4, pp. 1738–1752, 1990.

[26] L. Gillick and S. Cox, "Some statistical issues in the comparison of speech recognition algorithms," in *Proc. ICASSP*, 89, pp. 532–535.