Learning to Transfer Initializations for Bayesian Hyperparameter Optimization

Jungtaek Kim, Saehoon Kim,* and Seungjin Choi
Department of Computer Science and Engineering
Pohang University of Science and Technology
77 Cheongam-ro, Nam-gu, Pohang 37673, Korea
{jtkim,kshkawa,seungjin}@postech.ac.kr

Abstract

We propose a neural network to learn *meta-features* over datasets, which is used to select initial points for Bayesian hyperparameter optimization. Specifically, we retrieve k-nearest datasets to transfer a prior knowledge on initial points, where similarity over datasets is computed by learned meta-features. Experiments demonstrate that our learned meta-features are useful in optimizing several hyperparameters of deep residual networks for image classification.

1 Introduction

Hyperparameter optimization aims to find the best configuration of hyperparameters for a particular machine learning model, which typically requires many cross-validations on various combinations of hyperparameters. This is a crucial performance bottleneck of automated machine learning, because it is practically impossible to evaluate validation errors on every possible combination of hyperparameters under limited computational resources. To effectively reduce the search space, sequential model-based optimization [Hutter et al., 2011, Bergstra et al., 2011, Snoek et al., 2012] updates a regression model that maps hyperparameters into the performance of learning model to select a plausible hyperparameter sequentially. It still requires several initial points to build the regression model, which is commonly referred to as a cold-start problem.

Human experts on machine learning transfer their prior knowledge to resolve such cold-start problem by assuming that similar datasets in views of human experts typically have similar hyperparameters to achieve the best performance. Meta-learning [Schmidhuber, 1987, Thrun and Pratt, 1998, Chen et al., 2017] attempts to resolve that problem, because it consists of methods that effectively learn a model by transferring a prior knowledge from similar tasks. In the context of sequential model-based optimization, notable meta-learning methods can be categorized by two orthogonal approaches: (1) how to develop covariance functions to capture the shared information between tasks [Bonilla et al., 2008, Bardenet et al., 2013, Swersky et al., 2013] in the context of Gaussian process (GP) regression; (2) how to design hand-crafted and simple learned *meta-features* that describe similarity over datasets [Michie et al., 1994, Pfahringer et al., 2000, Feurer et al., 2015] to transfer a prior knowledge.

In this paper, we propose a meta-learning framework to find the best hyperparameter for a classifier by directly learning meta-features with a Siamese network [Bromley et al., 1994] where each network is composed of convolutional bi-directional long short-term memory network (LSTM) [Hochreiter and Schmidhuber, 1997] to generate feature vectors that describe datasets. We train the Siamese convolutional bi-directional LSTM by minimizing the difference between meta-feature distance and ground-truth target distance: (1) meta-feature distance is simply defined as Euclidean distance

*S. Kim is also affiliated with AItrics.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

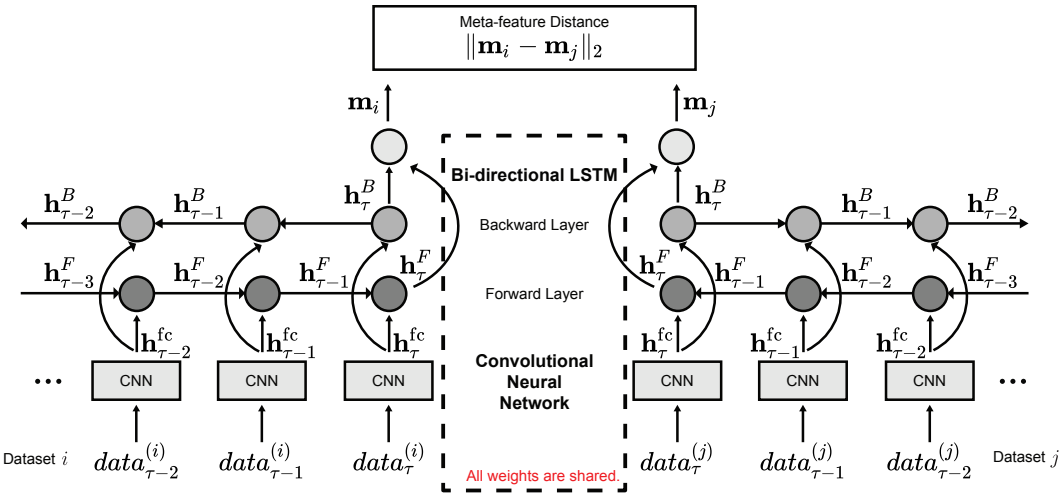


Figure 1: Our model with a Siamese architecture is composed of two identical convolutional bi-directional LSTMs.

between feature vectors obtained by the outputs of the identical networks in the Siamese network; (2) ground-truth target distance is measured by L_1 distance between two mappings from a hyperparameter space to classification accuracy.

2 Background

2.1 Hyperparameter Optimization

Suppose that we are given a dataset $\mathcal{D} = \{\mathcal{D}_{train}, \mathcal{D}_{val}\}$ (training and validation set) with which we train a model involving hyperparameters $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_n]^\top$. Assuming that $\theta_i \in \Theta_i$, the hyperparameter optimization searches the best configuration of hyperparameters over the space $\boldsymbol{\Theta} = \Theta_1 \times \cdots \times \Theta_n$. Given a dataset $\mathcal{D}$, the best hyperparameter configuration is determined by minimizing the validation error $\mathcal{J}(\boldsymbol{\theta}, \mathcal{D}_{train}, \mathcal{D}_{val})$. Earlier work on hyperparameter optimization is based on grid or random search [Bergstra and Bengio, 2012]. Recently various methods based on sequential model-based optimization have been proposed, including sequential model-based algorithm configuration (SMAC) [Hutter et al., 2011], Spearmint [Snoek et al., 2012], and tree-structured Parzen estimator (TPE) [Bergstra et al., 2011].

2.2 Sequential Model-based Optimization

Sequential model-based optimization (SMBO) referred to as Bayesian hyperparameter optimization (BHO) tries to find the best configuration of hyperparameters $\boldsymbol{\theta}^*$, when we are given a target function $\mathcal{J}(\boldsymbol{\theta}, \mathcal{D}_{train}, \mathcal{D}_{val})$ that returns validation error and a few different configurations of hyperparameters at initial design. BHO searches a minimum, gradually accumulating $(\boldsymbol{\theta}_t, \mathcal{J}(\boldsymbol{\theta}_t))$ with t increasing. Starting with a set of initial design $\{(\boldsymbol{\theta}_1, \mathcal{J}_1), \ldots, (\boldsymbol{\theta}_t, \mathcal{J}_t)\}$, a Gaussian process (GP) regression model $\mathcal{M}_{\text{GP}}$ fits to this set of examples. The GP regression model $\mathcal{M}_{\text{GP}}$ serves as a surrogate function that approximates the landscape of $\mathcal{J}$ over the space $\boldsymbol{\Theta}$. The surrogate function well approximates the regions exploited so far but has high uncertainty about the regions that are not yet explored. Thus, rather than optimizing the surrogate function itself, the acquisition function $a(\boldsymbol{\theta}|\mathcal{M}_{\text{GP}})$, which is constructed to balance a trade-off between exploitation and exploration, is optimized to select the next configuration of hyperparameters at which the validation error $\mathcal{J}$ is evaluated. Assuming that the current GP has mean $\mu(\boldsymbol{\theta})$ and variance $\sigma^2(\boldsymbol{\theta})$, two popular acquisition functions that we use in this paper are expected improvement (EI) [Mockus et al., 1978] and GP upper confidence bound (GP-UCB) [Srinivas et al., 2010]. Instead of using a zero mean function for GP, a suitable mean function could be useful to BHO in case that a small number of initial points is given.

3 Proposed Method: Meta-feature Learning with Siamese Architecture

We describe our model with a Siamese LSTM architecture in detail. As shown in Fig. 1, our model for meta-feature learning is composed of two identical deep neural networks that share the same set of weights. Each identical neural network is referred to as *wing*. Each wing of our model consists of CNNs (with two convolutional layers and two fully-connected layers) followed by the bi-directional LSTMs. We use the LSTM to learn the bi-directional characteristics in a set of image features.

Suppose that a pair of two image datasets $(\mathcal{D}^{(i)}, \mathcal{D}^{(j)})$ is given, where each dataset contains τ number of images, denoted by $data_1^{(k)}, data_2^{(k)}, \ldots, data_\tau^{(k)}$ for $k = i, j$. The goal is to learn meta-features $\mathbf{m}_i$ and $\mathbf{m}_j$ that encode the characteristics of datasets $\mathcal{D}^{(i)}$ and $\mathcal{D}^{(j)}$, respectively. Each image data is provided to the individual CNN which produces image features, fed into the bi-directional LSTM.

We train a set of weights and biases that are shared by two wings, such that the Euclidean distance $\|\mathbf{m}_i - \mathbf{m}_j\|_2$ between meta-features matches the target distance $d_{\text{target}}(\mathcal{D}^{(i)}, \mathcal{D}^{(j)})$ between datasets that should be provided. Performance measure over hyperparameters for dataset pair $(\mathcal{D}^{(i)}, \mathcal{D}^{(j)})$ has been observed preemptively, and the distance between two datasets is computed by comparing preemptively observed mappings from hyperparameters to performance measure via target distance function (target metric) $d_{\text{target}}(\cdot, \cdot)$ such as L_1 distance. More precisely, we compute L_1 distance of all pairwise configurations from two mappings to measure the target distance between datasets.

Algorithm 1 Meta-feature Learning over Datasets

Input: A set of n datasets $\{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$, target distance function $d_{\text{target}}(\cdot, \cdot)$, batch size $\beta \in \mathbb{N}$, step size $\tau \in \mathbb{N}$, number of iterations $T \in \mathbb{N}$
Output: Siamese LSTM model $\mathcal{M}_{\text{S-LSTM}}$ trained over $\{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$

```
Initialize M_S-LSTM.
for t = 1, 2, ..., T do
    Sample β different pairs of datasets, i.e., {(D_i, D_j)} for |i ≠ j| = β, i, j = 1, ..., n.
    Sample τ data points from each dataset in the pair {(D_i, D_j)} selected above, to make |D_i| = |D_j| = τ.
    Update parameters in M_S-LSTM using d_target(·, ·) and {(D_i, D_j)} via backpropagation.
end for
return M_S-LSTM
```

Algorithm 1 shows how meta-feature over datasets is learned. Inputs to Algorithm 1 are: (1) a set of n datasets $\{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$; (2) a target distance function that is mentioned above; (3) batch size β; (4) step size τ; (5) a number of iterations T. And it returns the learned Siamese LSTM model $\mathcal{M}_{\text{S-LSTM}}$. For given number of iterations T, β different pairs of datasets are sampled and τ data points from each dataset in the selected pairs are subsampled. Using those pairs that have same τ data points, the Siamese network, $\mathcal{M}_{\text{S-LSTM}}$ is trained end-to-end via backpropagation. The difference between the meta-feature distance and the target distance is minimized to optimize all shared weights in the Siamese bi-directional LSTM: $\mathcal{L}(\mathcal{D}^{(i)}, \mathcal{D}^{(j)}) = \left[d_{\text{target}}(\mathcal{D}^{(i)}, \mathcal{D}^{(j)}) - \|\mathbf{m}_i - \mathbf{m}_j\|_2\right]^2$.

4 Experimental Results

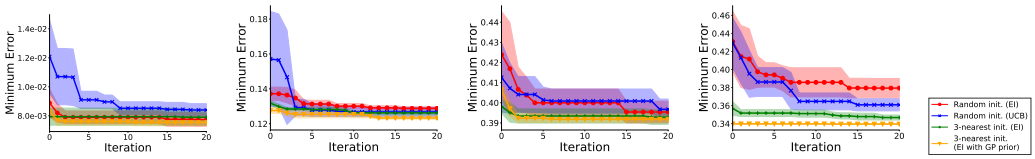
Before training the Siamese network for BHO, the mappings from certain hyperparameters to performance measure were measured. We used randomly subsampled datasets from MNIST, CIFAR-10, ImageNet 200 (for recent ILSVRC object detection challenges), and Places 205. In this paper, we measured classification accuracy with respect to batch size and initial learning rate for ResNet-26. The accuracy was measured from the subsampled datasets that have 5 classes and 2000 images per class. 20 scaled-down datasets for four image datasets were subsampled respectively, and 100 hyperparameter configurations were trained for each subsampled dataset.

After that, the Siamese bi-directional LSTMs were trained by the subsampled datasets. The pairs of all 80 datasets were used to train the Siamese convolutional bi-directional LSTM. For ground-truth labels, we computed L_1 distance between histograms that show classification accuracy with respect to hyperparameters. Given the trained Siamese network $\mathcal{M}_{\text{S-LSTM}}$, it is possible to measure the distance between test dataset and the training datasets by the learned meta-features. Then, k-nearest datasets were selected, and their associated histograms between hyperparameters and classification accuracy were employed in initializing BHO. After initializing with k hyperparameter configurations and prior

Algorithm 2 Bayesian Hyperparameter Optimization with Transferred Initial Points and GP Prior

Input: Learned Siamese LSTM model $\mathcal{M}_{\text{S-LSTM}}$, target function $\mathcal{J}(\cdot)$, limit $T \in \mathbb{N} > k$
Output: Best configuration of hyperparameters $\boldsymbol{\theta}^*$

Find k-nearest neighbors using the learned Siamese bi-directional LSTM, $\mathcal{M}_{\text{S-LSTM}}$.
Obtain k classification accuracy histograms over hyperparameters $\{\mathcal{H}_1, \ldots, \mathcal{H}_k\}$.
for $i = 1, 2, \ldots, k$ **do**
    Find the best configuration $\boldsymbol{\theta}_i$ on grid of the i-th histogram $\mathcal{H}_i$.
    Evaluate $\mathcal{J}_i = \mathcal{J}(\boldsymbol{\theta}_i)$.
end for
for $j = k+1, k+2, \ldots, T$ **do**
    $\mathcal{M} \leftarrow$ GP regression with the prior mean function $\frac{1}{k}\sum_{h=1}^{k} \mathcal{H}_h$ on $\{(\boldsymbol{\theta}_i, \mathcal{J}_i)\}_{i=1}^{j-1}$.
    Find $\boldsymbol{\theta}_j = \arg\max_{\boldsymbol{\theta}} a(\boldsymbol{\theta}|\mathcal{M})$.
    Evaluate $\mathcal{J}_j = \mathcal{J}(\boldsymbol{\theta}_j)$.
end for
return $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}_j \in \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_T\}} \mathcal{J}_j$



(a) MNIST (b) CIFAR-10 (c) CIFAR-100 (d) STL-10

Figure 2: Hyperparameter optimization for ResNet in the cases of four entire datasets: MNIST, CIFAR-10, CIFAR-100, and STL-10. Three initial points were given for all experiments, and all ResNets were trained for 50 epochs. Moreover, all experiments were repeated 5 times.

mean function of GP regression, we found the best candidate of hyperparameters for the machine learning model as applying Bayesian optimization. EI and GP-UCB criteria in Bayesian optimization were used to find the best hyperparameter configuration $\boldsymbol{\theta}^*$. These steps are described in Algorithm 2.

We implemented our method to initialize hyperparameter optimization based on GPyOpt [The GPyOpt authors, 2016] and GPflow [Matthews et al., 2017]. We employed Bayesian optimization with EI and GP-UCB criteria, as shown in Fig. 2. The entire datasets of MNIST, CIFAR-10, CIFAR-100, and STL-10 (subsampled from labeled images on ImageNet) were used to learn ResNet-26 for BHO. MNIST and CIFAR-10 were used to train the Siamese convolutional bi-directional LSTM. On the other hand, CIFAR-100 and STL-10 were not used to train the Siamese network. We conducted BHOs with (1) random initializations for EI; (2) random initializations for GP-UCB; (3) 3-nearest datasets initializations for EI; (4) 3-nearest datasets initializations for EI with given prior mean. Initializations of two hyperparameters, batch size and initial learning rate were set to k previous best hyperparameter configurations from each nearest dataset, and GP prior mean function was set to the average of the k-nearest mappings. A shown in Fig. 2, our methods outperformed rather than other BHOs with random initializations.

5 Conclusion

In this paper, we proposed the method to learn meta-features over datasets using the Siamese bi-directional LSTMs. We showed that the Siamese networks can train a distance function between two datasets that is learned to match the target distance. Furthermore, the k-nearest datasets determined by the learned networks can employ in initializing hyperparameter optimization.

Acknowledgments

A portion of this work was supported by NAVER. S. Kim was partly supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2017-0-01779, A machine learning and statistical inference framework for explainable artificial intelligence).

References

R. Bardenet, M. Brendel, B. Kégl, and M. Sebag. Collaborative hyperparameter tuning. In *Proceedings of the International Conference on Machine Learning (ICML)*, Atlanta, GA, USA, 2013.

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems (NIPS)*, volume 24, Granada, Spain, 2011.

E. V. Bonilla, K. M. A. Chai, and C. K. I. Williams. Multi-task Gaussian process prediction. In *Advances in Neural Information Processing Systems (NIPS)*, volume 21, Vancouver, Canada, 2008.

J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a "Siamese" time delay neural network. In *Advances in Neural Information Processing Systems (NIPS)*, volume 7, Denver, CO, USA, 1994.

Y. Chen, M. W. Hoffman, S. G. Colmenarejo, M. Denil, T. P. Lillicrap, M. Botvinick, and N. de Freitas. Learning to learn without gradient descent by gradient descent. In *Proceedings of the International Conference on Machine Learning (ICML)*, Sydney, Austrailia, 2017.

M. Feurer, J. T. Springerberg, and F. Hutter. Initializing Bayesian hyperparameter optimization via meta-leraning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, Austin, TX, USA, 2015.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proceedigns of the International Conference on Learning and Intelligent Optimization*, pages 507–523, Rome, Italy, 2011.

A. G. d. Matthews, M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18:1–6, 2017.

D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine learning, neural and statistical classification*. Ellis Horwood, 1994.

J. Mockus, V. Tiesis, and A. Zilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2:117–129, 1978.

B. Pfahringer, H. Bensusan, and C. Giraud-Carrier. Meta-learning by landmarking various learning algorithms. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 743–750, Stanford, CA, USA, 2000.

J. Schmidhuber. *Evolutionary Principles in Self-Referential Learning*. PhD thesis, Technical University of Munich, 1987.

J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, volume 25, Lake Tahoe, NV, USA, 2012.

N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the International Conference on Machine Learning (ICML)*, Haifa, Israel, 2010.

K. Swersky, J. Snoek, and R. P. Adams. Multi-task Bayesian optimization. In *Advances in Neural Information Processing Systems (NIPS)*, volume 26, Lake Tahoe, NV, USA, 2013.

The GPyOpt authors. GpyOpt: A Bayesian optimization framework in python, 2016. `https://github.com/SheffieldML/GPyOpt`.

S. Thrun and L. Pratt. *Learning to Learn*. Kluwer Academic Publishers, 1998.