Learning how to learn

Machine Learning and Data Mining

Maxim Borisyak

National Research University Higher School of Economics (HSE)

December 10, 2018

Meta-learning

Traditional learning:

- given problem $D = (X_i, Y_i)_{i=1}^N$, $X_i, Y_i \sim P(x, y)$
- find decision function f
- that minimizes error $R(f, P)$.

Meta-learning:

- given family of problems $D_j = (X_i^j, Y_i^j)_{i=1}^{N_j}$,
- $D_j \sim P_\theta^{N_j}(x, y)$, $\theta \sim P(\theta)$
- find mapping $g : D \to f$
- such that minimizes error $\mathbb{E}_\theta \, \mathbb{E}_{D \sim P_\theta^N} \, R(g(D), P_\theta)$

This definition is not commonly accepted.

Meta-learning

$$\text{data} = \underbrace{\underbrace{\text{shared knowledge}}_{\text{pre learn}} + \underbrace{\text{concept}}_{\text{learn to infer}}}_{\text{learn to combine}}$$

Meta-learning flavors

- concept learning, few-shot learning;
- learning optimization procedure;
- learning model:
  - hyper-parameter optimization.

While in practice these are quite distinct areas, theoretically, the lines are blurry.

Few-shot learning

Weight sharing

- model $f(\psi, \theta_i)$ with two sets of parameters:
  - ψ — independent of task;
  - θ_i — parameters for i-th task.

Train:

$$\psi^* = \arg\min \min_{\theta_1, \theta_2, \ldots} \sum_i \mathcal{L}(f(\psi, \theta_i), D_i)$$

Test:

$$\theta^* = \arg\min_\theta \mathcal{L}(f(\psi^*, \theta), D)$$

Also known as 'replace-head-of-an-already-trained-network' or transfer learning.

'Soft' weight sharing

Train:

$$\psi^* = \arg\min_{\psi} \left[\min_{\theta_1,\theta_2,\ldots} \sum_i \mathcal{L}(f(\theta_i), D_i) + \sum_i \|\psi - \theta_i\|^2 \right]$$

Test:

$$\theta^* = \arg\min_{\theta} \left[\mathcal{L}(f(\theta), D) + \|\psi^* - \theta\|^2 \right]$$

Also known as fine-tuning of a pretrained network or transfer learning.

Model Agnostic Meta-Learning

- 'soft' weight sharing constrains weights to be close to each other in l_2 space;
- instead let's directly incorporate test weights inference into training:

$$\psi^* = \arg\min_{\psi} \left[\sum_i \mathcal{L}(f(\psi_i), D_i) \right]$$

where:

$$\psi_i = \psi - \alpha \left. \frac{\partial \mathcal{L}(f(\theta), D_i)}{\partial \theta} \right|_{\theta=\psi}$$

Model Agnostic Meta-Learning

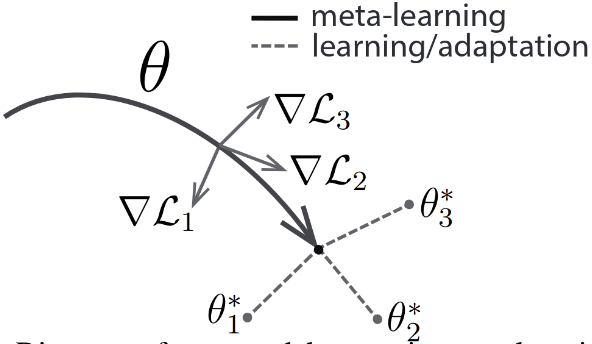
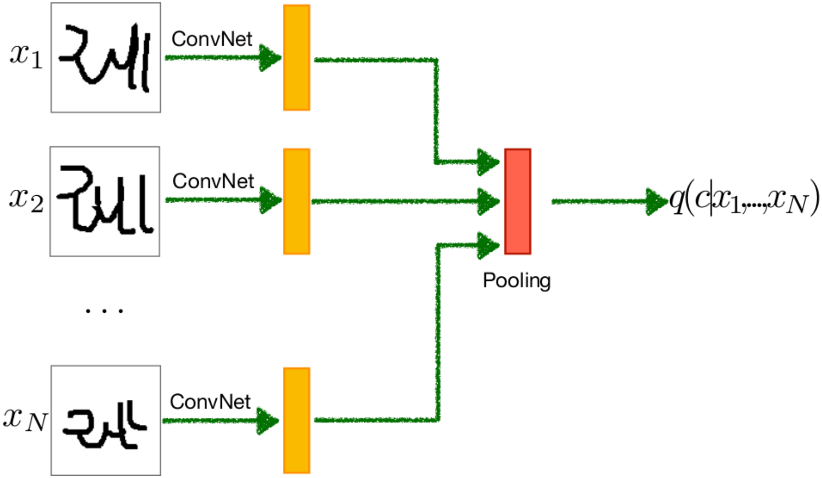


Figure 1. Diagram of our model-agnostic meta-learning algorithm (MAML), which optimizes for a representation θ that can quickly adapt to new tasks.

Generative models

Neural Statistician



Neural Statistician

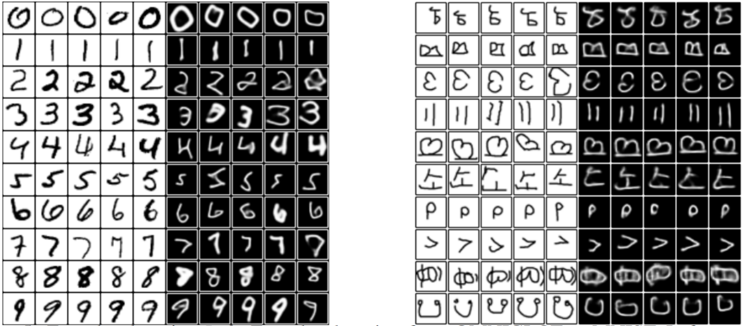
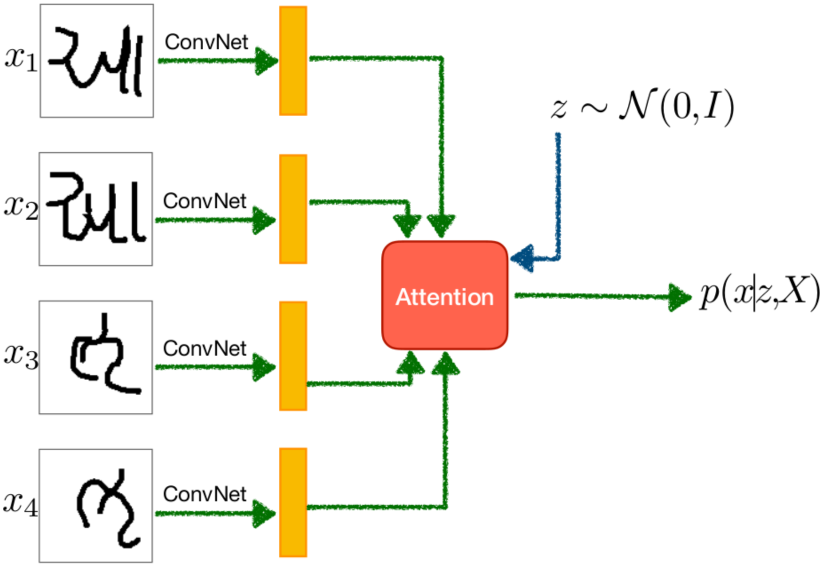


Figure 5: Few-shot learning *Left*: Few-shot learning from OMNIGLOT to MNIST. Left rows are input sets, right rows are samples given the inputs. *Right*: Few-shot learning from with OMNIGLOT data to unseen classes. Left rows are input sets, right rows are samples given the inputs. Black-white inversion is applied for ease of viewing.

Neural Statistician

Figure 6: Few-shot learning for face data. Samples are from model trained on Youtube Faces Database. *Left*: Each row shows an input set of size 5. *Center*: Each row shows 5 samples from the model corresponding to the input set on the left. *Right*: Imagined new faces generated by sampling contexts from the prior. Each row consists of 5 samples from the model given a particular sampled context.

Generative Matching Networks



Generative Matching Networks

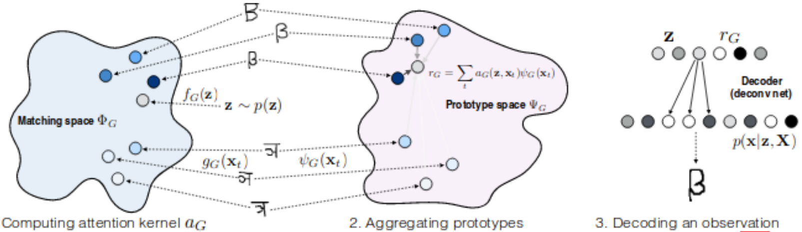
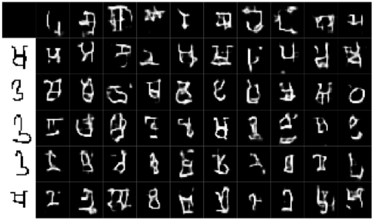


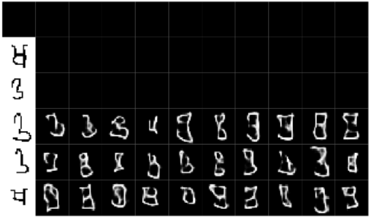
Figure 1: Generation of a new sample in a basic generative matching network, see section $\boxed{3.1}$ for the description of functions f, g and ψ.

Generative Matching Networks

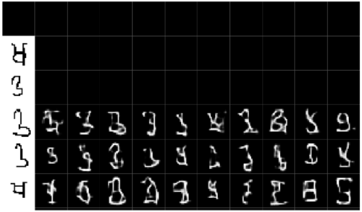
(a) GMN (no pseudo-input)



(b) GMN (one pseudo-input)



(c) GMN (no attention, no pseudo-input)



(d) Neural statistician

Learning optimizers

Replacing gradient descent

$$\theta^{t+1} = \theta^{t} + g_{\psi}(\nabla f(\theta_t))$$

$$\mathcal{L}(\psi) = \mathop{\mathbb{E}}_{f}\left[w_t f(\theta_t)\right]$$

Replacing gradient descent

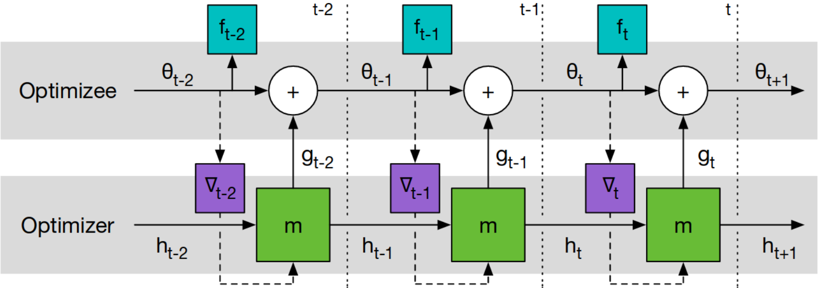


Figure 2: Computational graph used for computing the gradient of the optimizer.

Replacing gradient descent

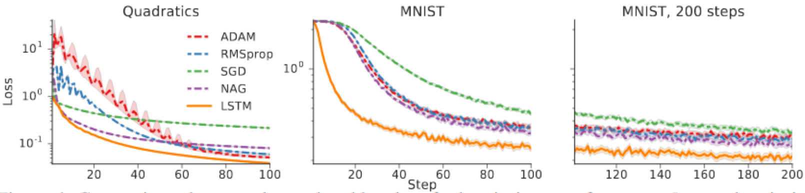


Figure 4: Comparisons between learned and hand-crafted optimizers performance. Learned optimizers are shown with solid lines and hand-crafted optimizers are shown with dashed lines. Units for the y axis in the MNIST plots are logits. **Left:** Performance of different optimizers on randomly sampled 10-dimensional quadratic functions. **Center:** the LSTM optimizer outperforms standard methods training the base network on MNIST. **Right:** Learning curves for steps 100-200 by an optimizer trained to optimize for 100 steps (continuation of center plot).

Replacing gradient descent

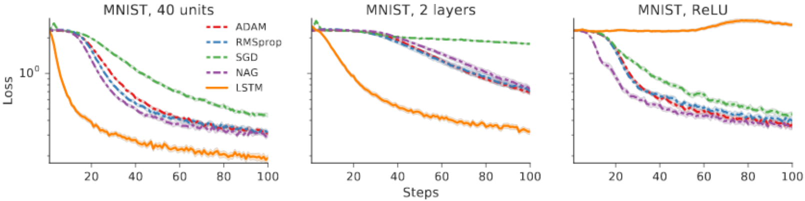


Figure 5: Comparisons between learned and hand-crafted optimizers performance. Units for the y axis are logits. **Left:** Generalization to the different number of hidden units (40 instead of 20). **Center:** Generalization to the different number of hidden layers (2 instead of 1). This optimization problem is very hard, because the hidden layers are very narrow. **Right:** Training curves for an MLP with 20 hidden units using ReLU activations. The LSTM optimizer was trained on an MLP with sigmoid activations.

References

References

- Edwards H, Storkey A. Towards a neural statistician. arXiv preprint arXiv:1606.02185. 2016 Jun 7.
- Bartunov S, Vetrov DP. Fast adaptation in generative models with generative matching networks. arXiv preprint arXiv:1612.02192. 2016 Dec 7.
- Finn C, Abbeel P, Levine S. Model-agnostic meta-learning for fast adaptation of deep networks. arXiv preprint arXiv:1703.03400. 2017 Mar 9.
- Munkhdalai T, Yu H. Meta networks. arXiv preprint arXiv:1703.00837. 2017 Mar 2.

References

- Jia X, De Brabandere B, Tuytelaars T, Gool LV. Dynamic filter networks. InAdvances in Neural Information Processing Systems 2016 (pp. 667-675).
- Andrychowicz M, Denil M, Gomez S, Hoffman MW, Pfau D, Schaul T, Shillingford B, De Freitas N. Learning to learn by gradient descent by gradient descent. InAdvances in Neural Information Processing Systems 2016 (pp. 3981-3989).