BLACK BOX OPTIMIZATION FOR AUTOMATIC SPEECH RECOGNITION

Shinji Watanabe and Jonathan Le Roux

Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA

ABSTRACT

State-of-the-art automatic speech recognition (ASR) systems are very complex, combining multiple techniques and involving many types of tuning parameters (e.g., numbers of states and Gaussians in HMMs, numbers of neurons/layers and learning rates in neural networks, etc.). To reach optimal performance in such systems, deep understanding and expertise of each component is necessary, thus limiting the development of ASR systems to skilled experts. To overcome the problem, this paper studies the use of black box optimization, which automatically tunes systems without any prior knowledge. We consider an ASR system as a function with tuning parameters as input and speech recognition performance (e.g., word accuracy) as output, and we investigate two probabilistic black box optimization techniques: Covariance Mean Adaptation Evolution Strategy (CMA-ES) and Bayesian optimization using Gaussian process. Middle-vocabulary speech recognition experiments show the effectiveness of black box optimization, as performance approaching that of fine-tuned systems obtained by experts and/or outperforming that of sub-optimal systems can be automatically obtained.

Index Terms— Speech recognition, Black box optimization, CMA-ES, Bayesian optimization, Gaussian process

1. INTRODUCTION

The recent research trend in large vocabulary continuous speech recognition is to combine multiple techniques in a large system, resulting in a highly complicated system [1–3]. Each of these techniques has some tuning parameters, which have to be optimized depending on the task, and thus need to be tuned in advance for practical use. This trend has further accelerated with the emergence of deep learning, where many types of configurations with different network topologies and learning rates can be considered [4]. For example, the numbers of states and Gaussians in HMMs or the numbers of neurons/layers and the learning rates in neural networks need to be tuned. As their impact on performance is complex and highly intertwined, it is necessary to have deep understanding and expertise about each component of the system to tune the parameters for optimal performance, thus limiting the development and deployment of automatic speech recognition (ASR) systems to skilled experts.

To overcome this problem, there have been many efforts to optimize these tuning parameters for specific components of ASR systems. For example, as the HMM is a simple generative model, information criterion approaches and Bayesian approaches can be used to automatically determine optimal model topologies [5–7]. Bayesian approaches also enable the optimization of some other hyper-parameters based on the evidence framework [8]. However, such analytical treatments can be only applied to simple probabilistic models, and it is virtually impossible to apply them to current complicated systems that combine various probabilistic and non-probabilistic models.

This paper focuses on an alternative strategy for the optimization of tuning parameters: by considering an ASR system as a function with the tuning parameters as inputs and speech recognition performance (e.g., word accuracy) as output, we can make use of black box optimization, which automatically tunes systems without using any prior knowledge about their inner workings. We consider here two probabilistic black box optimization techniques based on Covariance Mean Adaptation Evolution Strategy (CMA-ES) [9] and Bayesian optimization using Gaussian process [10, 11]. CMA-ES has been developed as an evolutionary algorithm for continuous value optimization, while Bayesian optimization has been mainly developed in the machine learning community followed by the progress of Gaussian process studies [12]. These approaches are starting to be widely used for various pattern recognition problems including neural networks with complex topology [13, 14]. They both attempt to efficiently find a set of tuning parameters that yields optimal performance of the black box system by optimizing objective functions computed using some concept of expectation. CMA-ES iteratively estimates a probability distribution on the input parameters such that the expectation of the output with respect to the tuning parameters is highest; neighboring tuning parameters are sampled from the distribution, thus gradually evolving from initial values to the optimal values. Bayesian optimization greedily searches for the set of tuning parameters to try next that will lead to the highest expected improvement of the output compared to previous steps.

This paper provides analytical and empirical discussions of both approaches, from formulation to middle-vocabulary speech recognition experiments. Section 2 introduces the formulations of CMA-ES and Bayesian optimization, and discusses their characteristics for a practical use. Section 3 shows the effectiveness of black box optimization, especially CMA-ES, by using various ASR experiments.

Relation to prior work

Black box optimization has been previously applied to *each component module* of speech recognition (e.g., HMM training using genetic algorithm and particle swarm optimization [15–17], neural network using Bayesian optimization [18]). However, this paper investigates the use of black box optimization, especially CMA-ES, to the paradigm of *whole speech recognition systems* dealing with various tuning parameters from various components (e.g., HMM topologies, deep neural network topologies and learning rates, discriminative training, and context modeling via LDA).

2. PROBABILISTIC BLACK BOX OPTIMIZATION

Let us represent an ASR system evaluation as a function f of a vector $\mathbf{x}$ of tuning parameters which returns the accuracy $y = f(\mathbf{x})$ (or some other correctness measures). The D dimensional vector $\mathbf{x}$ denotes for example the number of shared tri-phone states, learning rates, etc. The process of finding the optimal tuning parameter $\mathbf{x}^*$, which maximizes the ASR accuracy, can be formulated as the following optimization problem:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmax}}\, f(\mathbf{x}). \tag{1}$$

As ASR systems are extremely complicated, there is no analytical form for f, and it is difficult to include specific knowledge on f. Such situations are best handled by considering f as a black box. Moreover, evaluating $f(\mathbf{x})$ takes a very long time, because we need to train a model and evaluate it on a development set. The key point here is thus for the black box optimization to generate appropriate hypotheses $\hat{\mathbf{x}}$ to find the best $\mathbf{x}^*$ in as few ASR evaluations ($f(\mathbf{x})$) as possible. We consider two probabilistic black box optimization approaches.

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

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

CMA-ES assumes the sample distribution as multivariate Gaussian whose parameters (mean vector and covariance matrix) are estimated so as to optimize the expected value of $f(\mathbf{x})$ under the distribution.

Similarly to CMA-ES, Bayesian optimization also relies on some concept of expectation. But while CMA-ES involves a distribution over the tuning parameter $\mathbf{x}$ and takes the expectation over $\mathbf{x}$, Bayesian optimization uses a probabilistic model of the output y and considers the expectation of some quantity defined over y. Several expected objective functions have been proposed [10], and we use here *expected improvement*, which is suggested as a practical choice [11]. We define the *improvement* from the best score among $m-1$ previous scores as $\max\{0, y - y^*_{m-1}\}$ where $y^*_{m-1} = \max_{1 \le m' \le m-1} y_{m'}$. Bayesian optimization then performs a deterministic search for the next candidate $\hat{\mathbf{x}}_m$ by optimizing the expected improvement over y, $a^{\mathrm{EI}}(\mathbf{x}_m)$:

$$\hat{\mathbf{x}}_m = \underset{\mathbf{x}_m}{\operatorname{argmax}} \underbrace{\int \max\{0, y - y^*_{m-1}\} p(y|D_{1:m-1}, \mathbf{x}_m)dy}_{\triangleq a^{\mathrm{EI}}(\mathbf{x}_m)}, \tag{3}$$

where $p(y|D_{1:m-1}, \mathbf{x}_m)$ is a predictive distribution of y given observed data $D_{1:m-1} \triangleq \{\mathbf{x}_{1:m-1}, y_{1:m-1}\}$ and $\mathbf{x}_m$, modeled as a Gaussian process. Equation (3) selects $\mathbf{x}_m$ that is likely to lead to a high score y_m with high confidence given the predictive distribution.

The next subsections describe each method in more details.

2.1. Covariance Matrix Adaptation Evolution Strategy

We describe CMA-ES by following the derivation based on natural evolution strategies [14,19], which makes the theoretical relationship with Bayesian optimization more apparent. Since the concrete function 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] with Fisher information matrix $\mathbf{F}$ and iteratively update $\hat{\boldsymbol{\theta}}$ as:

$$\hat{\boldsymbol{\theta}}_n = \hat{\boldsymbol{\theta}}_{n-1} + \epsilon \mathbf{F}^{-1} \nabla_{\boldsymbol{\theta}} \mathbb{E}[f(\mathbf{x})|\boldsymbol{\theta}]|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_{n-1}}, \tag{4}$$

where n is an iteration index and ϵ a step size that can be changed depending on the elements of $\boldsymbol{\theta}$ in the implementation. The gradient of the expectation in Eq. (4) can be derived as follows:

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}[f(\mathbf{x})|\boldsymbol{\theta}] = \mathbb{E}[f(\mathbf{x}) \nabla_{\boldsymbol{\theta}} \log \mathcal{N}(\mathbf{x}|\boldsymbol{\theta})|\boldsymbol{\theta}]. \tag{5}$$

This expectation can be approximately computed by using Monte Carlo sampling with the function evaluation $y_k = f(\mathbf{x}_k)$:

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}[f(\mathbf{x})|\boldsymbol{\theta}] \approx \frac{1}{K} \sum_{k=1}^{K} y_k \nabla_{\boldsymbol{\theta}} \log \mathcal{N}(\mathbf{x}_k|\boldsymbol{\theta}), \tag{6}$$

Algorithm 1 CMA-ES

```
Initialization of μ̂_0 and Σ̂_0, and y*_0 = ∅
for n = 1 to N do
    for k = 1 to K do
        Sample x_k from N(x|μ̂_{n-1}, Σ̂_{n-1})
        Evaluate y_k = f(x_k)
    end for
    Update μ̂_n and Σ̂_n
    Store y*_n = max{y_{1:K}, y*_{n-1}} and corresponding x*_n
end for
return (x*_N, y*_N)
```

where $\mathbf{x}_k$ is sampled from the previously estimated distribution $\mathcal{N}(\mathbf{x}|\hat{\boldsymbol{\theta}}_{n-1})$. Since CMA-ES uses a multivariate Gaussian distribution $\mathcal{N}(\mathbf{x}|\boldsymbol{\theta})$ 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. (6) and $\mathbf{F}$, leading to[1]:

$$\begin{cases} \hat{\boldsymbol{\mu}}_{n-1} + \epsilon_{\boldsymbol{\mu}} \sum_{k=1}^{K} w(y_k)(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_{n-1}) \\ \hat{\boldsymbol{\Sigma}}_{n-1} + \epsilon_{\boldsymbol{\Sigma}} \sum_{k=1}^{K} w(y_k)\big((\mathbf{x}_k - \hat{\boldsymbol{\mu}}_{n-1})(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_{n-1})^{\intercal} - \hat{\boldsymbol{\Sigma}}_{n-1}\big) \end{cases} \tag{7}$$

where $^{\intercal}$ is the matrix transpose. Note that, as in [9], y_k in Eq. (6) is approximated in Eq. (7) as a weight function $w(y_k)$, defined as:

$$w(y_k) = \frac{\max\{0, \log(K/2+1) - \log(\mathrm{R}(y_k))\}}{\sum_{k'=1}^{K} \max\{0, \log(K/2+1) - \log(\mathrm{R}(y_{k'}))\}} - \frac{1}{K}, \tag{8}$$

where $\mathrm{R}(y_k)$ is a 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). The number of samples K is automatically determined from the number of dimensions of $\mathbf{x}$ [9].

Algorithm 1 summarizes the CMA-ES optimization procedure, which gradually samples neighboring tuning parameters from the initial values to the optimal values. Note that, as CMA-ES is a gradient method, initial values need to be set. As CMA-ES assumes a multivariate Gaussian for $\mathbf{x}$, it is originally suitable for tuning parameters that take continuous values, and needs some extra discretization for discrete value optimization. Finally, the evaluation of $f(y_k)$ can be performed independently for each k and can thus be easily parallelized.

2.2. Bayesian optimization based on Gaussian process

Bayesian optimization analytically obtains the expected improvement from the predictive distribution $p(y|D_{1:m-1}, \mathbf{x}_m)$ as defined in Eq. (3). To obtain the predictive distribution, we first consider the distribution of the observation vector $p(\mathbf{y}_{1:m-1}|\mathbf{x}_{1:m-1})$. In the Gaussian process framework (see [12, 21] for an overview of Gaussian processes), it is represented by an $(m-1)$-dimensional Gaussian with zero mean vector and a Gram matrix $\mathbf{K}$ as covariance matrix:

$$p(\mathbf{y}_{1:m-1}|\mathbf{x}_{1:m-1}) = \mathcal{N}(\mathbf{y}_{1:m-1}|\mathbf{0}, \mathbf{K}). \tag{9}$$

The Gram matrix $\mathbf{K}$ is defined as $\mathbf{K} = (k(\mathbf{x}_i, \mathbf{x}_j))_{i,j}$ with a kernel function $k(\mathbf{x}, \mathbf{x}')$ whose concrete form is discussed later. Similarly,

[1] Practical implementations of CMA-ES [9] also consider the update of a global scalar variance term and an additional smoothing term in the covariance matrix update to consider the effect of the change of $\hat{\boldsymbol{\mu}}_{n-1}$ for its update.

Algorithm 2 Bayesian optimization

1: Set the domain $\mathcal{X}$ of $\mathbf{x}_0$, and $y_0^* = \emptyset$
2: **for** $m = 1$ to M **do**
3: Compute $\hat{\mathbf{x}}_m = \text{argmax}_{\mathbf{x}_m} a^{\text{EI}}(\mathbf{x}_m)$
4: Evaluate $y_m = f(\hat{\mathbf{x}}_m)$
5: Store $y_m^* = \max\{y_m, y_{m-1}^*\}$ and corresponding $\mathbf{x}_m^*$
6: **end for**
7: **return** $(\mathbf{x}_M^*, y_M^*)$

the posterior distribution of $[\mathbf{y}_{1:m-1}^\intercal, y]^\intercal$ is also represented by the following m-dimensional multivariate Gaussian:

$$p(\mathbf{y}_{1:m-1}, y|\mathbf{x}_{1:m}) = \mathcal{N}\left(\begin{bmatrix}\mathbf{y}_{1:m-1}\\ y\end{bmatrix}\middle|\mathbf{0}, \begin{bmatrix}\mathbf{K} & \mathbf{k}(\mathbf{x}_m)\\ \mathbf{k}(\mathbf{x}_m)^\intercal & k(\mathbf{x}_m, \mathbf{x}_m)\end{bmatrix}\right), \tag{10}$$

where $\mathbf{k}(\mathbf{x}_m) = [k(\mathbf{x}_1, \mathbf{x}_m), \ldots, k(\mathbf{x}_{m-1}, \mathbf{x}_m)]^\intercal$. Therefore, by using the classical formula of the conditional multivariate normal distribution based on the Schur complement, the predictive distribution is analytically solved as the following Gaussian distribution:

$$\begin{aligned} p(y|D_{1:m-1}, \mathbf{x}_m) &= p(y|\mathbf{y}_{1:m-1}, \mathbf{x}_{1:m}) \\ &\propto \mathcal{N}(y|\mu(\mathbf{x}_m), \sigma^2(\mathbf{x}_m)), \end{aligned} \tag{11}$$

where the mean parameter $\mu(\mathbf{x}_m)$ and variance parameter $\sigma(\mathbf{x}_m)$ are defined as:

$$\begin{aligned} \mu(\mathbf{x}_m) &\triangleq \mathbf{k}(\mathbf{x}_m)^\intercal \mathbf{K}^{-1} \mathbf{y}_{1:m-1}, \\ \sigma^2(\mathbf{x}_m) &\triangleq k(\mathbf{x}_m, \mathbf{x}_m) - \mathbf{k}(\mathbf{x}_m)^\intercal \mathbf{K}^{-1} \mathbf{k}(\mathbf{x}_m). \end{aligned} \tag{12}$$

Based on this predictive distribution, we can analytically obtain the expected improvement function $a^{\text{EI}}(\mathbf{x}_m)$ by substituting Eq. (11) into Eq. (3) as follows:

$$a^{\text{EI}}(\mathbf{x}_m) = \sigma(\mathbf{x}_m)\big(z(\mathbf{x}_m)\Phi(z(\mathbf{x}_m)) + \mathcal{N}(z(\mathbf{x}_m)|0, 1)\big), \tag{13}$$

where Φ is the cumulative distribution function of the standard normal distribution (i.e., $\Phi(x) = 1/\sqrt{2\pi}\int_{-\infty}^{x} e^{-t^2/2}dt$), and $z(\mathbf{x}_m)$ is a normalized average gain defined as follows:

$$z(\mathbf{x}_m) \triangleq (\mu(\mathbf{x}_m) - y_{m-1}^*)/\sigma_m(\mathbf{x}_m). \tag{14}$$

We can thus obtain $\hat{\mathbf{x}}_m$ by evaluating Eq. (13) numerically, which can be performed quickly for a small number of m thanks to the analytical expression.

As for the kernel function $k(\mathbf{x}, \mathbf{x}')$, [11] proposes to use the following kernel:

$$k(\mathbf{x}, \mathbf{x}') = \Big(1 + \sqrt{5s(\mathbf{x}, \mathbf{x}')} + \frac{5}{3}s(\mathbf{x}, \mathbf{x}')\Big)\phi_0 e^{-5s(\mathbf{x}, \mathbf{x}')}, \tag{15}$$

where $s(\mathbf{x}, \mathbf{x}')$ is the l^2 distance of $\mathbf{x}$ and $\mathbf{x}'$ with a metric ϕ_d:

$$s(\mathbf{x}, \mathbf{x}') = \sum_{d=1}^{D} \frac{(x_d - x_d')^2}{\phi_d^2}. \tag{16}$$

The coefficients $\{\phi_d\}_{d=0}^D$ are hyper-parameters in this kernel, and [11] marginalizes them out in obtaining the expected improvement function by using Markov Chain Monte Carlo (MCMC).

The basic algorithm of Bayesian optimization is shown in Algorithm 2. While for CMA-ES one needs to set initial values for $\mathbf{x}$, for Bayesian optimization one needs to set the domain of $\mathbf{x}$. Unlike CMA-ES, Bayesian optimization makes no assumption on the type of tuning parameters, and can handle continuous and discrete values without extra processing. Parallelization can be performed when computing the expected improvement function $a^{\text{EI}}(\mathbf{x}_m)$ with the Monte Carlo sampling. However, the greedy search resulting from Bayesian optimization often selects tuning parameters on the edges of the parameter domains $\mathcal{X}$ (e.g., large model complexities, small step sizes), which leads to extremely long function evaluations. This actually makes the ASR evaluation difficult in our experiments.

Table 1. WERs (%) for CMA-ES and Bayesian Optimization (BO) on Resource Management tasks **RM-ML** and **RM-NN**.

	CMA-ES	BO	Random	Human expert
RM-ML	1.56	1.56	1.63	1.92
RM-NN	1.42	N/A	N/A	1.54

Table 2. WERs (%) for CMA-ES on the WSJ-SI284 task (**WSJ-DT**)

	CMA-ES	Human expert
Dev.	7.52	7.65
Eval.	4.55	4.59

3. EXPERIMENTS

We performed multiple middle-vocabulary ASR experiments for the following task settings with different types of tuning parameters $\mathbf{x}$:

- **RM-ML:** Resource Management with ML training
  $\mathbf{x} = \{\#$ of HMM states, # of Gaussians$\}$
- **RM-NN:** Resource Management with neural networks
  $\mathbf{x} = \{\#$ of HMM states, # of Gaussians, # of hidden units, # of layer, initial learning rate, final learning rate$\}$
- **WSJ-DT:** Wall Street Journal with discriminative training
  $\mathbf{x} = \{\#$ of HMM states, # of Gaussians, # of UBM Gaussians, boosting factor, i-smoothing factor, learning rate$\}$
- **Reverb:** Reverb Challenge 2014 with ML training
  $\mathbf{x} = \{\#$ of HMM states, # of Gaussians, # of LDA left contexts, # of LDA right contexts$\}$

We used CMA-ES, as explained in Section 2.1, for most of the experiments because of its popularity in black box optimization [22], and because it proved to be easier to handle than Bayesian optimization: as discussed above, Bayesian optimization had the problematic tendency of trying tuning parameters that had too large model complexities and/or too small step sizes, resulting in extremely long function evaluations[2]. However, since the RM-ML task is relatively small, we were able to compare CMA-ES, Bayesian optimization (as described in Section 2.2), and random greedy search on that task. In the results, we listed for each task the best score y^* among all evaluations performed through Algorithms 1 and 2. We used the python version of CMA-ES[3] with some discretization for discrete tuning parameters in our implementation, and the Spearmint package for Bayesian optimization[4]. The ASR experiments were performed by using the Kaldi ASR toolkit [23], and followed the standard recipes in the toolkit for RM-ML, RM-NN, and WSJ-DT tasks. The human expert results in our experiments were basically obtained using the parameters as tuned in the recipes. The Reverb Challenge [24] is a reverberant speech enhancement and recognition challenge which is part of the IEEE SPS AASP challenge series based on the 5k WSJCAM0 task. We also used the Kaldi ASR toolkit for the multi-condition ASR task of the Reverb Challenge, with ML training, LDA feature transformation, and MLLR adaptation [25].

[2] Careful setting of the tuning parameter domains might solve the problem, but it would assume human expert knowledge.

[3] https://www.lri.fr/~hansen/cmaes_inmatlab.html

[4] https://github.com/JasperSnoek/spearmint

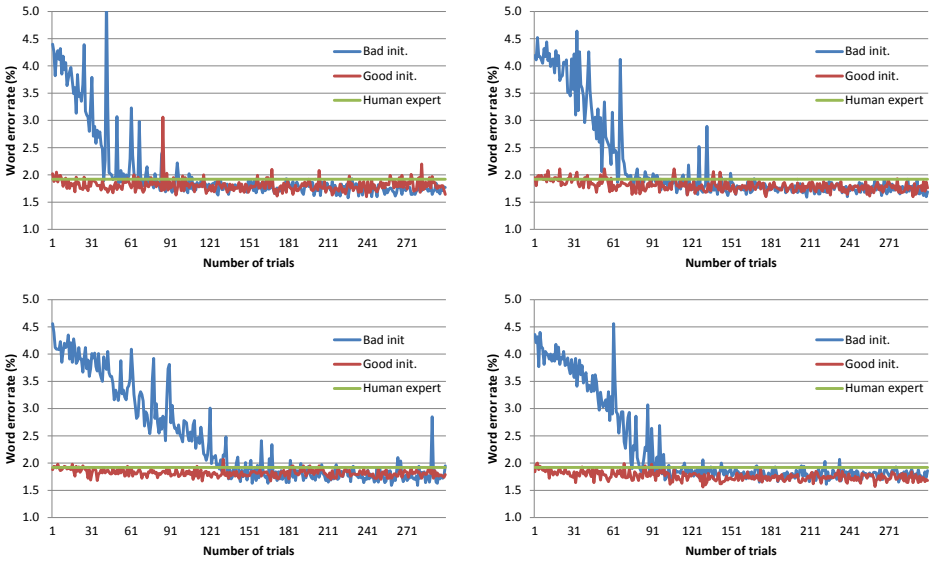

Fig. 1. CMA-ES results on the Resource Management ML training task (**RM-ML**) for bad and good initializations using four seeds.

Table 3. WERs (%) on the Reverb Challenge (**Reverb**).

	SimData							RealData		
	Room1		Room2		Room3		Ave.	Room1		Ave.
	Near	Far	Near	Far	Near	Far		Near	Far	
Baseline	13.27	17.08	20.80	36.83	23.54	39.44	25.16	47.91	46.55	47.23
LDA+fMLLR	10.82	13.30	14.81	28.40	16.25	31.16	19.12	43.11	39.71	41.41
CMA-ES	**10.37**	**13.20**	**13.48**	**25.46**	**15.03**	**27.30**	**17.47**	**36.68**	**34.11**	**35.40**

3.1. Robustness against initializations

The first aim of the experiments was to investigate the robustness of CMA-ES to the quality of the initialization. We used the RM-ML task, which is relatively small, to compare results with good and bad initializations of the tuning parameters with four random seeds for sampling ($\mathbf{x} \sim \mathcal{N}(\mathbf{x}|\hat{\boldsymbol{\theta}})$). The tuning parameters considered were the number of HMM states in the triphone clustering and the number of Gaussians, which are the main tuning parameters in HMM-based ASR [7]. Figure 1 shows that, even when using a bad initialization ($\mathbf{x} = (500, 1000)$), the performance converged to that obtained by the human expert and the good initialization ($\mathbf{x} = (1800, 9000)$) for all random seeds. Although many trials (from 60 to 150) were required for convergence, they were parallelized on 6 CPUs, thus dividing the wall clock time by 6 (roughly 17.5 hours on average on an Intel® Core™ i7-2600 CPU @ 3.40GHz machine). These results confirm the robustness of CMA-ES optimization with respect to tuning parameter initializations.

3.2. Robustness against task variations

The next experiments investigated the robustness of black box optimization to variations in the ASR tasks. Table 1 provides the word error rates for RM-ML and RM-NN tasks. In the RM-ML tasks, both CMA-ES and Bayesian optimization achieved comparable (actually better, but it is not statistically significant) performance to that of the human expert and random search. CMA-ES also achieved comparable performance to that of the human expert in the RM-NN task.

Since this kind of optimization runs the risk of over-tuning, which would result in degraded performance on an open evaluation set, we compare in Table 2 the CMA-ES and human expert results on both the development (dev93) and evaluation (eval92) sets of the WSJ 20k tri-gram task, where the parameters were tuned by only using the development set. CMA-ES again achieved comparable performance to that of the human expert on both sets, which empirically confirms the mitigation of a potential risk of over-tuning.

Finally, Table 3 compares results, for the multi condition ASR task of the Reverb Challenge, of the HTK baseline with adaptation [24], the Kaldi baseline with adaptation and LDA [25], and the Kaldi baseline with CMA-ES. Since reverberation corrupts speech features across several frames, we included the context sizes in LDA in the parameters tuned by CMA-ES. The results clearly show that CMA-ES optimization successfully tuned the previous and future context sizes as 7 and 5 frames respectively, largely improving reverberant speech recognition performance for SimData (19.12% $\rightarrow$ 17.47%) and RealData (41.41% $\rightarrow$ 35.40%).

This series of experiments confirmed the effectiveness of black box optimization, through which we were able to automatically obtain similar performance to fine-tuned systems designed by experts.

4. SUMMARY

This paper investigated the application of black box optimization techniques based on CMA-ES and Bayesian optimization to ASR parameter tuning. Multiple experiments show that the construction of ASR systems with black box optimization is robust against tuning initializations and task variations. This can reduce the effort of tuning ASR systems by experts, and accelerate the deployment of ASR for wider applications. Future work will develop black box optimization methods more specific to speech recognition to achieve faster convergences.

Acknowledgments

We thank Dr. Toshiaki Koike-Akino (MERL) for discussions on various black box optimization techniques, and Felix Weninger (Technische Universität München) for building the Kaldi ASR baseline for the Reverb Challenge task during his internship at MERL.

5. REFERENCES

[1] X. D. Huang, A. Acero, and H. W. Hon, *Spoken language processing, a guide to theory, algorithm, and system development*, Prentice Hall PTR, 2001.

[2] G. Saon and J-T Chien, "Large-vocabulary continuous speech recognition systems: A look at some recent advances," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 18–33, 2012.

[3] Y. Tachioka, S. Watanabe, J. Le Roux, and J. R. Hershey, "Discriminative methods for noise robust speech recognition: A CHiME challenge benchmark," in *Proc. International Workshop on Machine Listening in Multisource Environments (CHiME)*, 2013.

[4] 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.

[5] K. Shinoda and T. Watanabe, "Acoustic modeling based on the MDL criterion for speech recognition," in *Proc. Eurospeech*, 1997, vol. 1, pp. 99–102.

[6] S.S. Chen and P. S. Gopalakrishnan, "Clustering via the Bayesian information criterion with applications in speech recognition," in *Proc. ICASSP*, 1998, vol. 2, pp. 645–648.

[7] S. Watanabe, Y. Minami, A. Nakamura, and N. Ueda, "Variational Bayesian estimation and clustering for speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 12, pp. 365–381, 2004.

[8] Y. Zhang, P. Liu, J.T. Chien, and F. Soong, "An evidence framework for Bayesian learning of continuous-density hidden Markov models," in *Proc. ICASSP*, 2009, pp. 3857–3860.

[9] 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.

[10] E. Brochu, V. M. Cora, and N. De Freitas, "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv preprint arXiv:1012.2599*, 2010.

[11] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *arXiv preprint arXiv:1206.2944*, 2012.

[12] C. E. Rasmussen and C. K. I. Williams, "Gaussian processes for machine learning," 2006.

[13] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proc. NIPS*, 2011.

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

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

[16] 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.

[17] 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.

[18] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *Proc. ICASSP*, 2013, pp. 8609–8613.

[19] 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.

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

[21] D. Barber, *Bayesian reasoning and machine learning*, Cambridge University Press, 2012.

[22] 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.

[23] 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.

[24] K. Kinoshita, M. Delcroix, T. Yoshioka, T. Nakatani, E. Habets, R. Haeb-Umbach, V. Leutnant, A. Sehr, W. Kellermann, R. Maas, S. Gannot, and B. Raj, "The REVERB challenge: A common evaluation framework for dereverberation and recognition of reverberant speech," in *Proc. of WASPAA*, 2013.

[25] F. Weninger, S. Watanabe, Y. Tachioka, and B. Schuller, "Deep recurrent de-noising auto-encoder and blind de-reverberation for reverberated speech recognition," in *Proc. ICASSP*, 2014, (accepted).