Tighter risk certificates for (probabilistic) neural networks

Omar Rivasplata
o.rivasplata@cs.ucl.ac.uk

01 July 2020

The crew

- María Pérez-Ortiz (UCL)
- Yours truly (UCL / DeepMind)
- Csaba Szepesvári (DeepMind)
- John Shawe-Taylor (UCL)

Overview of this talk

- Motivation
  - Classic NNs: weights
    - Probabilistic NNs: random weights
      - Highlights of experiments
        - Conclusions

What motivated this project

Blundell et al. (2015)

Weight Uncertainty in Neural Networks

Charles Blundell CBLUNDELL@GOOGLE.COM
Julien Cornebise JUCOR@GOOGLE.COM
Koray Kavukcuoglu KORAYK@GOOGLE.COM
Daan Wierstra WIERSTRA@GOOGLE.COM
Google DeepMind

- Variational Bayes : $\min_\theta KL(q_\theta(w) \| p(w|D))$
- Objective : $f(\theta) = \mathbb{E}_{q_\theta(w)}[\log(1/p(D|w))] + KL(q_\theta(w) \| p(w))$ (ELBO)
- Algorithm : 'Bayes by Backprop'

Thiemann et al. (2017)

A Strongly Quasiconvex PAC-Bayesian Bound

Niklas Thiemann NIKLASTHIEMANN@GMAIL.COM
Department of Computer Science, University of Copenhagen

Christian Igel IGEL@DI.KU.DK
Department of Computer Science, University of Copenhagen

Olivier Wintenberger OLIVIER.WINTENBERGER@UPMC.FR
LSTA, Sorbonne Universités, UPMC Université Paris 06

Yevgeny Seldin SELDIN@DI.KU.DK
Department of Computer Science, University of Copenhagen

- PAC-Bayes-lambda :

$$\mathbb{E}_{q_\theta(w)}[L(w)] \leq \frac{\mathbb{E}_{q_\theta(w)}[\hat{L}_n(w,D)]}{1-\lambda/2} + \frac{KL(q_\theta(w)\|p(w)) + C_n}{n\lambda(1-\lambda/2)} \qquad \lambda \in (0,2)$$

- Algorithm : $f(\theta,\lambda)$ = RHS, alternated optimization over θ and λ

Dziugaite & Roy (2017)

Computing Nonvacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data

Gintare Karolina Dziugaite
Department of Engineering
University of Cambridge

Daniel M. Roy
Department of Statistical Sciences
University of Toronto

- Optimized a classic PAC-Bayes bound
- Experiments on ‘binary MNIST’ ([0-4] vs. [5-9])
- Demonstrated non-vacuous risk bound values

Classic Neural Nets

What to achieve from data?

Use the available data to:

(1) learn a weight vector $\hat{w}$

(2) certify $\hat{w}$'s performance

- split the data, part for (1) and part for (2)?
- the whole of the data for (1) and (2) simultaneously?
  - ▹ self-certified learning!

$\boxed{\mathsf{ALG} : \mathcal{Z}^n \to \mathcal{W}}$ $\boxed{\mathcal{W} \to \mathcal{H}}$

- $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$
  $\mathcal{X}$ = set of inputs
  $\mathcal{Y}$ = set of labels
- $\mathcal{W} \subset \mathbb{R}^p$
  weight space
  $\hat{w} = \mathsf{ALG}(\text{data})$
- $\mathcal{H}$ function class
  predictors
  $h_{\hat{w}} : \mathcal{X} \to \mathcal{Y}$

data set: $D = (Z_1, \ldots, Z_n) \in \mathcal{Z}^n$ (e.g. training set)
a finite sequence of input-label examples $Z_i = (X_i, Y_i)$.

A measure of performance

Empirical risk: $\hat{L}_n(w) = \hat{L}_n(w, D) = \frac{1}{n} \sum_{i=1}^{n} \ell(w, Z_i)$

(in-sample error)

Tied to the choice of a loss function $\ell(w, z)$

- the square loss (regression)
- the 0-1 loss (classification)
- the cross-entropy loss (NN classification)
  - surrogate loss, nice properties

Empirical Risk Minimization

Training set error: $\hat{L}_{\mathrm{trn}}(w) = \frac{1}{n\_\mathrm{trn}} \sum_{Z_i \in D_{\mathrm{trn}}} \ell(w, Z_i)$

ERM: $\hat{w} \in \arg\min_{w} \hat{L}_{\mathrm{trn}}(w)$

Penalized ERM: $\hat{w} \in \arg\min_{w} \hat{L}_{\mathrm{trn}}(w) + \mathrm{Reg}(w)$

Generalization

If learned weight $\hat{w}$ does well on the train set examples...

...will it still do well on unseen examples?

data set: $D = (Z_1, \ldots, Z_n) \in \mathcal{Z}^n$

a finite sequence of input-label examples $Z_i = (X_i, Y_i)$.

Assumptions:

- A data-generating distribution $P \in M_1(\mathcal{Z})$.
- P is unknown, only the training set is given.
- The input-label examples are *i.i.d.* $\sim P$.

Population risk: $\boxed{L(w) = \mathbb{E}[\ell(w, Z)]} = \int_{\mathcal{Z}} \ell(w, z)\, dP(z)$

(out-of-sample)

Certifying performance: test set error

Test set error: $\hat{L}_{\mathrm{tst}}(\hat{w}) = \frac{1}{n\_\mathrm{tst}} \sum_{Z_i \in D_{\mathrm{tst}}} \ell(\hat{w}, Z_i)$

- $\hat{w}$ obtained from the training set
- test set not used for training
- $\hat{L}_{\mathrm{tst}}(\hat{w})$ serves as estimate of $L(\hat{w})$
- Note: $L(\hat{w})$ remains unknown!

Certifying performance: confidence bound

Risk upper bound: For any given $\delta \in (0,1)$, with probability at least $1-\delta$ over random datasets of size n, simultaneous for all w :

$$\boxed{L(w) \leq \hat{L}_n(w) + \epsilon(n,\delta)}$$

For $\hat{w} = \mathsf{ALG}(\text{train\_set})$ this gives: $L(\hat{w}) \leq \hat{L}_{\text{tst}}(\hat{w}) + \epsilon(n\_\text{tst}, \delta)$

Recommendable practice:

- report confidence bound
  together with your test set error estimate

Self-certified learning?

Risk upper bound: For any given $\delta \in (0, 1)$, with probability at least $1 - \delta$ over random datasets of size n, simultaneous for all w :

$$\boxed{L(w) \leq \hat{L}_n(w) + \epsilon(n, \delta)}$$

Alternative practice: Find $\hat{w}$ by minimizing the risk bound

- A form of regularized ERM
- the learned $\hat{w}$ comes with its own risk certificate
- best if the risk bound is non-vacuous, ideally tight!
- may avoid the need of data-splitting
- may lead to self-certified learning!

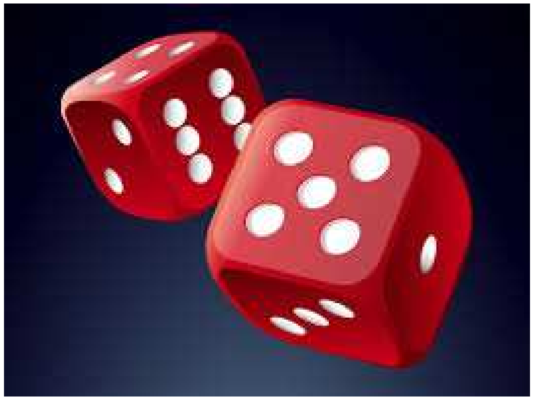
Probabilistic Neural Nets

Randomized weights

- Based on data D, learn a distribution over weights:

$$Q_D \in M_1(\mathcal{W}), \qquad Q_D = \mathsf{ALG}(\text{train\_set}).$$

- Predictions:
  - draw $w \sim Q_D$ and predict with the chosen w.
  - each prediction with a fresh random draw.



The risk measures $L(w)$ and $\hat{L}_n(w)$ are extended to Q by averaging:

$$Q[L] \equiv \int_{\mathcal{W}} L(w)\, dQ(w) = \mathbb{E}_{w \sim Q}[L(w)]$$

$$Q[\hat{L}_n] \equiv \int_{\mathcal{W}} \hat{L}_n(w)\, dQ(w) = \mathbb{E}_{w \sim Q}[\hat{L}_n(w)]$$

Two usual PAC-Bayes bounds

'prior' 'posterior'

Fix a distribution Q_0.

For any sample size n,
for any confidence parameter $\delta \in (0, 1)$,
with probability at least $1 - \delta$
over random samples (of size n)

simultaneously for all distributions Q

$$\boxed{Q[L] \leq Q[\hat{L}_n] + \sqrt{\frac{KL(Q\|Q_0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}}} \qquad \text{(PB-classic)}$$

$$\boxed{\mathrm{kl}(Q[\hat{L}_n]\|Q[L]) \leq \frac{KL(Q\|Q_0) + \log(\frac{2\sqrt{n}}{\delta})}{n}} \qquad \text{(PB-kl)}$$

Two more PAC-Bayes bounds

Fix a distribution Q_0. For any size n, for any confidence $\delta \in (0,1)$, with probability at least $1-\delta$ over random samples (of size n)

PB-quad: simultaneously for all distributions Q

$$\boxed{Q[L] \leq \left(\sqrt{Q[\hat{L}_n] + \frac{KL(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}} + \sqrt{\frac{KL(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}} \right)^2}$$

PB-lambda: simultaneously for all distributions Q and $\lambda \in (0,2)$

$$\boxed{Q[L] \leq \frac{Q[\hat{L}_n]}{1-\lambda/2} + \frac{KL(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{n\lambda(1-\lambda/2)}}$$

Cornerstone: change of measure inequality

Donsker & Varadhan (1975),   Csiszár (1975)

$$KL(Q\|Q_0) = \sup_{f:\mathcal{W}\to\mathbb{R}} \left\{ Q[f] - \log Q_0[e^f] \right\}$$

- Let $f : \mathcal{Z}^n \times \mathcal{W} \to \mathbb{R}$.   For a given Q_0 :

  $Q[f(D,w)] \leq KL(Q\|Q_0) + \log Q_0[e^{f(D,w)}]$.

- Apply Markov's inequality to $Q_0[e^{f(D,w)}]$.

- w.p. $\geq 1-\delta$ over the random draw of $D \sim P^n$, simultaneously for all distributions Q :

  $Q[f(D,w)] \leq KL(Q\|Q_0) + \log P^n[Q_0[e^{f(D,w)}]] + \log(1/\delta)$.

- Use with suitable f, upper-bound the exponential moment $P^n[Q_0[e^{f(D,w)}]]$.

Using a PAC-Bayes bound

- Use your favourite ALG to find $Q_D = \text{ALG}(\text{train\_set})$, and plug Q_D into the PAC-Bayes bound to certify its risk:

$$\boxed{Q_D[L] \leq Q_D[\hat{L}_n] + \sqrt{\frac{KL(Q_D \| Q_0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}}}$$

- Use the PAC-Bayes bound itself as a training objective:

$$\boxed{Q_D \in \arg\min_{Q} \; Q[\hat{L}_n] + \sqrt{\frac{KL(Q \| Q_0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}}}$$

Note: both uses illustrated here with PB-classic, but the same can be done with PB-quad or PB-lambda (or any other)

Training objectives

$$f_{\text{classic}}(Q) = Q[\hat{L}_n^{\text{ce}}] + \sqrt{\frac{KL(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}}$$

$$f_{\text{quad}}(Q) = \left(\sqrt{Q[\hat{L}_n^{\text{ce}}] + \frac{KL(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}} + \sqrt{\frac{KL(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}} \right)^2$$

$$f_{\text{lambda}}(Q, \lambda) = \frac{Q[\hat{L}_n^{\text{ce}}]}{1 - \lambda/2} + \frac{KL(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{n\lambda(1 - \lambda/2)}$$

Experiments

PAC-Bayes with Backprop

Algorithm 1 PAC-Bayes with Backprop (PBB)

Input:

μ_0 ▷ Prior center parameters (random init.)

ρ_0 ▷ Prior scale hyper-parameter

$Z_{1:n}$ ▷ Training examples (inputs + labels)

$\delta \in (0,1)$ ▷ Confidence parameter

$\alpha \in (0,1), T$ ▷ Learning rate; # of iterations

Output: Optimal μ, ρ ▷ Centers, scales

1: **procedure** PB_QUAD_GAUSS

2: $\mu \leftarrow \mu_0$ ▷ Set posterior centers to init. of prior

3: $\rho \leftarrow \rho_0$ ▷ Set posterior scale to ρ_0 hyperparam.

4: **for** $t \leftarrow 1 : T$ **do** ▷ Run SGD for T iterations.

5: Sample $V \sim \mathcal{N}(0, I)$

6: $W = \mu + \log(1 + \exp(\rho)) \odot V$

7: $f(\mu, \rho) = f_{\text{quad}}(Z_{1:n}, W, \mu, \rho, \mu_0, \rho_0, \delta)$

8: SGD gradient step using $\begin{bmatrix} \nabla_\mu f \\ \nabla_\rho f \end{bmatrix}$

9: **return** μ, ρ

Prior mean at the random initialization

- (PAC-Bayes) prior $Q_0 = \text{Gauss}(w_0, \Sigma_0)$
  $\Sigma_0 = \lambda_0 I$ (λ_0 is hyperparameter)
  w_0 = randomly initialized weights
- (PAC-Bayes) posterior $Q_D = \text{Gauss}(w, \Sigma)$
  w, Σ learned by PAC-Bayes with Backprop

Experiments (ours) on MNIST

f_{quad}

Test acc. = 86.36
Test error = 0.1364
RUB value = 0.24107

f_{classic} (cf. D & R (2017))

Test acc. = 84.22
Test error = 0.1578
RUB value = 0.24375

Prior mean learned from data

- (PAC-Bayes) prior $Q_0 = \mathrm{Gauss}(w_0, \Sigma_0)$
  $\Sigma_0 = \lambda_0 I$ (λ_0 is hyperparameter)
  w_0 = ERM on a split of the data
- (PAC-Bayes) posterior $Q_D = \mathrm{Gauss}(w, \Sigma)$
  w, Σ learned by PAC-Bayes with Backprop

Experiments (ours) on MNIST

f_{quad}

Test acc. = 97.89
Test error = 0.0211
RUB value = 0.04588

f_{classic} (cf. D & R (2018))

Test acc. = 97.21
Test error = 0.0279
RUB value = 0.06029

Closing remarks

Bayesian Learning

posterior Q_D, density $q_D(w)$ prior Q_0, density $q_0(w)$

$$q_D(w) = \mathcal{L}(D|w)\ q_0(w)\ /C$$

- Bayes rule update on prior to form posterior
  - likelihood factor $\mathcal{L}(D|w)$
- principled approach, e.g. MAP learning
- derive learning algorithms
  - balance 'fit to data' and 'fit to prior'

A bit more general:
"temperature" $\lambda > 0$

$$q_D(w) = \mathcal{L}(D|w)^\lambda \, q_0(w) \; / C$$

Even more general:
data-dependent factor $\mathcal{F}$

$$q_D(w) = \mathcal{F}(D, w) \, q_0(w)$$

- P.G. Bissiri, C.C. Holmes, S.G. Walker (2016)
  A general framework for updating belief distributions

$q_D(w)$ $\boxed{\text{no update factor}}$ $q_0(w)$

- more general than generalized Bayes
- increased flexibility in choice of distributions
- balance $q_D[\hat{L}_n]$ and $KL(q_D \| q_0)$
  - ‘fit to data’ versus ‘fit to prior’

Future

- choice of distributions
- understand properties
- scaling to larger problems?
- architecture vs. PAC-Bayes bounds?
- problem-specific PAC-Bayes bounds?

Thank you!

Wait...

some PAC-Bayes history

- J. Shawe-Taylor & R.C. Williamson (1997)
  A PAC analysis of a Bayesian estimator
- D.A. McAllester (1998)
  Some PAC-Bayesian Theorems
- D.A. McAllester (1999)
  PAC-Bayesian Model Averaging
- J. Langford & M. Seeger (2001)
  Bounds for Averaging Classifiers
- J. Langford & R. Caruana (2002)
  (Not) Bounding the True Error
- M. Seeger (2002)
  PAC-Bayesian generalization bounds for gaussian processes

some more PAC-Bayes history

- J. Langford & J. Shawe-Taylor (2002)
  PAC-Bayes & Margins
- D.A. McAllester (2003)
  Simplified PAC-Bayesian Margin Bounds
- A. Maurer (2004)
  A note on the PAC Bayesian theorem
- J.-Y. Audibert (2004)
  A better variance control for PAC-Bayesian classification
- O. Catoni (2007)
  PAC-Bayesian supervised classification:
  The thermodynamics of statistical learning
- P. Germain, A. Lacasse, F. Laviolette, M. Marchand (2009)
  PAC-Bayesian learning of linear classifiers

some recent PAC-Bayes

- J. Keshet, D.A. McAllester, T. Hazan (2011)
  PAC-Bayesian approach for minimization of phoneme error rate
- A. Noy & K. Crammer (2014)
  Robust forward algorithms via PAC-Bayes and Laplace distributions
- P. Germain, F. Bach, A. Lacoste, S. Lacoste-Julien (2016)
  PAC-Bayesian theory meets Bayesian inference
- N. Thiemann, C. Igel, O. Wintenberger, Y. Seldin (2017)
  A Strongly Quasiconvex PAC-Bayesian Bound
- G.K Dziugaite & D. Roy (2017)
  Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data
- G.K Dziugaite & D. Roy (2018)
  Data-dependent PAC-Bayes priors via differential privacy

more recent PAC-Bayes

- O. Rivasplata, E. Parrado-Hernández, J. Shawe-Taylor, S. Sun, Cs. Szepesvári (2018)
  PAC-Bayes bounds for stable algorithms with instance-dependent priors

- P. Alquier & B. Guedj (2018)
  Simpler PAC-Bayesian bounds for hostile data

- S.S. Lorenzen, C. Igel, Y. Seldin (2019)
  On PAC-Bayesian Bounds for Random Forests

- G. Letarte, P. Germain, B. Guedj, F. Laviolette (2019)
  Dichotomize and Generalize: PAC-Bayesian Binary Activated Deep Neural Networks

- O. Rivasplata, V.M. Tankasali, Cs. Szepesvári (2019)
  PAC-Bayes with Backprop (in arXiv)

Thank you again!