Supervised Learning of Unsupervised Learning Rules

Luke Metz[1], Brian Cheung[2], and Jascha Sohl-dickstein[1]

[1]Google Brain
[2]Berkeley
{lmetz, jascha}@google.com, bcheung@berkeley.edu

1 Introduction

Supervised learning has proven extremely effective for many problems where large amounts of labeled training data are available. There is a common hope that unsupervised learning will prove similarly powerful in situations where labels are expensive or impractical to collect, or where the prediction target is unknown during training. However, unsupervised learning has yet to fulfill this promise. One explanation for this failure is that unsupervised training rules are typically mismatched to the target task. Ideally, learned representations should linearly expose high level attributes of data (e.g. object identity) and perform well in semi-supervised settings. However, many current unsupervised objectives optimize for objectives such as log-likelihood of a generative model or reconstruction error, and produce useful representations only as a side effect.

Unsupervised representation learning seems uniquely suitable for metalearning (Hochreiter et al., 2001; Schmidhuber, 1995). Unlike most tasks to which metalearning has been applied, in unsupervised representation learning we are unable even to directly express the desired objective function. However, it is possible to directly express a *meta-objective* that captures the quality of an unsupervised update rule. In this work, we propose to *meta-learn* an unsupervised update rule, by meta-training on a meta-objective that directly rewards the usefulness of the unsupervised representation. Unlike hand-tuned proxies, this meta-objective directly measures the usefulness of a representation generated from unlabeled data.

By re-casting unsupervised learning as metalearning, we treat creating the unsupervised update rule as a transfer learning problem. Unlike previous work which transfers feature extractors between different data domains, we transfer the learning rule between different domains, and between different network architectures. Instead of learning transferable features, we learn a transferable learning rule which does not depend on *labels or dataset*.

Much previous work on meta-learning using gradient descent on the meta-objective has focused on improving the optimization process. Maclaurin et al. (2015); Andrychowicz et al. (2016); Ravi & Larochelle (2016); Wichrowska et al. (2017) learn an update rule that ‘ We leverage their insights here. By treating the entire learning process as a differentiable function of the loss, derivatives with respect to the update rule can be computed which in turn allow parameters within the update rule to be updated with backpropagation.

2 Methods

We consider a multilayer perceptron (MLP) $f(\cdot;\phi_t)$, with parameters ϕ_t, as the *base model* which an update rule targets. In standard supervised learning, that update rule is Stochastic Gradient Descent (SGD). A supervised loss $l\left(x,y\right)$ is associated with this model, where x is a minibatch of inputs, and y are the corresponding labels. The parameters ϕ_t are then updated iteratively until convergence, by performing SGD using the gradient $\frac{\partial l(x,y)}{\partial \phi_t}$. This supervised update rule can be written as

$$\phi_{t+1} = SupervisedUpdate(\phi_t, x_t, y_t; \theta), \tag{1}$$

where θ are the parameters of the optimizer (e.g. learning rate), which we will refer to as the meta-parameters.

In this work, we instead define a parametric update process which does not depend on label information,

$$\phi_{t+1} = UnsupervisedUpdate(\phi_t, x_t; \theta). \tag{2}$$

We then train the $UnsupervisedUpdate$ function by performing SGD on a meta-loss, in terms of the meta-parameters θ,

$$\theta^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_t MetaObjective(\phi_t(\theta)). \tag{3}$$

In the following sections, we briefly review the main functional pieces to this model, the base model ($f(\cdot, \phi)$), the $UnsupervisedUpdate$, and the $MetaObjective$.

2.1 Base Model: $f(\cdot; \phi)$

We restrict our attention to training a family of feed forward MLPs. The weight matrix and bias vector for layer l are W^l and b^l respectively. The network has pre-activations $z^1..z^L$, and post-activations $x^0..x^L$, where L is the number of layers, and $x^0 \equiv x$ is the network input. Batch norm (Ioffe & Szegedy, 2015) is applied at each layer.

To encourage generalization of the update rule across model architectures, during meta-training we sample the layer width and number of layers from a distribution (except where otherwise noted for specific experiments).

2.2 Learned update rule: $UnsupervisedUpdate(\cdot; \theta)$

In order to build an unsupervised update rule that generalizes across architectures, we design our unsupervised update rule to be neuron-local. That is, every neuron i in every layer l in the base model has an update network $h_i^l(\cdot; \theta)$, itself an MLP, associated with it. All update networks share parameters θ, and $h_i^l(\cdot; \theta)$ is evaluated during unsupervised training only. Evaluating the statistics of unit activation over a batch of data has proven helpful in supervised learning, in the case of batch norm. It has similarly proven helpful in hand designed unsupervised learning rules, for instance in sparse coding. We therefore allow $h_i^l(\cdot; \theta)$ to accumulate statistics across training minibatches.

During an unsupervised training step, the base-model is first run in a standard feedforward fashion, populating x_{ib}^l, z_{ib}^l, where b is the training minibatch index. As in supervised learning, an error signal δ_{ib}^l is then propagated backwards through the network. Unlike in supervised backprop however, this error signal is generated by the corresponding update network for each unit, $\delta_{ib}^l \leftarrow h_i^l(\cdot; \theta)$. Again as in supervised learning, the weight updates are a product of pre-and-post-synaptic signals. Unlike in supervised learning however, those signals are also generated from the update networks, $\Delta W_{ij}^l = \sum_b c_{ib}^l d_{jb}^{l-1}$, $\{c_{ib}^l, d_{ib}^l\} \leftarrow h_i^l(\cdot; \theta)$. The input to the update network consists of unit pre- and post-activations, and backwards propagated error signal, i.e. it can be written $h_i^l\left(x_{i.}^l, z_{i.}^l, \left[\left(W^{l+1}\right)^T \delta^{l+1}\right]_{i.}; \theta\right)$. Additional lateral interaction terms, not described in this short submission, enable units within a layer to remain decorrelated with each other. The internal architecture of the update networks $h_i^l(\cdot; \theta)$ is similarly beyond the scope of this submission, but involves repeated convolution along the batch dimension.

2.3 Meta-loss: $MetaObjective(\phi)$

The meta-loss we use in this work is based on few shot classification on the outputs of the base model, X^L. This classification task is performed via linear ridge regression on one-hot feature label vectors. The benefits of ridge regression over e.g. cross entropy loss are that it is differentiable, with a closed form expression for the coefficients. In order to encourage the learning of features which generalize well, we estimate the regression coefficients on one minibatch $\{x_a, y_a\}$ of data, and evaluate the classification performance on a second minibatch $\{x_b, y_b\}$,

$$\hat{v} = \underset{v}{\operatorname{argmin}} \left(\left\| y_a - x_a^L v \right\|^2 + \lambda \left\| v \right\|^2 \right); \qquad MetaObjective = \left\| y_b - x_b^L \hat{v} \right\|^2, \tag{4}$$

where x^L is a function of the base-model parameters ϕ, and through ϕ is a function of the learned update rule's parameters θ. The meta-loss therefore encourages a learning rule that leads to good performance on a test set after semi-supervised training.

3 Experiments

3.1 Training setup

To explore transfer of our learned update rule, we use a variety of datasets. We train on 14x14 black and white images containing characters of the English alphabet and evaluate our technique on datasets such as 14x14 MNIST. We sample 10 way classification problems from the 26 letters of the alphabet. We train θ, by estimating $\frac{\partial[MetaObjective]}{\partial\theta}$ via truncated backprop. During training gradients are accumulated from 512 CPU workers, and θ is updated with mini batch asynchronous SGD. By batching workers as they complete we eliminate most gradient staleness while retaining the compute efficiency of asynchronous SGD.

3.2 Application of the Learned Unsupervised Update Rule

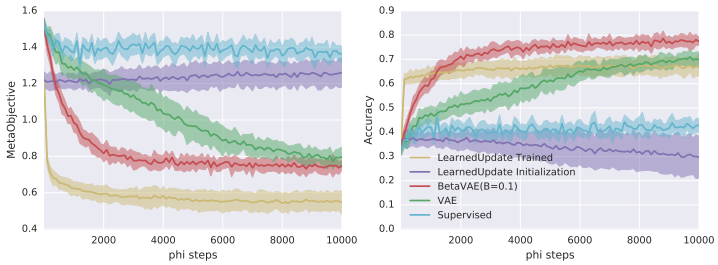

Figure 1: Learning curves of our method for a test task of 14x14 MNIST (both before and after training), as well as comparisons with existing unsupervised learning techniques (VAE and $\beta-$VAE(Higgins et al., 2016)). In addition, we show supervised learning baseline. In all settings, only 10 labeled examples per class are used. Despite being meta-trained on a different task, our learned update rule is capable of minimizing $MetaObjective$ and creating useful features. We do not, however, perform as well as VAE base methods on accuracy.

We evaluate our learned update rule's performance by iteratively applying it on a held out task. Results for 10 shot classification performance on a dataset of 14x14 MNIST digits can be found in Figure 1. Our learned update rule is able to reach higher performance when measured on the $MetaObjective$ than several existing unsupervised methods, but does not perform as well as existing unsupervised techniques when evaluating classification accuracy. It does train faster than existing techniques in early learning.

Next we evaluate performance on a much larger domain gap. We construct a dataset consisting of 3 28x28 MNIST digits stacked in the channel dimension. Results can be found in Figure 2. Early in meta-training, the learned update rule is able to learn some independent features for each of the color channels. This is surprising as there is no hand coded tendency to learn independent features (or any features). However, late in meta-training, the learned updater collapses to a single color channel. We hypothesize this an effect of "overfitting" to 10-way classification problems.

Finally, we test base model generalization, by varying depth and number of hidden units. In both settings, we are able to generalize and learn useful representations early in ϕ iterations. We find that increasing the number of hidden units beyond the range used in meta-training leads to *better* performance. See Figure 3.

3.3 Analyzing Learning Strategy

Our learned algorithm is capable of learning interesting features. We visualize first layer filters of f over the course of meta-training (updates of θ). Base-models appear to learn to create templates of input examples as filters, and then refine those input filters into more compositional features.

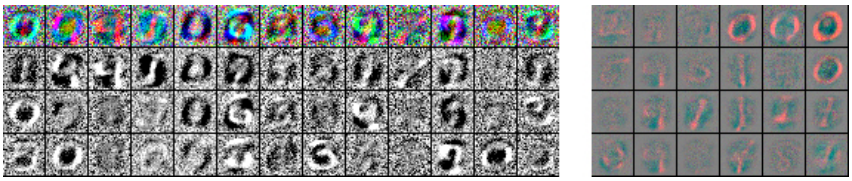

Figure 2: Filters from ϕ training obtained by applying our learned update rule to a dataset consisting of 3 concatenated MNIST digits. Left: Early in meta-training the filters show variation and some compositionallity. Top row corresponds to color filters, bottom three rows break out each color channel. Right: Later in meta-training the filters "collapse" to model only one color (in this case red). We believe this is a form of overfitting to the 10 way classification task.

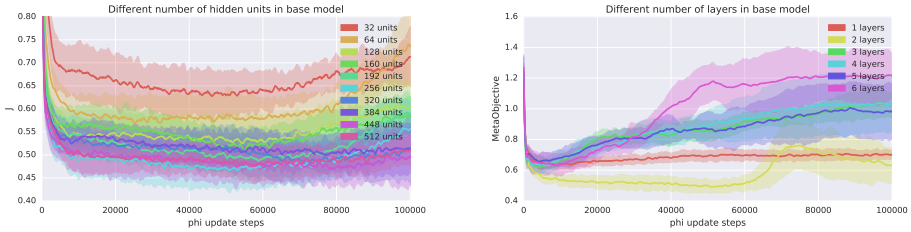

Figure 3: Generalization of the unsupervised update rule to new base-model architectures. Left: different numbers of hidden units with a fixed layer size. Right: diferent numbers of layers with a fixed hidden size. The unsupervised update rule used here was meta-trained only on base models with 2 hidden layers, and < 200 units per layer.

Figure 4: From left to right we show filters extracted from ϕ after every 10k steps of θ training. Each image represents a different point in θ training. At meta-initialization, the process is unstable and collapses. After some meta-training, the learned update rule first learns templates of increasing diversity, before finally transitioning to slightly more compositional, and less interpretable, features.

4 Limitations

Optimization posed an enormous challenge over the course of this work. We used truncated back-prop during training. Careful balancing was required between short truncation windows, which lead to a non-stationary meta-training distribution, and biased and non-stationary gradients, and long truncation windows, which often lead to gradients with diverging variance. This is similar to the issues encountered in off-policy reinforcement learning. Changes to the learned policy change the state visitation distribution. Unlike RL, it is not tractable for us to compute these probabilities, and thus we cannot easily use importance sampling to yield unbiased gradients.

Currently, we encounter a performance gap when transitioning to more complex or noisier data. Upon inspection, we appear to have discovered a local minimum in algorithm space which consists of learning input templates. The computation and capacity available in higher layers is largely unused. Diagnosing this phenomenon is ongoing work.

5 Conclusion

In this work we introduce a method of meta-learning an unsupervised representation learning algorithm. We show that this algorithm behaves reasonably in both performance, on a restricted setting, and generalization across task and model architecture.

Acknowledgments

We would like to thank Hugo Larochelle, Nando de Freitas, Niru Maheswaranathan, Olga Wichrowska, Samy Bengio, Pavel Sountsov, and Alex Toshev for extremely helpful conversations and feedback on this work.

References

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pp. 3981–3989, 2016.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

Sepp Hochreiter, A Steven Younger, and Peter R Conwell. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pp. 87–94. Springer, 2001.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL `http://proceedings.mlr.press/v37/ioffe15.html`.

Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, pp. 2113–2122, 2015.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

Juergen Schmidhuber. On learning how to learn learning strategies. 1995.

Olga Wichrowska, Niru Maheswaranathan, Matthew W Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Nando de Freitas, and Jascha Sohl-Dickstein. Learned optimizers that scale and generalize. *arXiv preprint arXiv:1703.04813*, 2017.