Meta-learning update rules for unsupervised representation learning

Vadym Kalashnykov

Higher School of Economics

5 декабря 2019 г.

Overview

1. Unsupervised representation learning
2. Meta-Learning Architecture
3. Implementation details
4. Experimental results

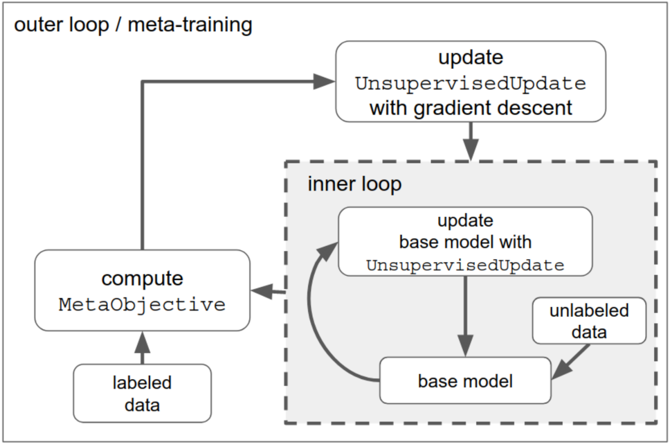
Unsupervised representation learning

How can we construct useful low-dimensional representations?

- Autoencoders / reconstruction loss
- GANs / adversarial loss
- Clustering

What is meta-learning

- Inner loop (optimize base model wights)
- Outer loop (optimize meta-parameters of the inner loop)



What is meta-learning

Inner loop:

- UnsupervisedUpdate (parameterized by θ)
- Unlabeled data
- Base Model (parameterized by ϕ)

Outer loop:

- MetaObjective
- Labeled data

Core ideas

- Meta-learn an unsupervised representation learning update rule
- Treat the creation of the unsupervised update rule as a transfer learning problem
- Try to generalize across input data modalities, datasets, permutation of the input dimensions, neural network architectures

What is meta-learning

Base model: MLP with parameters ϕ_t
In supervised learning the 'learned' optimizer is SGD variation

- t - the inner-loop iteration
- $\phi_{t+1} = \text{ SupervisedUpdate } (\phi_t, x_t, y_t; \theta)$
- θ - the meta-parameters of the optimizer (learning rate, momentum)

Instead consider parametric function which does not depend on label information

- $\phi_{t+1} = \text{ UnsupervisedUpdate } (\phi_t, x_t; \theta)$

What is meta-learning

- Train UnsupervisedUpdate with SGD
- ϕ_t is a function of θ since θ affects the optimization trajectory.

$$\theta^* = \underset{\boldsymbol{\theta}}{\text{argmin}} \mathbb{E}_{\text{task}} \left[\sum_t \text{MetaObjective } (\phi_t) \right]$$

Meta Objective

- We want objective to be differentiable
- Fit a linear regression to labeled examples
- Estimate the linear regression weights on one minibatch $\{xa, ya\}$ of K data points, and evaluate the classification performance on a second minibatch $\{xb, yb\}$ also with K datapoints
- Use cosine distance to improve stability
- $\hat{v} = \underset{v}{\operatorname{argmin}} \left(\left\| y_a - v^T x_a^L \right\|^2 + \lambda \|v\|^2 \right)$
- The latest can be solved in a closed form!
- $\text{MetaObjective}\ (\cdot; \phi) = \text{CosDist}\left(y_b, \hat{v}^T x_b^L\right)$
- Meta-objective is only used during meta-training

Base Model

- FC, BatchNorm, ReLU
- No inductive bias
- The parameters are $\phi = \left\{W^1, b^1, V^1, \cdots, W^L, b^L, V^L\right\}$
- V^l is used on backward pass

Learned update rule

UnsupervisedUpdate

- $x^0 \sim \mathcal{D}, x^0 \in \mathbb{R}^{B,N^0}$
- $z^l = \text{ BatchNorm } \left(x^{l-1}W^l\right) + b^l$
- $x^l = \text{ReLU}\left(\mathrm{z}^l\right)$
- $h^l_b i = \text{MLP}\left(x^l_b i, z_b i^l, V^{l+1}, \delta^{l+1}; \theta\right)$, θ are shared
- $\delta^l_{bi} = \text{lin}\left(h^l_{bi}\right)$
- $\Delta W^l_{ij} = \text{func}\left(h^l_{bi}, h^{l-1}_{bj}, W_{ij}\right)$
- Generalize across architectures and model topologies
- Neuron-local

Training the update rule

How to optimize θ?

- Training and computing derivatives through long recurrent computation is notoriously difficult
- Approximate the gradients via truncated backprop through time
- Batch norm, restricting the norm of the UnsupervsedUpdate step
- Hard implementation and engineering task

Training the update rule

Datasets

- Meta-training distribution is composed of both datasets and base model architectures.
- Train on CIFAR10 / Imagenet classification / rendered fonts
- Increase training dataset variation to improve the meta-optimization process.
- Restrict the input data to 16x16 pixels

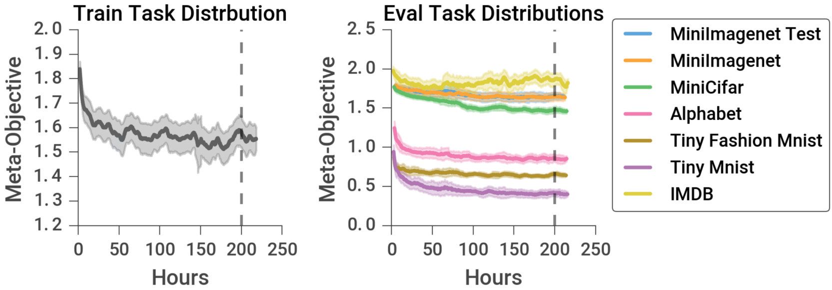
Training the update rule

Datasets

- Permute all inputs along the feature dimension
- Add augmentation coefficients as additional regression targets for the meta-objective
- Evaluate on MNIST, Fashion MNIST, IMDB, unseen Imagenet classes
- Sample the base model architecture
- distributed TensorFlow, 512 workers, 8 days

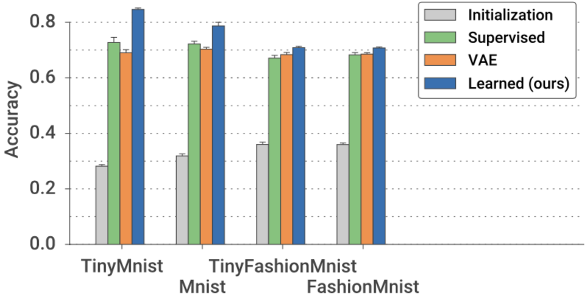
It's alive!

Compute a rolling average of the MetaObjective averaged across all datasets and model architectures:



Generalization over datasets

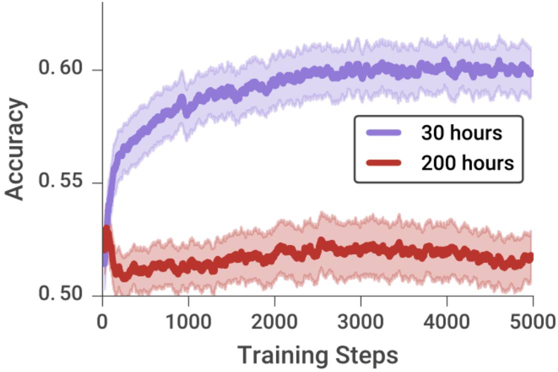
Performance on unseen dataset with different resolutions $(14 \times 14, 28 \times 28)$



Generalization over domain

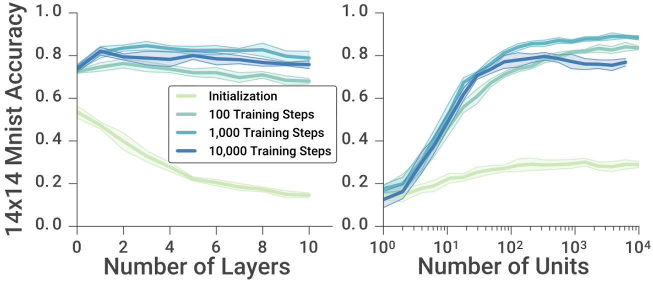
Test on task from different domain:

- dataset: IMDB movie reviews
- binary text classification
- "meta-overfits" to the image domain when trainded for 200+ hours



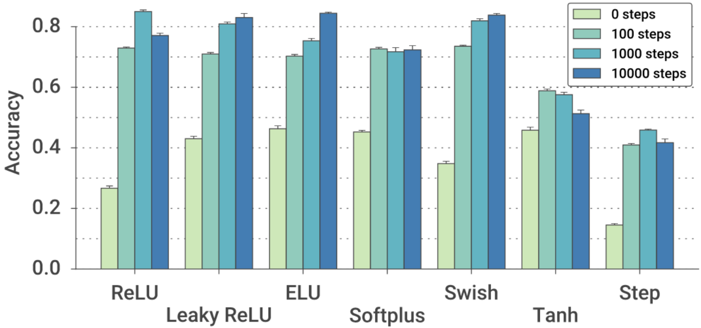
Generalization over architectures

- trained with 64 to 512 units per layer; tested with up to 10,000
- trained with 2 to 5 layers; tested with up to 11 layers



Generalization over architectures

- trained with ReLU activation only
- tested with LReLU, ELU, Tanh, Step etc.



TL;DR

- We meta-learned an unsupervised representation learning update rule

Sources

- Luke Metz, Niru Maheswaranathan, Brian Cheung, Jascha Sohl-Dickstein, et al. "Meta-Learning Update Rules for Unsupervised Representation Learning"preprint arXiv:1804.00222 (2018).

Questions

- Опишите формулу для meta-objective.
- Что означает понятие "обобщающей способности"в контексте предложенного алгоритма обучения? Кратно описать аргументы/эксперименты авторов статьи.
- В чем принципиальные отличия схемы обучения из статьи по сравнению с известными unsupervised representation learning подходами? (насколько вообще корректен этот вопрос? :)
- Опишите схему обновления весов базовой модели из статьи. Как она соотносится с vanilla SGD?