Meta-Learning Update Rules for Unsupervised Representation Learning

Authors: Luke Metz, Niru Maheswaranathan, Brian Cheung, Jascha Sohl-Dickstein (Google Brain, UC Berkeley)

Presenter: Bang An

10/02/2019
Core Idea

A novel meta-learning approach to unsupervised representation learning by meta-learning an *unsupervised learning rule* which leads to representations useful for downstream tasks

- The unsupervised learning rule is neuron-local, which enables it to generalize to different neural network architectures, datasets, and data modalities
Motivation

• A major goal of unsupervised learning is to discover data representations that are useful for subsequent tasks, without access to supervised labels during training.
• One problem of current approaches is that unsupervised representation learning algorithms are typically mismatched to the target task. Many current unsupervised objectives optimize for objectives such as log-likelihood of a generative model or reconstruction error, producing useful representations only as a side effect.
• Unsupervised representation learning seems uniquely suited for meta-learning (learning to learn).

• In this work, they propose to meta-learn an unsupervised update rule by meta-training on a meta-objective that directly optimizes the utility of the unsupervised representation (e.g. on semi-supervised classification task).
• Instead of learning the feature, they learn the learning rule.
Related Work—Unsupervised Representation Learning

- Autoencoders
- Generative adversarial networks
- Techniques rely on feature space design such as clustering
- Techniques rely on manually-defined desirable properties of latent representation

In contrast to this work, each method imposes a \textit{manually} defined training algorithm or loss function whereas this work \textit{learns} the algorithm that creates useful representations as determined by a meta-objective.
Related Work- Meta Learning

• Most meta-learning algorithms consist of two levels of learning, or ‘loops’ of computation:
  • an *inner loop*, where some form of learning occurs (e.g. an optimization process)
  • an *outer loop* or *meta-learning loop*, which optimizes some aspect of the inner loop, parameterized by *meta-parameters*.

• The performance of the inner loop computation for a given set of meta-parameters is quantified by a *meta-objective*. Meta-training is then the process of adjusting the meta-parameters so that the inner loop performs well on this meta-objective.
<table>
<thead>
<tr>
<th>Method</th>
<th>Inner loop updates</th>
<th>Outer loop updates, meta-parameters</th>
<th>Generalizes to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hyper parameter optimization Jones (2001); Snoek et al. (2012); Bergstra et al. (2011); Bergstra and Bengio (2012)</td>
<td>many steps of optimization</td>
<td>optimization hyper-parameters</td>
<td>test data from a fixed dataset</td>
</tr>
<tr>
<td>Neural architecture search Stanley and Miikkulainen (2002); Zoph and Le (2017); Baker et al. (2017); Zoph et al. (2018); Real et al. (2017)</td>
<td>supervised SGD training using meta-learned architecture</td>
<td>architecture</td>
<td>test loss within similar datasets</td>
</tr>
<tr>
<td>Task-specific optimizer (eg for quadratic function identification) (Hochreiter et al., 2001)</td>
<td>adjustment of model weights by an LSTM</td>
<td>LSTM weights</td>
<td>similar domain tasks</td>
</tr>
<tr>
<td>Learned optimizers Jones (2001); Maclaurin et al. (2015); Andrychowicz et al. (2016); Chen et al. (2016); Li and Malik (2017); Wichrowska et al. (2017); Bello et al. (2017)</td>
<td>many steps of optimization of a fixed loss function</td>
<td>parametric optimizer</td>
<td>new loss functions (mixed success)</td>
</tr>
<tr>
<td>Prototypical networks Snell et al. (2017)</td>
<td>apply a feature extractor to a batch of data and use soft nearest neighbors to compute class probabilities</td>
<td>weights of the feature extractor</td>
<td>new image classes within similar dataset</td>
</tr>
<tr>
<td>MAML Finn et al. (2017)</td>
<td>one step of SGD on training loss starting from a meta-learned network</td>
<td>initial weights of neural network</td>
<td>new goals, similar task regimes with same input domain</td>
</tr>
<tr>
<td>Evolved Policy Gradient Houthooft et al. (2018)</td>
<td>performing gradient descent on a learned loss</td>
<td>parameters of a learned loss function</td>
<td>new environment configurations, both in and not in meta-training distribution.</td>
</tr>
<tr>
<td>Method</td>
<td>Inner loop updates</td>
<td>Outer loop updates, meta-parameters</td>
<td>Generalizes to</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-------------------------------------------------------------</td>
<td>-------------------------------------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>Few shot learning (Vinyals et al., 2016; Ravi and Larochelle, 2016; Mishra et al., 2017)</td>
<td>application of a recurrent model, e.g. LSTM, Wavenet.</td>
<td>recurrent model weights</td>
<td>test loss on training tasks</td>
</tr>
<tr>
<td>Meta-unsupervised learning for clustering Garg (2018)</td>
<td>run clustering algorithm or evaluate binary similarity function</td>
<td>clustering algorithm + hyperparameters, binary similarity function</td>
<td>empirical risk minimization</td>
</tr>
<tr>
<td>Learning synaptic learning rules (Bengio et al., 1990; 1992)</td>
<td>run a synapse-local learning rule</td>
<td>parametric learning rule</td>
<td>supervised loss, or similarity to biologically-motivated network</td>
</tr>
<tr>
<td><strong>Our work — metalearning for unsupervised representation learning</strong></td>
<td><strong>many applications of an unsupervised update rule</strong></td>
<td><strong>parametric update rule</strong></td>
<td><strong>few shot classification after unsupervised pre-training</strong></td>
</tr>
</tbody>
</table>
Model Design

Inner loop:

- **base model**: MLP with parameter $\phi_t$
- train the base model via iterative apply *learned update rule*.

In standard supervised learning:

- the ‘learned’ optimizer is SGD
- the parameter $\phi_t$ is updated iteratively by performing SGD using the gradient $\frac{\partial l(x,y)}{\partial \phi_t}$
- the update rule is:
  $$\phi_{t+1} = \text{SupervisedUpdate}(\phi_t, x_t, y_t; \theta)$$

In unsupervised learning:

- the learned update is a parametric function
  $$\phi_{t+1} = \text{UnsupervisedUpdate}(\phi_t, x_t; \theta)$$
Model Design

Inner loop:
- train the base model via iterative apply learned update rule.
  \[ \phi_{t+1} = \text{UnsupervisedUpdate}(\phi_t, x_t; \theta) \]

Outer loop:
- train these meta-parameters by performing SGD on the sum of the MetaObjective over the course of (inner loop) training in order to find optimal parameters \( \theta^* \), that minimize the meta-objective over a distribution of training tasks

\[ \theta^* = \arg\min_{\theta} \mathbb{E}_{\text{task}} \left[ \sum_t \text{MetaObjective}(\phi_t) \right] \]
Model Design - Base Model

Base model:
- MLP
- We call the pre-nonlinearity activations $z^1, \ldots, z^L$ and post-nonlinearity activations $x^1, \ldots, x^L$
- parameters:
  \[
  \phi = \{W^1, b^1, V^1, \ldots, W^L, b^L, V^L\}
  \]
  where
  \[
  W^l \text{ and } b^l : \text{the weights and biases for layer } l
  \]
  \[
  V^l : \text{the corresponding weights used in the backward pass}
  \]
Model Design - Learned update rule

- Wish the update rule to generalize across architectures
  - design the update rule to be neuron-local
    (a function of pre- and post-synaptic neurons)
- Each neuron $i$ in every layer $l$ has an MLP (update network) with output:
  $$ h^l_b i = \text{MLP} \left( x^l_b i, z^l_b i, V^{l+1}, \delta^{l+1}, \theta \right) $$
  where
  - $b$: is the index of mini batch
  - $(x^l & z^l)$: feedforward activation
  - $V^l$: feedback weight
  - $\delta^l$: error signal

All update networks share meta-parameters $\theta$
Model Design - Learned update rule

\[ h^l_{bi} = \text{MLP} \left( x^l_{bi}, z^l_{bi}, V^{l+1}, \delta^{l+1}, \theta \right) \]

\[ \delta_{bi} = \text{lin} \left( h^l_{bi} \right) \]

\[ \Delta W^l_{ij} = \text{func} \left( h^l_{bi}, h^{l-1}_{bj}, W_{ij} \right) \]
Model Design - Meta Objective

- The *meta-objective* in this work is based on fitting a linear regression to labeled examples with a small number of data points.
- In order to encourage the learning of features that generalize well, we estimate the linear regression weights on one minibatch \( \{x_a, y_a\} \) of \( K \) data points, and evaluate the classification performance on a second minibatch \( \{x_b, y_b\} \) also with \( K \) data points.

\[
\hat{v} = \arg\min_v \left( \|y_a - v^T x_a^L\|^2 + \lambda \|v\|^2 \right)
\]

(features extracted using the learned base model)

\[
\text{MetaObjective}(\cdot; \phi) = \text{CosDist} \left( y_b, \hat{v}^T x_b^L \right)
\]

\[
\theta^* = \arg\min_\theta \mathbb{E}_{\text{task}} \left[ \sum_t \text{MetaObjective}(\phi_t) \right]
\]
Training

Meta-training distribution is composed of both datasets and base model architectures

Training:
- CIFAR10
- Subsets of Imagenet
- Rendered fonts dataset

Evaluation:
- MNIST
- Fashion MNIST
- IMDB (movie reviews, binary text classification)
- Subsets of Imagenet

- Sample the base model architecture.
- Sample number of layers uniformly between 2-5
- Sample number of units per layer logarithmically between 64 to 512

Distributed implementation:
- 512 workers (CPUs), each trains on one task by first sampling a dataset, architecture, and a number of training steps.
- Training takes ~8 days, and consists of ~200 thousand updates to \( \theta \) with minibatch size 256
Experimental Results

• Generalizing over datasets and domains
  • Left: compare performance on few shot classification with 10 examples per class
  • Right: train the update rule on image and evaluate on text (IMDB)

The goal of this work is to learn a general purpose unsupervised representation learning algorithm.
Experimental Results

- Generalizing over network architectures
  - Left: the learned update rule is capable of optimizing base models with hidden sizes and depths outside the meta-training regime
  - Right: the learned update rule generalizes across many different activation functions not seen in training
Experimental Results

• How it learns and how it learns to learn

Left: first layer base model receptive fields produced by the learned update rule over the course of meta-training

Right: Visualization of learned representations before (left) and after (right) training a base model with the learned update rule.

The learned model is capable of manipulating and partially separating the data manifold in a purely unsupervised manner.

template-like for MNIST, and local-feature-like for CIFAR10

The learned model is capable of manipulating and partially separating the data manifold in a purely unsupervised manner.
Takeaway

A novel meta-learning approach to unsupervised representation learning by meta-learning an *unsupervised learning rule* which leads to representations useful for downstream tasks

- The unsupervised learning rule is neuron-local, which enables it to generalize to different neural network architectures, datasets, and data modalities
Q & A
Meta-Learning Update Rules for Unsupervised Representation Learning

a.) Meta-Training / Outer Loop

b.) Meta-Objective Calculation

c.) Inner Loop Training – No Labels Used

d.) Forward and Backward Pass Details

e.) Parameterization of the Weight Updates