Learning to Learn

Siddharth Swaroop and Will Tebbutt

University of Cambridge

23-11-2017

Learning to Learn

Introduction

- No universally agreed upon definition.
- Possible definitions:
  - Replace hand-crafted components of learning algorithms with learned ones.
  - Have an outer learning algorithm that seeks to optimise an inner algorithm.
- e.g.
  - Learning to optimise.
  - Few-shot learning.
- Synonymous with Meta-Learning.

Learning to Optimise

Learning as Optimisation

- (Old & well-known) idea: View learning as an optimisation problem.
- $$\theta^* = \operatorname*{argmin}_{\theta} f(\theta)$$

e.g.

- (Old & well-known) idea: View learning as an optimisation problem.
-
$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, f(\theta)$$

e.g.

- MLE: $$f(\theta) := -\log p(\mathcal{D} \mid \theta)$$

Learning to Optimise

Learning as Optimisation

- (Old & well-known) idea: View learning as an optimisation problem.
- $$\theta^* = \underset{\theta}{\operatorname{argmin}}\, f(\theta)$$

e.g.

- MLE: $f(\theta) := -\log p(\mathcal{D} \mid \theta)$
- VB: $f(\theta) := \mathcal{KL}\left[q_\theta(\cdot) \;||\; p(\cdot \mid \mathcal{D})\right]$

Learning to Optimise

Iterative Optimisation

- (Old & well-known) idea: Pick a function g where

$$\theta_{t+1} = g\left(f, \theta_{1:t}\right)$$

- Construct g s.t. $\theta_t \to \theta^*$ as t becomes large. (Hopefully)

e.g.

Learning to Optimise

Iterative Optimisation

- (Old & well-known) idea: Pick a function g where

$$\theta_{t+1} = g\left(f, \theta_{1:t}\right)$$

- Construct g s.t. $\theta_t \to \theta^*$ as t becomes large. (Hopefully)

e.g.

- (S)GD-like: $g\left(f, \theta_{1:t}\right) := \theta_t - \eta\left(f, \theta_{1:t}\right) \nabla f\left(\theta_t\right)$

Learning to Optimise

Iterative Optimisation

- (Old & well-known) idea: Pick a function g where

$$\theta_{t+1} = g\left(f, \theta_{1:t}\right)$$

- Construct g s.t. $\theta_t \to \theta^*$ as t becomes large. (Hopefully)

e.g.

- (S)GD-like: $g\left(f, \theta_{1:t}\right) := \theta_t - \eta\left(f, \theta_{1:t}\right) \nabla f\left(\theta_t\right)$
- Bayes Opt.: $g\left(f, \theta_{1:t}\right) = \text{argmin}_\theta\, \mathcal{A}\left(\theta, p\left(\hat{f} \,\middle|\, \theta_{1:t}, y_{1:t}\right)\right)$

Learning to Optimise

Iterative Optimisation

- (Old & well-known) idea: Pick a function g where

$$\theta_{t+1} = g\left(f, \theta_{1:t}\right)$$

- Construct g s.t. $\theta_t \to \theta^*$ as t becomes large. (Hopefully)

e.g.

- (S)GD-like: $g\left(f, \theta_{1:t}\right) := \theta_t - \eta\left(f, \theta_{1:t}\right) \nabla f\left(\theta_t\right)$
- Bayes Opt.: $g\left(f, \theta_{1:t}\right) = \operatorname{argmin}_\theta \mathcal{A}\left(\theta, p\left(\hat{f} \,\middle|\, \theta_{1:t}, y_{1:t}\right)\right)$
- Observation: g is hand-crafted (up to a couple of free parameters).

Learning to Optimise

Learning to Iteratively Optimise

- (New-ish) idea: Learn g.
- e.g. Parameterise g

$$\theta_{t+1} = g_{\varphi}(f, \theta_{1:t})$$

and learn φ from data. (data = optimisation problems)

Learning to Optimise

"Learning step size controllers for robust neural network training" [Daniel et al., 2016]

Learn a controller for the step-size of SGD [Daniel et al., 2016]:

$$g_{\varphi}\left(f, \theta_{1:t}\right) = \theta_t - \underbrace{\exp\left(\varphi^T \hat{\phi}_t\right)}_{\text{replaces } \eta} \nabla f\left(\theta_t\right), \quad \hat{\phi}_t = \phi\left(\theta_t, \hat{\phi}_{t-1}\right),$$

where ϕ is a hand-crafted vector-valued basis function.

Advantages

- Low memory footprint
- Small number of parameters

Disadvantages

- Hand-engineered features
- No second-order info. used

Learning to Optimise

"Learning to Learn by Gradient Descent by Gradient Descent" [Andrychowicz et al., 2016]

$$g_\varphi\left(f, \theta_{1:t}\right) = \theta_t - \underbrace{r_\varphi\left(h_t\right)}_{\substack{\text{replaces} \\ \eta \nabla f(\theta_t)}}, \quad h_t = H_\varphi\left(h_{t-1}, \nabla f\left(\theta_{t-1}\right)\right).$$

Advantages

- Flexible
- Variable dimensionality

Disadvantages

- Large memory footprint
- No coupling

Learning to Optimise

"Learning to Optimize" [Li and Malik, 2016]

Learn an MLP autoregressor m_φ:

$$g_\varphi\left(f, \theta_{1:t}\right) = m_\varphi\left(\theta_t, f\left(\theta_{t-H:t}\right), \nabla f\left(\theta_{t-H:t}\right)\right),$$

for some $H \in \mathbb{N}$.

Advantages

- Flexible
- Coupling between variables

Disadvantages

- Large memory footprint
- Not flexible in no. dims.

Learning to Optimise

"Learning to Learn without Gradient Descent by Gradient Descent" [Chen et al., 2017]

Let g_φ be an RNN r_φ with hidden state h_t given by

$$g_\varphi\left(f, \theta_{1:t}\right) = r_\varphi\left(h_t\right), \quad h_t = H_\varphi\left(h_{t-1}, \theta_{t-1}, f\left(\theta_{t-1}\right)\right).$$

Advantages

- Flexible parametrisation
- Coupling between dims

Disadvantages

- Large memory footprint
- Requires ∇f during training
- Fixed dimensionality.

Learning to Optimise

"Learning to Learn without Gradient Descent by Gradient Descent" [Chen et al., 2017]

- EI training criterion:

$$L_{\mathsf{EI}}(\theta) = -\mathbb{E}_{f, y_{1:T-1}} \underbrace{\left[\sum_{t=1}^{T} \mathsf{EI}\left(\theta_t \mid y_{1:t-1}\right) \right]}_{\text{Computed via GP}}$$

Learning to Optimise

“Learning to Learn without Gradient Descent by Gradient Descent” [Chen et al., 2017]

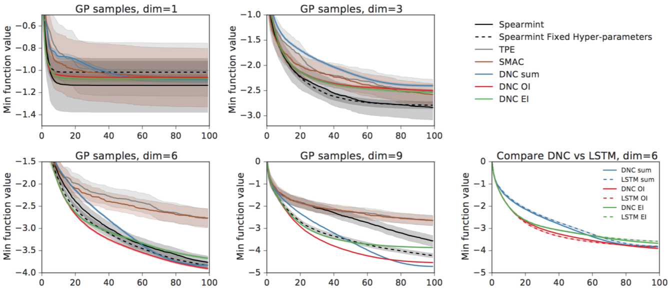


Figure 3. Average minimum observed function value, with 95% confidence intervals, as a function of search steps on functions sampled from the training GP distribution. Left four figures: Comparing DNC with different reward functions against Spearmint with fixed and estimated GP hyper-parameters, TPE and SMAC. Right bottom: Comparing different DNCs and LSTMs. As the dimension of the search space increases, the DNC's performance improves relative to the baselines.

Learning to Optimise

Conclusion

- Learning to optimise is conceptually appealing.
- Empirical results seem promising.
- Memory requirements large relative to simple methods.
  - Probably a careful trade-off required between flexibility and overhead.

Learning to Learn

Data-efficient learning

- Problem: ‘Insufficient’ training data
- Use related data
- k-shot learning
  - Image classification
  - Sentence completion

Learning to Learn

k-shot learning

$$\text{Large}\,\tilde{\mathcal{D}} = \{\underline{\tilde{u}}_i, \tilde{y}_i\}_{i=1}^{\tilde{N}},\ \tilde{y}_i \in \{1, ..., \tilde{C}\}$$
$$\text{Small}\,\mathcal{D} = \{\underline{u}_i, y_i\}_{i=1}^{kC},\ y_i \in \{\tilde{C}+1, ..., \tilde{C}+C\}$$

Learning to Learn
k-shot learning

$$\text{Large}\,\tilde{\mathcal{D}} = \{\underline{\tilde{u}}_i, \tilde{y}_i\}_{i=1}^{\tilde{N}},\ \tilde{y}_i \in \{1, ..., \tilde{C}\}$$
$$\text{Small}\,\mathcal{D} = \{\underline{u}_i, y_i\}_{i=1}^{kC},\ y_i \in \{\tilde{C}+1, ..., \tilde{C}+C\}$$

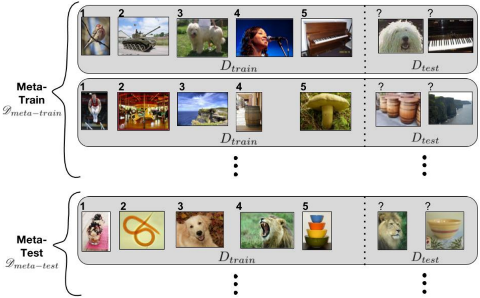
- LSTM meta-learner
- Learning on learnt features

Learning to Learn

"Optimisation as a model for few-shot learning" [Ravi and Larochelle, 2017]

$$\theta_t = \quad \theta_{t-1} - \alpha_t \nabla_{\theta_{t-1}} \mathcal{L}_t$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

- Learn f_t, i_t, c_0



Learning to Learn

Learning on features

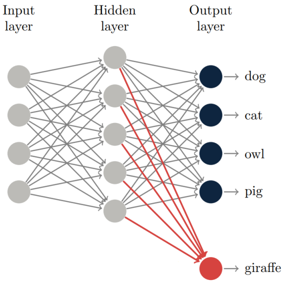
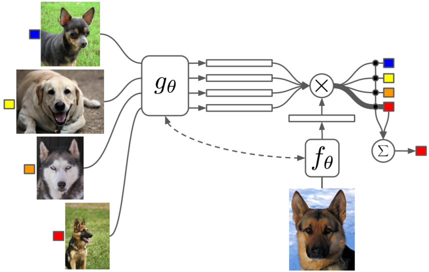


Figure 1: Learning on features [Burgess et al., 2016]

- Train a Neural Network classifier on $\tilde{\mathcal{D}}$
  - Train on embedded features obtained from last hidden layer
- Common baseline: Nearest neighbour matching

Learning to Learn

"Matching Networks for One Shot Learning" [Vinyals et al., 2016]



$$a(\hat{x}, x_i) = \frac{e^{c(f(\hat{x},S),g(x_i,S))}}{\sum_{j=1}^{k} e^{c\left(f(\hat{x},S),g(x_j,S)\right)}}$$

$$\hat{y} = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i$$

Learning to Learn

"Discriminative k-shot learning using probabilistic models" [Bauer et al., 2017]

- Bayesian approach on softmax weights
- Found single Gaussian worked best

$$p(\mathrm{W}|\mathcal{D}, \tilde{\mathcal{D}}) \propto \mathcal{N}\left(\mathrm{W}|\mu^{\mathrm{MAP}}, \Sigma^{\mathrm{MAP}}\right) \prod_{n=1}^{N} p(y_n|\mathbf{x}_n, \mathrm{W})$$

Learning to Learn

"Discriminative k-shot learning using probabilistic models" [Bauer et al., 2017]

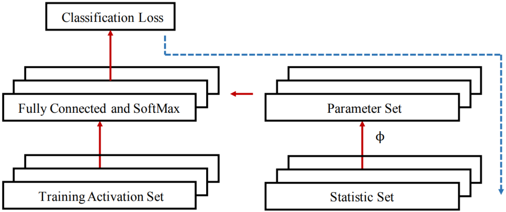
- Bayesian approach on softmax weights
- Found single Gaussian worked best

$$p(\mathrm{W}|\mathcal{D}, \tilde{\mathcal{D}}) \propto \mathcal{N}\left(\mathrm{W}|\mu^{\mathrm{MAP}}, \Sigma^{\mathrm{MAP}}\right) \prod_{n=1}^{N} p(y_n|\mathbf{x}_n, \mathrm{W})$$

Method	1-shot	5-shot
ResNet-34 + Isotropic Gaussian (ours)	$\mathbf{56.3 \pm 0.4\%}$	$\mathbf{73.9 \pm 0.3\%}$
Matching Networks (reimplemented, 1-shot)	$46.8 \pm 0.5\%$	-
Matching Networks (reimplemented, 5-shot)	-	$62.7 \pm 0.5\%$
Meta-Learner LSTM (Ravi & Larochelle, 2017)	$43.4 \pm 0.8\%$	$60.6 \pm 0.7\%$
Prototypical Nets (1-shot) (Snell et al., 2017)	$49.4 \pm 0.8\%$	$65.4 \pm 0.7\%$
Prototypical Nets (5-shot) (Snell et al., 2017)	$45.1 \pm 0.8\%$	$68.2 \pm 0.7\%$

Learning to Learn

"Few-Shot Image Recognition by Predicting Parameters from Activations" [Qiao et al., 2017]



$$P(y_i|x_i) = \frac{e^{\mathbf{a}(x_i)\cdot\phi\left(\mathbb{E}_S[\mathbf{s}_{y_i}]\right)}}{Z}$$

- Statistic set (1st moment)
- At k-shot train, each sample is new category

Learning to Learn

Potential future research

- Similarities between classes (eg animals)
  - Attempted with GMMs [Bauer et al., 2017]
- Combine feature-learning with learning on features for general k-shot
  - Currently ‘fine-tuning’ is heuristic

Bibliography I

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., and de Freitas, N. (2016).
Learning to learn by gradient descent by gradient descent.
In *Advances in Neural Information Processing Systems*, pages 3981–3989.

Bauer, M., Rojas-Carulla, M., Swiatkowski, J. B., Schölkopf, B., and Turner, R. E. (2017).
Discriminative k-shot learning using probabilistic models.
arXiv:1706.00326 [cs, stat].
arXiv: 1706.00326.

Burgess, J., Lloyd, J. R., and Ghahramani, Z. (2016).
One-Shot Learning in Discriminative Neural Networks.
NIPS Bayesian Deep Learning Workshop.

Chen, Y., Hoffman, M. W., Colmenarejo, S. G., Denil, M., Lillicrap, T. P., Botvinick, M., and Freitas, N. (2017).
Learning to learn without gradient descent by gradient descent.
In *International Conference on Machine Learning*, pages 748–756.

Daniel, C., Taylor, J., and Nowozin, S. (2016).
Learning step size controllers for robust neural network training.
In *AAAI*, pages 1519–1525.

Li, K. and Malik, J. (2016).
Learning to optimize.
arXiv preprint arXiv:1606.01885.

Qiao, S., Liu, C., Shen, W., and Yuille, A. (2017).
Few-Shot Image Recognition by Predicting Parameters from Activations.
arXiv:1706.03466 [cs].
arXiv: 1706.03466.

Bibliography II

Ravi, S. and Larochelle, H. (2017).
Optimization as a model for few-shot learning.
International Conference on Learning Representations, 1(2):6.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. (2016).
Matching Networks for One Shot Learning.
Advances in Neural Information Processing Systems.
arXiv: 1606.04080.