Stochastic Hyperparameter Optimization with Hypernets

Jonathan Lorraine, David Duvenaud
University of Toronto

Main Idea

- Machine learning models often nest optimization of model weights in the optimization of hyperparameters.
- We collapse the nested optimization into joint optimization by training a neural network to output optimal weights for each hyperparameter.
- The method converges to locally optimal weights and hyperparameters for large hypernets and effectively tunes thousands of hyperparameters.

Hyperparameter Tuning is Nested Optimization

- Selecting a hyperparameter is finding a solution to the following bi-level optimization problem:
  \[ \min_{\lambda} \mathbb{E}_{\tau} \left( \min_{w} \mathbb{E}_{\tau} \left[ L(w, \lambda) \right] \right) \]  
  (1)
- The optimized model weights depend on the choice of hyperparameter. This is a best-response function of the weights to the hyperparameters:
  \[ w^{\ast}(\lambda) = \min_{w} \mathbb{E}_{\tau} \left[ L(w, \lambda) \right] \]  
  (2)

Learning a Mapping from Hyperparameters to Optimal Weights

- A hypernet is a neural network which outputs network weights.
- The best-response takes hyperparameters and outputs weights, so approximate it with a hypernet.

**Theorem.** Sufficiently powerful hypernets can learn continuous best-response functions, which minimize the expected loss for any hyperparameter distribution.

There exists \( \phi^* \), such that for all \( \lambda \in \text{support}(p(\lambda)) \),

\[ L^\phi(w_\lambda(\lambda), \lambda) = \min_{\lambda} L(w, \lambda) \]

and \( \phi^* = \min_{\lambda} \mathbb{E}_{\tau} \left[ L(w_\lambda(\lambda), \lambda) \right] \)

Globally Optimizing the Hypernet

- We can learn the best-response without viewing pairs of hyperparameters and optimized weights, by substituting the hypernet output into the training loss. The algorithm is denoted Hyper Training.

```
1. initialize \( \phi \)
2. initialize \( \lambda \)
3. for \( T_{\text{hyper}} \) steps do
   a. \( \mathbf{x} \sim \text{Training data}, \lambda \sim p(\lambda) \)
   b. \( \phi = \phi - \alpha \nabla \phi \mathbb{E}_{\tau} L(w_\phi(\lambda), \lambda) \)
4. for \( T_{\text{test}} \) steps do
   a. \( \mathbf{x} \sim \text{Validation data} \)
   b. \( \hat{\lambda} = \lambda - \beta \nabla \phi \mathbb{E}_{\tau} L(w_\phi(\lambda)) \)
   c. return \( \hat{\lambda}, w_\phi(\hat{\lambda}) \)
```

Locally Optimizing the Hypernet

- It is difficult to learn the best-response globally due to finite network size and training time.
- It is easier to learn the best-response locally, update the hyperparameters and repeat.

```
1. initialize \( \phi, \hat{\lambda} \)
2. for \( T_{\text{test}} \) steps do
   a. \( \mathbf{x} \sim \text{Training data}, \lambda \sim p(\lambda) \)
   b. \( \phi = \phi - \alpha \nabla \lambda \mathbb{E}_{\tau} L(w_\lambda(\lambda), \lambda) \)
   c. \( \mathbf{x} \sim \text{Validation data} \)
   d. \( \lambda = \lambda - \beta \nabla \phi \mathbb{E}_{\tau} L(w_\phi(\lambda)) \)
   e. return \( \lambda, w_\phi(\lambda) \)
```

Optimizing 7,850 Hyperparameters

- We investigate our methods performance on tuning hyperparameters of dimensionality 10 and 7,850.

![Figure 2: Training and validation loss of a neural net for linear regression on MNIST, estimated by cross-validation (crosses) or by a hypernet (line), which outputs 7,850-dimensional network weights. The training and validation loss can be cheaply evaluated at any hyperparameter value using a hypernet. Standard cross-validation requires training from scratch each time. Left: A global approximation the best-response. Right: A local approximation to the best-response.](image)

![Figure 4: A visualization of exact (blue) and approximate (red) optimal weights as a function of given hyperparameters. Left: The training loss surface. Right: The validation loss surface. The approximately sampled hypernet \( \phi^* \) optimizes the test loss with a smooth, predictable, approximately optimal hypernet \( \phi^* \).](image)

Benefits of Hyper Training

- Our method provides two potential benefits. These are a better inductive bias by learning the weights instead of loss, and viewing many hyperparameter settings during training.
- We analyze this by comparing our algorithm to Bayesian optimization with 25 samples and a hypernet trained on the same 25 samples.

Conclusions

- We presented an algorithm that efficiently learns a differentiable approximation to a best-response for hyperparameter optimization.
- Hypernets can provide a better inductive bias for hyperparameter optimization than Bayesian optimization.

![Figure 3: Comparing three approaches to predicting validation loss. First row: A Gaussian process, fit on a small set of hyperparameters and the corresponding validation losses. Second row: A hypernet, fit on the same small set of hyperparameters and the corresponding optimized weights. Third row: Our proposed method, a hypernet trained with stochastically sampled hyperparameters.](image)