Supplementary materials for: 'Structured Variationally Auto-encoded Optimization'

Xiaoyu Lu [1] **Javier González** [2] **Zhenwen Dai** [2] **Neil D. Lawrence** [2 3]

1. Implementation and Experiment Details

For each dataset in the automatic statistician experiments in Section 3, we evenly split the dataset into a training and a test set and use the test Mean Square Error (MSE) as the model selection criterion.

We used the same configuration of Structure Generating Variational Auto-encoder (SG-VAE) for the automatic statistician experiments. The configuration is as follows:

- Use two hidden layers with 400 hidden units in both the encoder and the decoder
- Use a five dimensional latent space.
- Use Adam as the optimizer with learning rate equals to 0.005.

The details of search in latent space is as follows:

- Use squared exponential kernel in the Gaussian Process to model the relationship between the model criterion and the latent space, with the acquisition function being the expected improvement.
- Use GPyOpt `http://github.com/SheffieldML/GPyOpt` as the tool to do the optimization.

1.1. Visualization and interpretation of the latent space of the SG-VAE in the airline dataset

We display the learned latent space of the SG-VAE for the airline dataset. Each dot represents a kernel combination learned by the VAE in the projected 2-dimensional latent space. We can observe the effect of the data-based representation to the latent space that is able to distinguish between complicated kernel combinations.

1.2. Description of the Baselines in the Comparison

We have compared SVO with others baselines in the experiment section, namely the Vanilla BO, random search in the original space and random search in the latent space, we hereby give the details of those methods.

Although vanilla BO is not well suited for the problems in our work since the input space in our experiments is categorical (types of kernels and operations) and hierarchical (the number of operations in the combination affects the length of the combination), which is one of the main motivations of our paper, we have included it as a baseline comparison. There are 12 variables, with one of them being the variable representing the number of operations of the combination which takes values in $\{1, 2, 3, 4, 5, 6\}$, 6 of them being categorical variables with 4 categories representing the kernels , and 5 of them being categorical variables with 2 categories representing the operations (addition or multiplication). The hyper-priors for sampling the parameters of the kernel combinations and the setup of the BO search is the same as SVO.

We also compared our method with random search, both in the original and the latent space. For random (original), we randomly select a kernel combination from the generated grammar representation; for random (latent), we randomly sample from the latent space and use the decoder of the learned VAE to decode into a feasible kernel combination.

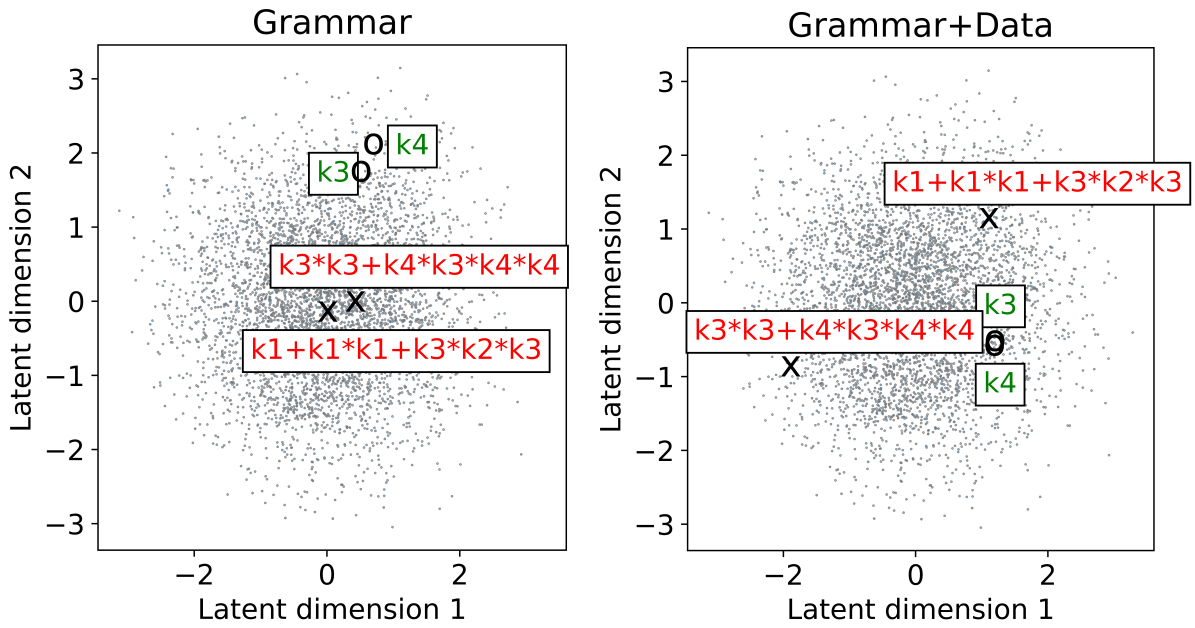

Figure 1. Visualization of the latent space of for the airline dataset. This is a 2D projection of the 5D latent space (a): latent space with the grammar kernel representation, (b): latent space with the data-based kernel representation.

1.3. Extra experiment on the milk production dataset

We conduct the experiment on an additional dataset where the setup is the same as the others. The dataset can be obtained from https://datamarket.com/data/set/22ox/monthly-milk-production-pounds-per-cow-jan-62-dec-75#!ds=22ox&display=line, which is a time series containing Monthly milk production in pounds per cow from Jan 62 to Dec 75 which 168 instances. It can be observed that the BO methods have better performance than the others, in particular, vanilla BO achieved the lowest MSE on this particular dataset where SVO without uncertainty had the fastest convergence.

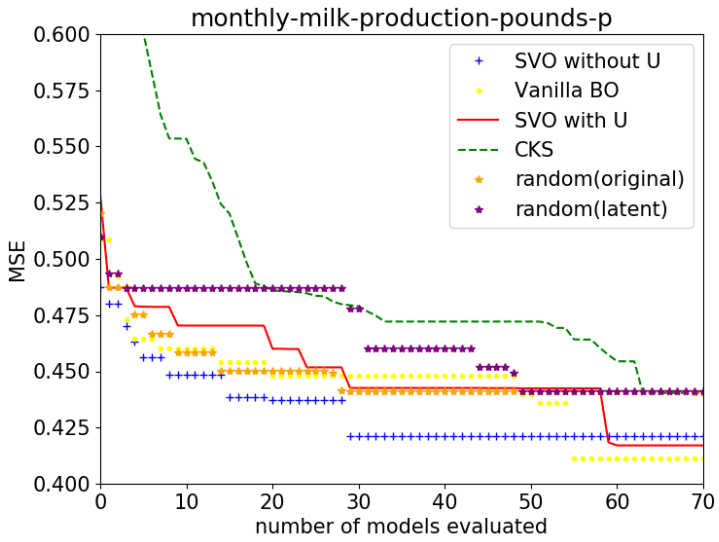

Figure 2. Experimental results on the monthly milk production dataset.

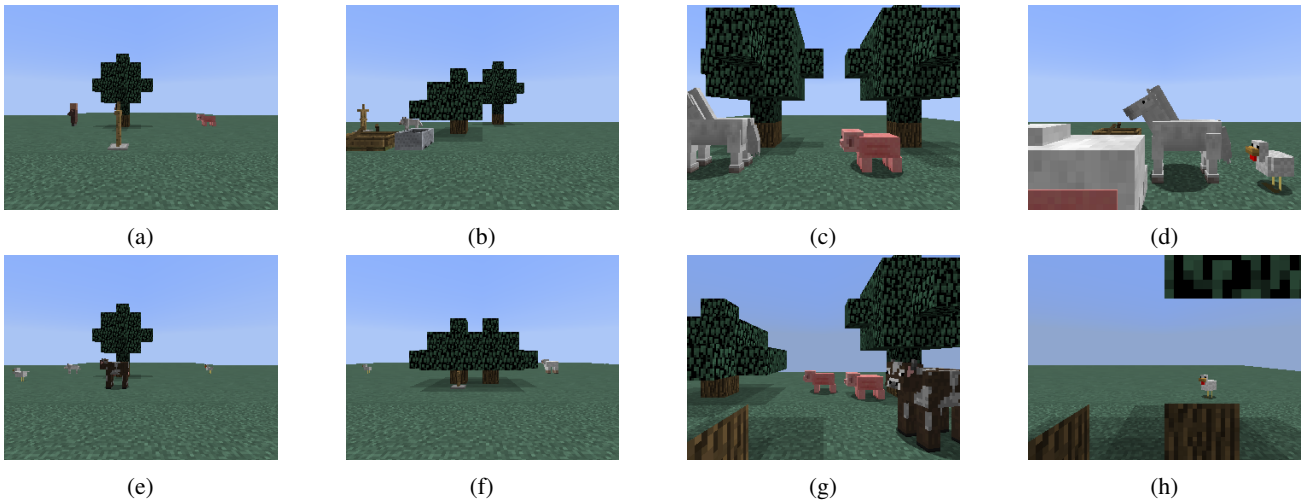

Figure 3. Use SVO to search for a XML configuration of the "Minecraft" engine to reproduce three target images (a), (b) and (c). The best configuration found by SVO are (e), (f) and (g) respectively. Images (d) and (h) were selected randomly to illustrate the complexity of the problem.

2. Natural Scene Understanding with Minecraft scenes

Understanding natural scene is a challenging task because of the almost infinite number of possible combinations of objects and large variation of their appearance on image due to various facts such as occlusions, variations in lighting conditions, reflection and refraction of object surface. On the other hand, we have sophisticated graphics rendering engines, which allow us to generate photorealistic images. Using computer graphics engine to improve scene understanding algorithms has been proposed recently in computer vision. We take a simplistic approach by formulating natural scene understanding as a search problem, i.e., given a natural image, whether we can find a configuration of a graphics rendering engine to reproduce the query.

Following the experiment setting of (Wu et al., 2017), we take "Minecraft" as the rendering engine and use XML as the description of configuration. We consider 12 Minecraft objects: pig, cow, sheep, chicken, wolf, horse, villager, armor stand, boat, Minecraft rideable and two types of trees. We consider two attributes of each object, i.e., the location of an object on the image parametrized by the distance to the camera and the 1D angle from right to left. Both attributes are discretized into five different values. The maximum number of objects in a scene is limited to six in our experiments, but can easily be extended to higher. The task is to search for the XML configuration of which the generated image matches the target image as close as possible. For simplicity, we use L2 norm to measure the dissimilarity between a generated image and the target image, but more sophisticated dissimilarity measure can be used here.

The target images are shown in Figure 3a, 3b, 3c. We apply SVO to this problem by defining a structure generation process like the one in Section 2.3. We consider two types of operations: adding a new object and stop. When adding a new object, we choose its object type and the values of two attributes. With this generation process, we generated 20,000 configurations for training the SG-VAE. We used the same configuration of SG-VAE as described above. We adapted the definition of likelihood for the grammar of Minecraft scene generation. Note that we only use the generated grammar-based representation but not include any data term, for simplicity and efficiency. Then, we apply the search algorithm described in Section 2.4 to look for the best configuration. We used Gaussian process as the surrogate model with squared exponential kernel. We use only the mean prediction of encoder for Bayesian Optimization search for efficiency. After 100 iterations, the image generated by the best configuration is shown in Figure 3e, 3f, 3g. The generated images generally match with the target ones with small disparity. Note that our approach is significantly different from (Wu et al., 2017). Wu et al. (2017) trained neural networks to directly predict a configuration given a target image. This requires generation of tens of thousands images from the rendering engine for training the neural networks. The number of required training images can easily go a few magnitudes higher if more objects or attributes are considered. SVO is a meta-approach for this problem, where we *do not* learn a model to directly analyze images. Instead, we build a SG-VAE for configurations and search in the latent space of configurations for a matching explanation of the target image. Our approach only requires to generate a few images to guide the search.