Theoretical Principles of Deep Learning

Class 6: PAC-Bayes Generalization bounds

Hédi Hadiji

Université Paris-Saclay - CentraleSupelec

hedi.hadiji@l2s.centralesupelec.fr

22 January, 2024

Table of Contents

Plan

Last time

- Neural Tangent Kernels
- A generalization bound for the NTK

Today: A different type of generalization bounds: PAC Bayes.

Reading Material:

- *User-friendly introduction to PAC-Bayes bounds*, Pierre Alquier
- *Computing Nonvacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data*, Dziugaite and Roy

Reminder: generalization bounds for finite classes

Setting for today, binary classification with the 0-1 loss. In particular the loss is bounded in $[0, 1]$.

Reminder: Union bound + Hoeffding

If the hypothesis class $\mathcal{H}$ is finite, then with probability at least $1 - \delta$,

$$\text{for all } h \in \mathcal{H}, \quad R(h) - R_S(h) \leqslant \sqrt{\frac{\log|\mathcal{H}| + \log(1/\delta)}{2n}}$$

In particular, if we use an ERM, then the resulting classifier is **Probably Approximately Correct**, in the sense that it is probably almost as good as the best classifier in $\mathcal{H}$.

With enough data points, $n \geqslant \log|\mathcal{H}|$ and learning occurs... but in practice $n \ll \log|\mathcal{H}|$. Then the bound above says **nothing** since we know already that $R(h) \leqslant 1$.

Vacuous bounds

Issue 0: we use infinite classes of classifiers. Even taking finite precision of computers into account, for a model on MNIST with three hidden layers with 100 nodes each, we have
$784 \times 100 + 2 \times 100 \times 100 + 100 \times 10 + 310 = 99\,710$ trainable parameters, so

$$\log |\mathcal{H}| \leqslant \log 2^{32 \times 99\,710} \approx 2.2 \times 10^6 .$$

So the union bounds guarantees that you can learn on MNIST using an MLP if you have millions of data points.

First direction of improvement: taking the structure of $\mathcal{H}$ into account, and deriving bounds that depend on a better notion of model complexity than $\log |\mathcal{H}|$.

Reminder 2: Model complexity bounds are still vacuous

Pure Rademacher complexity bound

With probability at least $1 - \delta$,

$$\text{for all } h \in \mathcal{H}, \quad R(h) - R_S(h) \leqslant 2\,\mathcal{R}_S(\mathcal{H}) + \sqrt{\frac{2\log(2/\delta)}{2n}}$$

Issue 1: Neural nets operate well in regimes in which they interpolate any kind of data, i.e., in which $\mathcal{R}_S(\mathcal{H}) \approx 1$.

If we want to explain deep learning in practical regimes, generalization bounds need to be data-dependent or algorithm-dependent.

We discussed a data-dependent improvement on Rademacher complexity bounds for kernel linear regression that can explain generalization in some regimes, in which neural nets stay close to their initialization.

Another type of improvement: non uniform bounds

We are not really interested in any $h \in \mathcal{H}$, only on the output of our training algorithm. Idea: Make it non-uniform.

Given a distribution over the classifiers P, called a prior by analogy with Bayesian thinking.

Union bound with a prior

Let P be a probability distribution over $\mathcal{H}$. With probability at least $1 - \delta$,

$$\text{for any } h \in \mathcal{H} \quad R(h) - R_S(h) \leqslant \sqrt{\frac{\log(1/P(h)) + \log(1/\delta)}{2n}}$$

P needs to be chosen before observing the data.

We recover the standard bound with a uniform prior. If P puts a lot of weight on the output of our algorithm, then the generalization bound is tight. (But if we could predict the outcome of the algorithm before seeing the data, then generalization should be easy.)

PAC-Bayes bounds

Modify the PAC learning framework to incorporate distributions over $\mathcal{H}$:

- A **prior** P measuring the belief that a hypothesis is good before seeing the data
- A **posterior** Q updated from the data.

PAC-Bayes style generalization bounds

Given a prior P, with probability at least $1 - \delta$,

$$\text{for any posterior } Q, \quad d(R(Q), R_S(Q)) \leqslant f(P, Q, \delta, \dots)$$

Notation: $\quad R(Q) = \int R(h) \mathrm{d}Q(h) \quad \text{and} \quad R_S(Q) = \int R_S(h) \mathrm{d}Q(h)$

(Be careful if you know about Bayesian statistics, this is quite different.)

Most important conceptual shift is the movement to distributions over hypotheses, a.k.a. **stochastic classifiers**.

Why PAC-Bayes bounds?

By optimizing over posteriors, PAC-Bayes bounds give the tightest generalization bounds for multiple settings.

In particular, they provide the only known non-vacuous generalization bounds for deep neural networks in practically relevant regimes (e.g. on MNIST). This has sparked a lot of interest in recent years.

Stochatic estimators can sometimes be derandomized, although derandomizing bounds in a tight way for neural nets is still open.

Aside: measuring difference between distributions

Definition (Kullback-Leibler divergence)

Let P and Q be two probability distributions, then

$$\mathrm{KL}(Q,P) = \begin{cases} \int \log\left(\frac{\mathrm{d}Q}{\mathrm{d}P}\right)\mathrm{d}Q & \text{if} \quad P \ll Q \\ +\infty & \text{otherwise} \end{cases}$$

Kullback-Leibler is **not** a distance

- not symmetric
- no triangle inequality

Better interpretation is that Kullback-Leibler is a measure of 'How surprised you are if you think a variable has distribution P but its true distribution is Q'.

Useful facts about the Kullback-Leibler divergence

Pinsker's inequality

$$\sup_{A \text{ event}} |Q(A) - P(A)| \leqslant \sqrt{\frac{\mathrm{KL}(Q, P)}{2}}$$

Duality / Donsker-Varadhan

$$\mathrm{KL}(Q, P) = \sup_{X \text{ bounded}} \left\{ \mathbb{E}_P[X] - \log \mathbb{E}_Q[e^X] \right\}$$

Joint convexity

$$\mathrm{KL}\left(\frac{Q_1 + Q_2}{2}, \frac{P_1 + P_2}{2} \right) \leqslant \frac{\mathrm{KL}(Q_1, P_1) + \mathrm{KL}(Q_2, P_2)}{2}$$

Special KLs

Bernoulli KL

For any $p, q \in [0, 1]$,

$$\mathrm{KL}\big(\,\mathrm{Ber}(q), \mathrm{Ber}(p)\big) = q \log \frac{q}{p} + (1-q) \log \frac{1-q}{1-p}$$

and we denote by this by $\mathrm{kl}(q, p)$.

Gaussian KL

If $P = \mathcal{N}(w_p, \sigma^2)$ and $Q = \mathcal{N}(w_q, \sigma^2)$, then

$$\mathrm{KL}\big(Q, P\big) = \frac{(w_q - w_p)^2}{2}$$

Table of Contents

Seeger's bound

Theorem (Seeger's bound)

With probability at least $1-\delta$*,*

$$\textit{for any posterior } Q, \quad \mathrm{kl}\big(R_S(Q), R(Q)\big) \leqslant \frac{\mathrm{KL}(Q,P) + \log(2\sqrt{n}/\delta)}{n}.$$

Proof: Chernov bound + binomal deviations.

Consequence: by Pinsker for the Bernoulli divergence, w.p. at least $1-\delta$:

$$R(Q) - R_S(Q) \leqslant \sqrt{\frac{\mathrm{KL}(Q,P) + \log(2\sqrt{n}/\delta)}{2n}}$$

also called McAllester's bound.

From bounds to algorithms

Given a prior and a bound, we may look for the posterior that minimizes the bound,

$$\widehat{Q} \in \operatorname{argmin}_Q \left\{ R_S(Q) + \sqrt{\frac{\mathrm{KL}(Q, P) + \log(2\sqrt{n}/\delta)}{2n}} \right\}$$

Even though this can be hard to minimize exactly, you know that the true error bound of any stochastic hypothesis $\widehat{Q}$ is less than the bound.

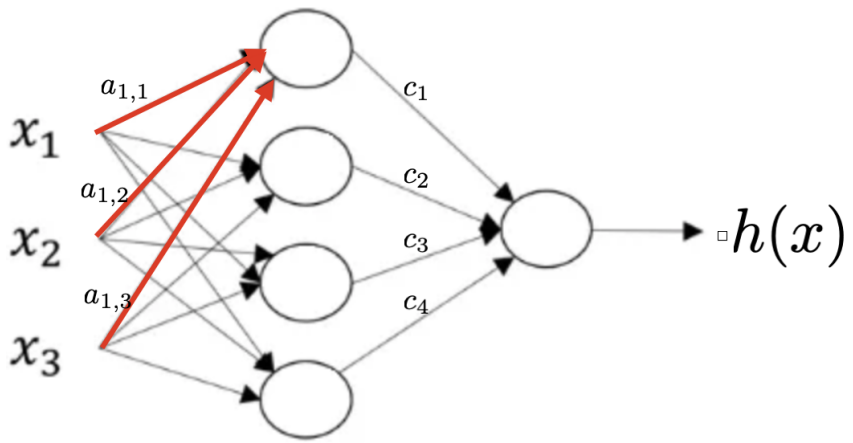
In rest of this class, we will describe how to obtain non-vacuous bounds for neural networks, thanks to a combination of tricks.

See Pierre Alquier's notes + the original paper by Dziugaite and Roy.

Table of Contents

Non vacuous bounds for deep nets

Identify networks with their weights and biases. I.e., fix an architecture, and parameterize the corresponding hypothesis space, by its weights and biases $w \in \mathbb{R}^d$.



Look for a **stochastic neural network** or **posterior** Q, in other words, a probability distribution over the space of weights and biases, R^d

Remark: none of this is specific to neural networks.

Plan for non-vacuous bounds

Here is the list of technical ingredients we will apply to obtain a non-vacuous bound

- Tight evaluation of the Bernoulli KL
- Convex surrogate loss
- Initialize posterior from a trained network
- Nice parameterization of the posteriors
- Data-dependent priors

Ingredient 1: Inverting the Bernoulli KL

Instead of using Pinsker's inequality to lower bound the kl, compute exactly the bound

$$\mathrm{kl}^{-1}(q, B) = \sup \left\{ p \in [0,1] \mid \mathrm{kl}(q,p) \leqslant B \right\}$$

(kl is convex in each argument, so non-decreasing in p for $p \geqslant q$)
to rewrite Seeger's bound as

$$R(Q) \leqslant \mathrm{kl}^{-1}\left(R_S(Q), \frac{\mathrm{KL}(Q,P) + \log(2\sqrt{n}/\delta)}{n} \right).$$

kl^{-1} can computed efficiently using Newton's method, initialized at the Pinsker upper bound.

Ingredient 2: Surrogate losses for optimization

Replace the 0-1 loss by the logistic loss, a convex (in h) upper bound

$$\mathbb{1}\{h(x) \neq y\} \leftarrow \frac{1}{\log 2} \log\left(1 + e^{-yh(x)}\right)$$

to make optimization possible.
Then it suffices to upper bound the bound with the surrogate loss

$$R(Q) \leqslant \mathrm{kl}^{-1}\left(R_S(Q),\ \frac{\mathrm{KL}(Q,P) + \log(2\sqrt{n}/\delta)}{n}\right)$$
$$\leqslant \mathrm{kl}^{-1}\left(\tilde{R}_S(Q),\ \frac{\mathrm{KL}(Q,P) + \log(2\sqrt{n}/\delta)}{n}\right).$$

Ingredient 3: Data-dependent priors

In principle, priors should be chosen independently of the data. However

if we have K different priors, then by a union bound, we can have bounds over all priors simultaneously, at the cost of replacing δ by δ/K, i.e., for a $\log K$ additive term. In fact, we can even have a prior over priors (!). Set

$\mu(\lambda_i)$ to be a probability distribution over a grid of values of λ. I.e. **for all** λ_i **simultaneously,** for all posteriors Q

$$R(Q) \leqslant \mathrm{kl}^{-1}\left(\tilde{R}_S(Q), \frac{\mathrm{KL}(Q, P_{\lambda_i}) + \log(2\sqrt{n}/\delta) + \log(1/\mu(\lambda_i))}{n}\right).$$

In particular, we can **also optimize over** λ**.** We use the grid:

$$\lambda_i = \lambda_{\max}\alpha^i \quad \text{and} \quad \mu(\lambda_i) = \frac{6}{\pi^2 i^2} = \frac{6}{\pi^2}\left(\frac{\log\alpha}{\log(\lambda_i/\lambda_{\max})}\right)^2$$

The extra cost is a loglog term in a square root.

Ingredient 4: Gaussian distributions

KL between Gaussians can be written in closed form:

$$\frac{1}{2}\left(\mathrm{Tr}(\Sigma_p^{-1}\Sigma_q) - d + (\mu_p - \mu_q)^\top \Sigma_p^{-1}(\mu_p - \mu_q) + \ln\left(\frac{\det \Sigma_p}{\det \Sigma_q}\right)\right)$$

Thus let us restrict our attention to Gaussian priors and posteriors.

$$P_\lambda = \mathcal{N}(w_0, \lambda^2 I)$$

We also restrict our choice of posterior to Gaussian posteriors with diagonal covariance, for $w \in \mathbb{R}^d$ and $s \in \mathbb{R}^d$

$$Q_{w,s} = \mathcal{N}(w, \mathrm{Diag}(s_i^2))$$

Then

$$\mathrm{KL}(Q_{w,s}, P_\lambda) = \frac{\|s\|^2}{2\lambda^2} - \frac{d}{2} + \frac{\|w - w_0\|^2}{2\lambda^2} + \sum_{i=1}^{d} \log \frac{\lambda}{s_i}$$

Now apply Pinsker

$$\mathrm{kl}^{-1}\left(\tilde{R}_S(Q_{w,s}),\ \frac{\mathrm{KL}(Q_{w,s}, P_{\lambda_i}) + \log(2\sqrt{n}/\delta) + \log \mu(\lambda_i)}{n}\right)$$
$$\leqslant \tilde{R}_S(Q_{w,s}) + \sqrt{\frac{\mathrm{KL}(Q_{w,s}, P_{\lambda_i}) + \log(2\sqrt{n}/\delta) + \log(1/\mu(\lambda_i))}{2n}}$$

Final optimization trick: use unbiased estimates of the gradient of $\tilde{R}_S(Q_{w,s})$ by sampling $\xi \sim \mathcal{N}(0, I_p)$ and take gradients of $\tilde{R}_S(h(w + \xi s))$, since

$$\mathbb{E}_\xi\left[\nabla_{w,s}\tilde{R}_S\big(h(w + \xi s)\big)\right] = \nabla_{w,s}\tilde{R}_S(Q_{w,s})\,.$$

Summary up to now

Algorithm 1 PAC-Bayes bound optimization by SGD

Input:

$w_0 \in \mathbb{R}^d$ ▷ Network parameters (random init.)
$w \in \mathbb{R}^d$ ▷ Network parameters (SGD solution)
S_m ▷ Training examples
$\delta \in (0,1)$ ▷ Confidence parameter
$b \in \mathbb{N}, c \in (0,1)$ ▷ Precision and bound for λ
$\tau \in (0,1), T$ ▷ Learning rate; # of iterations

Output: Optimal w, s, λ ▷ Weights, variances

procedure PAC-BAYES-SGD
  $\varsigma \leftarrow \mathrm{abs}(w)$ ▷ where $s(\varsigma) = \mathrm{e}^{2\varsigma}$
  $\varrho \leftarrow -3$ ▷ where $\lambda(\varrho) = \mathrm{e}^{2\varrho}$
  $\mathrm{B}(w, s, \lambda, w') = \breve{e}(h_{w'}, S_m) + \sqrt{\frac{1}{2} B_{\mathrm{RE}}(w, s, \lambda)}$
  for $t \leftarrow 1, T$ **do** ▷ Run SGD for T iterations.
    Sample $\xi \sim \mathcal{N}(0, I_d)$
    $w'(w, \varsigma) = w + \xi \odot \sqrt{s(\varsigma)}$
    ▷ Gradient step
    $$\begin{bmatrix} w \\ \varsigma \\ \varrho \end{bmatrix} \mathrel{-}= \tau \begin{bmatrix} \nabla_w B(w, s(\varsigma), \lambda(\varrho), w'(w, \varsigma)) \\ \nabla_\varsigma B(w, s(\varsigma), \lambda(\varrho), w'(w, \varsigma)) \\ \nabla_\varrho B(w, s(\varsigma), \lambda(\varrho), w'(w, \varsigma)) \end{bmatrix}$$
  return $w, s(\varsigma), \lambda(\varrho)$

Source: Dziugaite and Roy, 2017

Computing the final bound

Once we have fixed values for the prior and posterior, the true bound is still intractable because of $R_S(Q_{w,s})$.

Theorem (Bernoulli Concentration)

If $X_1, \dots, X_m \in [0,1]$ are i.i.d. with mean μ, then w.p. at least $1-\delta'$,

$$\mathrm{kl}(\bar{X}_m, \mu) \leqslant \frac{\log 2/\delta'}{m}$$

Approximate $R_S(Q_{w,s})$ by Monte-Carlo/Hoeffding, sample m hypotheses $\sim Q_{w,s}$ and compute the error of these hypotheses, then w.p. $\geqslant 1-\delta'$,

$$R_S(Q_{w,s}) \leqslant \mathrm{kl}^{-1}\left(R_S(\widehat{Q}_{w,s}), \frac{\log 2/\delta'}{m}\right)$$

Final bound

Final bound

Use the PAC-Bayes SGD to find (w, s, λ_i) that minimize the surrogate bound. Then w.p. $\geqslant 1 - \delta - \delta'$,

$$R(Q_{w,s}) \leqslant \mathrm{kl}^{-1}\left(R_S(Q_{w,s}),\ \frac{\mathrm{KL}(Q_{w,s}, P_{\lambda_i}) + \log(2\sqrt{n}/\delta) + \log(1/\mu(\lambda_i))}{n}\right)$$

and if $\widehat{Q}_{w,s}$ is the empirical distribution from m samples from $Q_{w,s}$,

$$R_S(Q_{w,s}) \leqslant \mathrm{kl}^{-1}\left(R_S(\widehat{Q}_{w,s}),\ \frac{\log(2/\delta')}{m}\right)$$

(kl^{-1} is non-decreasing in its first argument, so we can plug the second bound into the first one.)

Local conclusion

If this approach works, it provides a stochastic network, i.e., a distribution over weights for a given architecture, **with a guaranteed generalization bound, that is computable from the data.**

It guarantees that, given this posterior distribution Q, if one samples a network from Q, then w.p. $1 - \delta$, the expected loss they will incur is at most $B(w, s, \lambda, \delta)$.

Makings these bounds tighter and applying them to non-stochastic classifiers is an active research direction.

Results

Experiment	T-600	T-1200	T-300^2	T-600^2	T-1200^2	T-600^3	R-600
Train error	0.001	0.002	0.000	0.000	0.000	0.000	0.007
Test error	0.018	0.018	0.015	0.016	0.015	0.013	0.508
SNN train error	0.028	0.027	0.027	0.028	0.029	0.027	0.112
SNN test error	0.034	0.035	0.034	0.033	0.035	0.032	0.503
PAC-Bayes bound	0.161	0.179	0.170	0.186	0.223	0.201	1.352
KL divergence	5144	5977	5791	6534	8558	7861	201131
# parameters	471k	943k	326k	832k	2384k	1193k	472k
VC dimension	26m	56m	26m	66m	187m	121m	26m

Table 1: Results for experiments on binary class variant of MNIST. SGD is either trained on (T) true labels or (R) random labels. The network architecture is expressed as N^L, indicating L hidden layers with N nodes each. Errors are classification error. The reported VC dimension is the best known upper bound (in millions) for ReLU networks. The SNN error rates are tight upper bounds (see text for details). The PAC-Bayes bounds upper bound the test error with probability 0.965.

Source: Dziugaite and Roy, 2017

Insights from these results?

Why does this method good given tight(ish) generalization bounds?

One sugggested explanation is that SGD brings the network to wide minima of the loss.

- Broadly speaking, this method will work when we can compute a posterior such that both $R_S(Q)$ and $\mathrm{KL}(Q, P)$ are small at the same time.
- SGD brings the weight to a wide minimum, around which we take a wide posterior without degrading performance, leaving room to fit the prior.

Table of Contents

Conclusion and Next time

Today

- Introduced the PAC-Bayes framework for generalization
- Introduced a PAC-Bayes generalization bound
- Described a procedure to obtain stochastic neural networks with tight bounds

In lab: implement this on binary FashionMNIST