A PAC-Bayes perspective on binary-activated deep neural networks

Benjamin Guedj
https://bguedj.github.io

MLRW #5, Criteo
October 2, 2019

Context

Context

- Learning is to be able to **generalise**!

Context

- Learning is to be able to **generalise**!
- **PAC-Bayes** has been successfully used to analyse and understand generalisation abilities of machine learning algorithms.

Context

- Learning is to be able to **generalise**!

- **PAC-Bayes** has been successfully used to analyse and understand generalisation abilities of machine learning algorithms.
  - G., ”A Primer on PAC-Bayesian Learning”, invited for publication in the Proceedings of the French Mathematical Society, `https://arxiv.org/abs/1901.05353`
  - G. & Shawe-Taylor, ”A Primer on PAC-Bayesian Learning”, **ICML 2019 tutorial** `https://bguedj.github.io/icml2019/index.html`

Context

- Learning is to be able to **generalise**!
- **PAC-Bayes** has been successfully used to analyse and understand generalisation abilities of machine learning algorithms.
  - G., "A Primer on PAC-Bayesian Learning", invited for publication in the French Mathematical Society, `https://arxiv.org/abs/1901.05353`
  - G. & Shawe-Taylor, "A Primer on PAC-Bayesian Learning", **ICML 2019 tutorial** `https://bguedj.github.io/icml2019/index.html`
- Most PAC-Bayes generalisation bounds are **computable** tight upper bounds on the population error, *i.e.* an estimate of the error on any unseen future data.

Context

- Learning is to be able to **generalise**!
- **PAC-Bayes** has been successfully used to analyse and understand generalisation abilities of machine learning algorithms.
  - G., "A Primer on PAC-Bayesian Learning", invited for publication in the Proceedings of the French Mathematical Society, `https://arxiv.org/abs/1901.05353`
  - G. & Shawe-Taylor, "A Primer on PAC-Bayesian Learning", **ICML 2019 tutorial** `https://bguedj.github.io/icml2019/index.html`
- Most PAC-Bayes generalisation bounds are **computable** tight upper bounds on the population error, *i.e.* an estimate of the error on any unseen future data.
- PAC-Bayes bounds hold for **any distribution on hypotheses.** As such, they are a principled way to invent new learning algorithms.

This spotlight

G. Letarte, P. Germain, B. G., F. Laviolette. *Dichotomize and Generalize: PAC-Bayesian Binary Activated Deep Neural Networks*, to appear in NeurIPS 2019

`https://arxiv.org/abs/1905.10259`

This spotlight

G. Letarte, P. Germain, B. G., F. Laviolette. *Dichotomize and Generalize: PAC-Bayesian Binary Activated Deep Neural Networks*, to appear in NeurIPS 2019
`https://arxiv.org/abs/1905.10259`

We focused on DNN with a **binary activation function**: surprisingly effective while preserving low computing and memory footprints.

This spotlight

G. Letarte, P. Germain, B. G., F. Laviolette. *Dichotomize and Generalize: PAC-Bayesian Binary Activated Deep Neural Networks*, to appear in NeurIPS 2019
https://arxiv.org/abs/1905.10259

We focused on DNN with a **binary activation function**: surprisingly effective while preserving low computing and memory footprints.

- Very few meaningful generalisation bounds for DNN

This spotlight

G. Letarte, P. Germain, B. G., F. Laviolette. *Dichotomize and Generalize: PAC-Bayesian Binary Activated Deep Neural Networks*, to appear in NeurIPS 2019
https://arxiv.org/abs/1905.10259

We focused on DNN with a **binary activation function**: surprisingly effective while preserving low computing and memory footprints.

- Very few meaningful generalisation bounds for DNN
  Breakthrough: SOTA PAC-Bayes generalisation bound

This spotlight

G. Letarte, P. Germain, B. G., F. Laviolette. *Dichotomize and Generalize: PAC-Bayesian Binary Activated Deep Neural Networks*, to appear in NeurIPS 2019
https://arxiv.org/abs/1905.10259

We focused on DNN with a **binary activation function**: surprisingly effective while preserving low computing and memory footprints.

- Very few meaningful generalisation bounds for DNN
  Breakthrough: SOTA PAC-Bayes generalisation bound
- How to train a network with non-differentiable activation function?

This spotlight

G. Letarte, P. Germain, B. G., F. Laviolette. *Dichotomize and Generalize: PAC-Bayesian Binary Activated Deep Neural Networks*, to appear in NeurIPS 2019
`https://arxiv.org/abs/1905.10259`

We focused on DNN with a **binary activation function**: surprisingly effective while preserving low computing and memory footprints.

- Very few meaningful generalisation bounds for DNN
  Breakthrough: SOTA PAC-Bayes generalisation bound
- How to train a network with non-differentiable activation function?
  Breakthrough: training by minimising the bound (SGD + tricks)

This spotlight

G. Letarte, P. Germain, B. G., F. Laviolette. *Dichotomize and Generalize: PAC-Bayesian Binary Activated Deep Neural Networks*, to appear in NeurIPS 2019
`https://arxiv.org/abs/1905.10259`

We focused on DNN with a **binary activation function**: surprisingly effective while preserving low computing and memory footprints.

- Very few meaningful generalisation bounds for DNN
  Breakthrough: SOTA PAC-Bayes generalisation bound
- How to train a network with non-differentiable activation function?
  Breakthrough: training by minimising the bound (SGD + tricks)
- Who cares? Generalisation bounds are a theoretician's concern!

This spotlight

G. Letarte, P. Germain, B. G., F. Laviolette. *Dichotomize and Generalize: PAC-Bayesian Binary Activated Deep Neural Networks*, to appear in NeurIPS 2019
`https://arxiv.org/abs/1905.10259`

We focused on DNN with a **binary activation function**: surprisingly effective while preserving low computing and memory footprints.

- Very few meaningful generalisation bounds for DNN
  Breakthrough: SOTA PAC-Bayes generalisation bound
- How to train a network with non-differentiable activation function?
  Breakthrough: training by minimising the bound (SGD + tricks)
- Who cares? Generalisation bounds are a theoretician's concern!
  Breakthrough: Our bound is computable and serves as a safety check to practitioners

Binary Activated Neural Networks

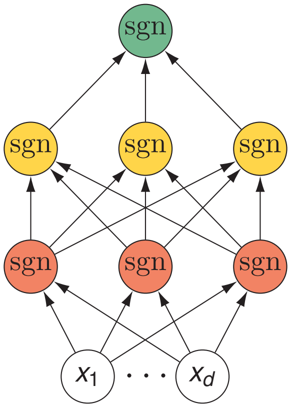
- $\mathbf{x} \in \mathbb{R}^{d_0}$, $y \in \{-1, 1\}$

Architecture:

- *L fully connected* layers
- d_k denotes the number of neurons of the k^{th} layer
- $\mathrm{sgn}(a) = 1$ if $a > 0$ and $\mathrm{sgn}(a) = -1$ otherwise

Parameters:

- $\mathbf{W}_k \in \mathbb{R}^{d_k \times d_{k-1}}$ denotes the weight matrices.
- $\theta = \mathrm{vec}\left(\{\mathbf{W}_k\}_{k=1}^{L}\right) \in \mathbb{R}^D$



Prediction

$$f_\theta(\mathbf{x}) = \mathrm{sgn}\left(\mathbf{w}_L \mathrm{sgn}\left(\mathbf{W}_{L-1} \mathrm{sgn}\left(\ldots \mathrm{sgn}\left(\mathbf{W}_1 \mathbf{x}\right)\right)\right)\right),$$

Generalisation bound

Generalisation bound

For an arbitrary number of layers and neurons, with probability at least $1 - \delta$, for any $\theta \in \mathbb{R}^D$

$$R_{\mathrm{out}}(F_\theta) \leqslant \inf_{C>0} \left\{ \frac{1}{1-e^{-C}} \left(1 - \exp\left(-C R_{\mathrm{in}}(F_\theta) - \frac{\frac{1}{2}\|\theta - \theta_0\|^2 + \log \frac{2\sqrt{m}}{\delta}}{m} \right) \right) \right\},$$

where

$$R_{\mathrm{in}}(F_\theta) = \mathop{\mathbf{E}}_{\theta' \sim Q_\theta} R_{\mathrm{in}}(f_{\theta'}) = \frac{1}{m} \sum_{i=1}^{m} \left[\frac{1}{2} - \frac{1}{2} y_i F_\theta(\mathbf{x}_i) \right].$$

(A selection of) numerical results

Model name	Cost function	Train split	Valid split	Model selection	Prior
MLP–tanh	linear loss, L2 regularized	80%	20%	valid linear loss	-
PBGNet$_\ell$	linear loss, L2 regularized	80%	20%	valid linear loss	random init
PBGNet	**PAC-Bayes bound**	**100 %**	**-**	**PAC-Bayes bound**	**random init**
PBGNet$_{pre}$					
– pretrain	linear loss (20 epochs)	50%	-	-	random init
– final	PAC-Bayes bound	50%	-	PAC-Bayes bound	pretrain

Dataset	MLP–tanh		PBGNet$_\ell$		PBGNet			PBGNet$_{pre}$		
	E_S	E_T	E_S	E_T	E_S	E_T	Bound	E_S	E_T	Bound
ads	0.021	0.037	0.018	**0.032**	0.024	0.038	0.283	0.034	0.033	0.058
adult	0.128	0.149	0.136	**0.148**	0.158	0.154	0.227	0.153	0.151	0.165
mnist17	0.003	**0.004**	0.008	0.005	0.007	0.009	0.067	0.003	0.005	0.009
mnist49	0.002	**0.013**	0.003	0.018	0.034	0.039	0.153	0.018	0.021	0.030
mnist56	0.002	0.009	0.002	0.009	0.022	0.026	0.103	0.008	**0.008**	0.017
mnistLH	0.004	**0.017**	0.005	0.019	0.071	0.073	0.186	0.026	0.026	0.033

Thanks!

We have several PhD / postdoc / visiting researcher positions available in my group, based in London and affiliated with Inria and UCL.

Feel free to reach out!

`https://bguedj.github.io`