Understanding the Generalization of Deep Neural Networks through PAC-Bayes bounds: A Survey

Andres Potapczynski, Sanae Lotfi, Anthony Chen, Chris Ick [*]
Center for Data Science
New York University
New York, NY 11238

Abstract

Guaranteeing that neural networks generalize to unseen data is of foremost importance to their success in practical applications. However, achieving these guarantees is a difficult task as many of the generalization bounds used for deep neural networks depend on the dimensionality of the parameters space, making them vacuous. In general, bounds that only rely on the number of parameters and/or *uniformly* consider equally *all* representable functions of a given neural network architecture do not result in practical complexity measurements for generalization. A framework that *can* concentrate on certain function representations is PAC-Bayes. In this survey, we analyze current progress and techniques that employ PAC-Bayes to yield nonvacuous generalization bounds for neural networks as well as some of the recent advancements that enable bounds for heavy-tailed or time-dependent distributions.

1 Introduction

Learning with deep neural networks has enjoyed huge empirical success in recent years across a wide variety of tasks. Despite being a complex, non-convex optimization problem, simple methods such as stochastic gradient descent (SGD) are able to recover good solutions that minimize the training error. More surprisingly, the networks learned this way exhibit good generalization behavior, even when the number of parameters is significantly larger than the amount of training data. In such an over-parametrized setting, the objective has multiple optima that minimize the training error, but many of those solutions do not generalize well. Hence, just minimizing the training error is not sufficient for learning: picking the wrong minima can lead to bad generalization behavior. In such situations, generalization depends implicitly on the algorithm used to minimize the training error. Different algorithmic choices for optimization such as the initialization, update rules, learning rate, and stopping condition, will lead to different global minima with different generalization behavior.

Metrics that rely on parameter counting are divorced on how the parameters of the model interact with the fit to the data and how compressible the models can become. For example, Zhou et al. (2019) show how flexible neural networks architectures can fit random labels, but then it is not possible to quantize and prune its weights without losing accuracy. This in contrast to models trained on actual data, which are substantially compressible (over 90% of the weights can be pruned without a significant loss in accuracy). This suggests that by itself, the number of parameters will not indicate which models will generalize better nor will it explain why certain solutions found during optimization lead to better generalization.

As an alternative to parameter counting metrics, we survey the PAC-Bayes framework. This survey is structured as follows: first, we revisit the classical PAC-Bayes formulations in order to understand their limitations. We discuss the bounds in McAllester (1999), Shawe-Taylor and Williamson (1997)

[*] Author order by order of contribution in report

and we also explore the improvements on these initial bounds for classification problems made by Langford and Seeger (2001). We conclude this section with the more general bounds developed by Catoni (2007).

Next, we survey how the generalization of neural networks has been studied through the lens of PAC-Bayes and explain the techniques developed to yield nonvacuous bounds. We motive this section by the empirical analysis of neural network generalization done by **?** and contrast it with the explanations provided by Neyshabur et al. (2017a) that depend on a combination of ideas including PAC-Bayes and loss-landscape flatness. We then study the optimization ideas of Dziugaite and Roy (2017) where they propose a loss objective that uses a differential PAC-Bayes bound as a component; resulting in the first nonvacous bounds for neural networks. We finalize this section with an exposition on the prunining and quantization techniques developed in Zhou et al. (2019). These last compression techniques resulted in nonvacuous bounds for neural networks on ImageNet (Deng et al., 2009) and also created, for the first time, a connection between compression and generalization.

To conclude our survey, we explore relevant recent developments such as PAC-Bayes derived objectives for training probabilistic neural nets (Pérez-Ortiz et al., 2021), avoiding randomised predictors through the use of margins (Biggs and Guedj, 2022), using informative priors (Letarte et al., 2019), and extending them to graph neural networks (Liao et al., 2021). Additionally, we study the computational aspects of PAC learning highlighting recent polynomial-time provable algorithms for learning simple families of neural networks as in Diakonikolas et al. (2020), and highlight some additional studies on PAC learning in simple neural nets in Vempala and Wilmes (2018) and Goel and Klivans (2018).

2 Background

In this section we motivate the PAC-Bayes framework and discuss the different bounds that have been derived in the literature. Assume we have a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$, a parametric model $f_\theta : \mathcal{X} \to \mathcal{Y}$ (most likely a neural network) where $\theta \in \Theta$ (and Θ is the parameter space of all the weights and biases of the neural network) and, finally, data $\mathcal{D}$ generated from the distribution $P_\mathcal{D}$ which we do not have direct access to. We would like to understand how our model will generalize to unseen data, in other words, we would like to understand the behaviour of

$$R\left(f_\theta\right) = \underset{(X,Y)\sim P_\mathcal{D}}{\mathbb{E}}\left[\ell\left(f_\theta\left(X\right), Y\right)\right].$$

To pursue a mathematical analysis of $R\left(f_\theta\right)$ we usually assume that either ℓ is bounded (as in the case of classification) or that the random variable $\ell\left(f_\theta\left(X\right), Y\right)$ is sub-Gaussian or sub-exponential with parameter σ^2 (this last assumption is what we will use in this section).

Given these assumptions, a natural approach that is common practice is to derive a Probably Approximately Correct (PAC) bound on $R\left(f_\theta\right)$ which would guarantee that its value will be bounded by the empirical risk $\hat{R}\left(f_\theta\right) = \frac{1}{N}\sum_{i=1}^{N} \ell\left(f_\theta\left(x_i\right), y_i\right)$ with a high probability. This PAC bound usually takes the following form, for a given confidence value $\delta \in (0, 1)$

$$P\left[R\left(f_{\hat{\theta}}\right) \leq \hat{R}\left(f_{\hat{\theta}}\right) + \sigma^2\sqrt{\frac{\log\left(\frac{1}{\delta}\right) + \log|\Theta|}{N}}\right] \geq 1 - \delta \tag{1}$$

where $|\Theta|$ denotes some measure on the space of parameters like VC-dimension or parameter counting and $\hat{\theta}$ is selected via an algorithm $\mathcal{A}$ which takes as input data $\mathcal{D}$ and output a choice $\hat{\theta}$. Eq (1) exemplifies the limitation of the PAC framework to neural networks as $|\Theta|$ is usually very large.

In the previous result, the value of $|\Theta|$ comes assuming that the bound should hold uniformly for Θ. Through this lens, the PAC-Bayes framework then considers if we can achieve tighter bounds by not considering the parameter space uniformly but rather by measuring it differently, through a prior and posterior distribution on Θ.

Before jumping into some specific forms of the PAC-Bayes bounds it is worth reflecting on a key lemma used to derive the bounds: the Donsker-Varadhan lemma.

Lemma 1 (Donsker-Varadhan)**.** *For any measurable, bounded function $h : \Theta \to \mathbb{R}$ we have:*

$$\log \mathop{\mathbb{E}}_{\theta\sim P}\left[e^{h(\theta)}\right] = \sup_{Q\in\mathcal{P}(\Theta)}\left[\mathop{\mathbb{E}}_{\theta\sim Q}\left[h\left(\theta\right)\right] - \mathbb{KL}\left(Q,P\right)\right]$$

where $\mathcal{P}\left(\Theta\right)$ denotes the space of distributions over Θ. Moreover, the supremum is reached for the Gibbs measure defined by

$$Q^{\star}\left(\theta\right) = \frac{e^{h(\theta)}}{\int e^{h(\tau)}P\left(\tau\right)d\tau}.$$

This lemma is crucial to understand the following PAC-Bayes bounds for two reasons. First, it allows us to change the base measure given by the prior P to the posterior Q by adding a divergence between the two measures exemplified in the $\mathbb{KL}\left(Q,P\right)$ term. This is vital as our prior measure cannot be informed by the data $\mathcal{D}$ and thus we obtain the flexibility of choosing another measure. Second, it guides us in choosing the best posterior Q which is proportional to $e^{h(\theta)}$. In our generalization study $h\left(\theta\right) = \hat{R}\left(f_{\theta}\right)$ thus, the Gibbs posterior chooses where to concentrate its mass given the distribution of the empirical risk under the prior.

First PAC-Bounds The first PAC-Bayes bound appears in the work of McAllester (1999) which was inspired by some results in Shawe-Taylor and Williamson (1997). The original paper focused on countable Θ. The first bound states that for any $Q \in \mathcal{P}\left(\Theta\right)$ and $\delta \in (0,1)$

$$\mathop{\mathbb{E}}_{\theta\sim Q}\left[R\left(f_{\theta}\right)\right] \leq \mathop{\mathbb{E}}_{\theta\sim Q}\left[\hat{R}\left(f_{\theta}\right)\right] + \sqrt{\frac{\mathbb{KL}\left(Q,P\right) + \log\frac{1}{\delta} + \frac{5}{2}\log N + 8}{2N-1}}. \tag{2}$$

Is this bound meaningfully different from the previous PAC bound that we exposed in the beginning of the section? The answer severely depends on the choice of Q and P. For example, assuming Θ is finite, if we let $Q = I\left[\theta = \hat{\theta}\right]$, where $\hat{\theta}$ is the empirical risk minimizer, and $P\left(\theta\right) = \frac{1}{|\Theta|}$, that is, uniform on Θ then we have that

$$\mathbb{KL}\left(Q,P\right) = \log\left(\frac{Q(\hat{\theta})}{1/\left|\Theta\right|}\right)Q(\hat{\theta}) = \log\left|\Theta\right|$$

and we recover the same bound as eq. (1) (up to some constant terms). This last case also highlights why the PAC bound is not tight, there is minimal overlap of Q and of P: they only overlap on a single point $\hat{\theta}$. Yet, even if the posterior and prior have good overlap on Θ, we can still get a vacuous bound. To illustrate this point, image that $Q = \mathcal{N}(\hat{\theta}, \text{diag}(\sigma^2)I)$, where $\hat{\theta}$ is a minimizer found during optimization and $P = \mathcal{N}\left(0, I\right)$. Then

$$\mathbb{KL}\left(Q,P\right) = \frac{1}{2}\sum_{i=1}^{K}\left(\sigma_i^2 + \hat{\theta}_i^2 - 1 - \log\sigma_i^2\right)$$

and so if all the parameters of the neural network found during optimization $\hat{\theta}_i \neq 0$ then the $\mathbb{KL}\left(Q,P\right)$ would be some form of parameter counting (as there would be K summands, where, in this case, K is the number of parameters in the neural network). Thus the difficulty of achieving "tight bounds" is passed onto $\mathbb{KL}\left(Q,P\right)$. In the next section we explore how to actually choose Q and P in order to make the PAC-Bounds as tight as possible.

Maurer's Bound (an improvement on Seeger and Langford's Bound) The following bound if based on work by Langford and Seeger (2001) which then was refined by Maurer (2004). This bound focuses on classification losses, that is, $\ell\left(f_{\theta}\left(x\right), y\right) = I\left[f_{\theta}\left(x\right) \neq y\right]$. Define the following function $\text{kl} : [0,1] \times [0,1]$ as

$$\text{kl}\left(q,p\right) = q\log\left(q/p\right) + \left(1-q\right)\log\left(\left(1-q\right)/\left(1-p\right)\right)$$

which is simply the KL divergence between two Bernoulli random variables. Using, this function, then the Maurer bound states that

$$\text{kl}\left(\mathop{\mathbb{E}}_{\theta\sim Q}\left[\hat{R}\left(f_{\theta}\right)\right], \mathop{\mathbb{E}}_{\theta\sim Q}\left[R\left(f_{\theta}\right)\right]\right) \leq \frac{\mathbb{KL}\left(Q,P\right) + \log\frac{1}{\delta} + \log\left(2\sqrt{N}\right)}{N}.$$

To gain some intuition, consider the common case where the neural network is achieving almost 0 training error. In this case the bound would become

$$\mathop{\mathbb{E}}_{\theta\sim Q}\left[R\left(f_\theta\right)\right] \leq 1-\exp\left(-\frac{\mathbb{KL}\left(Q,P\right)+\log\frac{1}{\delta}+\log\left(2\sqrt{N}\right)}{N}\right).$$

When compared to eq (2), we see that the amount of data N plays a higher role. Interestingly, a relaxation of this bound was used in Dziugaite and Roy (2017) to compute the first non-vacuous bounds for deep neural networks.

Catoni's Bound The following bound was introduced by Catoni (2007) and it states that for a fixed $\alpha > 1$ and for a loss such that $\ell \in [0,1]$ then

$$\mathop{\mathbb{E}}_{\theta\sim Q}[R\left(f_\theta\right)] \leq \inf_{\lambda>1}\Phi^{-1}_{\lambda/N}\left[\mathop{\mathbb{E}}_{\theta\sim Q}[\hat{R}\left(f_\theta\right)]+\frac{\alpha}{\lambda}\left[\mathbb{KL}\left(Q,P\right)+\log\frac{1}{\delta}+2\log\left(\frac{\log\left(\alpha^2\lambda\right)}{\log\alpha}\right)\right]\right] \quad (3)$$

and

$$\Phi^{-1}_{\gamma}\left(x\right)=\frac{1-e^{\gamma x}}{1-e^{\gamma}}.$$

This bound is quite interesting in that it involves and optimization over the parameter λ which trade-offs how much the bound should focus on the training errors or in the divergence term. This bound was used in Zhou et al. (2019) to derive the first non-vacuous bounds on ImageNet.

Generalization Bound The next bound was introduced in Germain et al. (2009) and it generalizes the Maurer and the Catoni bound. This bound is also valid only for losses such that take values on $[0,1]$. The bound states that for any convex function $H : [0,1]^2 \rightarrow \mathbb{R}$ we have that

$$\begin{aligned}&H\left(\mathop{\mathbb{E}}_{\theta\sim Q}[\hat{R}\left(f_\theta\right)], \mathop{\mathbb{E}}_{\theta\sim Q}[R\left(f_\theta\right)]\right)\\&\leq\frac{1}{N}\left[\mathbb{KL}\left(Q,P\right)+\log\frac{1}{\delta}+\log\mathop{\mathbb{E}}_{\mathcal{D}}\mathop{\mathbb{E}}_{\theta\sim P}[\exp(NH(\hat{R}f_\theta),Rf_\theta)))]\right]\end{aligned}$$

where if $H\left(p,q\right) = \mathrm{KL}\left(p,q\right)$ we recover the Maurer bound and if $H\left(p,q\right) = -\log\left(1-p\left(1-e^{-C}\right)\right)-Cq$ we recover the Catoni bound. This bound raises the question then if there is an optimal choice of H. However, it appears that no such H has been found, or worse, that it exists.

Convexity Optimized Bound The next bound was introduced in Thiemann et al. (2017) and appears to be both tight for low training errors and is convinient to optimize in practice. The bound states that for any $\lambda \in (0,2)$

$$\mathop{\mathbb{E}}_{\theta\sim Q}[R\left(f_\theta\right)] \leq \frac{\mathop{\mathbb{E}}_{\theta\sim Q}[\hat{R}\left(f_\theta\right)]}{1-\lambda/2}+\frac{\mathbb{KL}\left(Q,P\right)+\log\frac{1}{\delta}+\log\left(2\sqrt{N}\right)}{N\lambda\left(1-\lambda/2\right)}.$$

Notice that this is the first bound to achieve a $1/N$ convergence rate when the training error is almost zero (setting $\lambda = 1$). This bound if both convex in the posterior distribution and also convex in the trade-off between empirical performance and the divergence term. This bound has been recently used in Farid and Majumdar (2021) to provide guarantees on meta-learning.

In this section we have introduced and exposed a myriad of the ever so tighter PAC-Bayes bounds that have been developed since 1999. However, there is no single dominant bound as many of them present a different trade-off between the training accuracy and the divergence term that is present on the bound. Yet, as neural networks usually achieve low error rates, then bounds like the "convexity optimized bound" become more attractive. Unfortunately, the development of better mathematical machinery to tighten the bounds is one fragment of the story, as the bound depend on the divergence term $\mathbb{KL}\left(Q,P\right)$ which could be arbitrarily large for deep neural networks. In the next section we explore how researchers have been able to control for the size of the divergence achieving non-vacuous bounds for deep neural networks.

3 Non-vacuous PAC-Bayes Bounds

The naive application of traditional PAC-Bayesian bounds in the context of deep learning results in *vacuous* bounds, i.e., bounds that do not provide new information on generalization. For instance an error bound that is larger than 100% or even equal to 50% for a dataset like MNIST (Deng, 2012) is not informative because we know that the test error will be lower than these values. Hence the importance of obtaining tighter bounds to take advantage of the PAC-Bayesian framework in deep learning.

Dziugaite and Roy (2017) obtained the first non-vacuous generalization bounds for deep stochastic neural networks on the MNIST dataset. The authors use the PAC-Bayes bound from McAllester (1999) and optimize this bound to find a posterior distribution that covers a large volume of low-loss solutions around a local minima obtained using SGD. Their approach extends the work of Langford and Caruana (2001) and Langford (2002) who construct the posterior distribution by perturbing the parameters after training in order to find the largest deviation in each direction that does not increase the training error by more than a certain threshold. This sensitivity analysis is however not applicable to overparameterized neural networks since perturbations in individual parameters has little effect on the training error. In contrast, Dziugaite and Roy (2017) constrain the posterior distribution to be Gaussian which allows them to minimize directly the PAC-Bayes bound using the gradient signal to update the network's parameters and variances. The authors also use a data-dependent Gaussian prior, which is allowed in theory as long as a union bound argument – where the prior over the variance is taken into account – is applied to correct the bound. The authors obtain bounds between 16% and 22% on binary class variant of MNIST, which are non-vacuous but can still be improved upon.

Zhou et al. (2019) rely on the observation that trained neural networks can be compressed to smaller representations without significant loss in the performance to propose non-vacuous PAC-Bayes bounds on the MNIST and ImageNet datasets. They use sparsity-inducing compression schemes (Cheng et al., 2018) to compress the number of bits used to encode the model. Additionally, Zhou et al. (2019) use the well-known idea that deep neural networks are robust to small perturbations (Hinton and Van Camp, 1993; Hochreiter and Schmidhuber, 1997; Langford and Caruana, 2001; Langford, 2002; Keskar et al., 2016; Neyshabur et al., 2017b; Chaudhari et al., 2019). To leverage this observation, they consider a stochastic network similarly to Dziugaite and Roy (2017) and place a Gaussian posterior distribution over the network's parameters. Finally, they evaluate Catoni's PAC-Bayes bound (Catoni, 2007) in equation 3 using the compressed network, a Gaussian posterior and a mixture prior over all possible decoded points of the compression algorithm. They achieve non-vacuous bounds on the MNIST dataset, with a bound that is lower 46%, and on the ImageNet dataset, with a bound that is lower than 96.5%. Although these bounds are non-vacuous in the sense that they are not larger than 100%, they are still non-informative about generalization as the test error for both datasets is expected to be much lower than these error bounds.

Dziugaite et al. (2020) show that for linear PAC-Bayes bounds, i.e., bounds that depend linearly on the KL divergence term between the prior and posterior distributions as in McAllester (2013), choosing the prior to be equal to the posterior distribution can be suboptimal. They demonstrate that a tighter PAC-Bayes bound can be obtained by choosing the prior distribution to be data-dependent. More precisely, the authors prove that choosing the prior distribution to be the conditional expectation of the posterior given a subset of the training data can yield dramatically tighter bounds in some cases. The authors propose a practical approximation of the data-dependent prior by optimizing over a family of Gaussian distributions using a subset of the training data that is then completely discarded and not used in the evaluation of the bound. They then evaluate the bounds for SGD-trained networks and show that they obtain tight bounds on the MNIST and Fashion MNIST (FMNIST) (Xiao et al., 2017) datasets using data-dependent priors. Their experimental results also demonstrate that using almost 90% of the training data to learn the prior can result in a high test performance while still offering guarantees on the expected error. For example, they obtain an average accuracy of 98% on CIFAR-10 (Krizhevsky et al., 2009) with ResNets20 (He et al., 2016) with an error bound of 23% that holds with 95% probability.

Pérez-Ortiz et al. (2021) combine the data-dependent priors approach from Dziugaite et al. (2020) with the PAC-Bayes with Backprop (PBB) approach from Rivasplata et al. (2019) to obtain *state-of-the-art* PAC-Bayes non-vacuous bounds for MNIST and CIFAR-10. PAC-Bayes with Backprop (PBB) is a self-certified learning approach that consists of deriving new training objectives that minimize PAC-Bayes bounds directly. Therefore, training probabilistic neural networks by PBB methods guarantees both high quality predictions and tight PAC-Bayes bounds. Rivasplata et al. (2019) were able to achieve an error bound as low as 2.3% on binary class variant of the MNIST dataset, where digits $0-4$ were mapped to class 0 and digits $5-9$ were mapped to class 1.

Table 1 summarizes the PAC-Bayes non-vacuous bounds that we covered in this section for benchmark datasets in deep learning.

Table 1: Recent non-vacuous bounds obtained on popular image classification datasets in deep learning. Note that the non-vacuous bounds reported by Dziugaite and Roy (2017) and Rivasplata et al. (2019) were obtained on ***binary*** class variant of MNIST.

	Non-vacuous bounds (%)			
Paper	MNIST	FMNIST	CIFAR-10	ImageNet
Dziugaite and Roy (2017)	16.1	✗	✗	✗
Rivasplata et al. (2019)	2.3	✗	✗	✗
Zhou et al. (2019)	< 46	✗	✗	< 96.5
Dziugaite et al. (2020)	11	18	23	✗
Pérez-Ortiz et al. (2021)	1.5	✗	18	✗

4 Recent Developments

We survey some recent developments of PAC-Bayes as applied to various domains.

4.1 Graph Neural Networks

Liao et al. (2021) generalizes the Neyshabur et al. (2017a) bounds to work with graph neural networks. For the case of graph convolutional networks, the bound is similar to the result of Neyshabur et al. (2017a) with the exception of an additional term capturing the maximum node degree (d). Intuitively, this additional terms follows from the fact that feed-forward convolutional networks can be viewed as a graph neural net with zero edges, and thus have no node degrees.

For completeness we include the bound below, with probability $1-\delta$ and any $\gamma > 0$,

$$\mathbb{E}[R(f)] \leq \mathbb{E}[\hat{R}(f)] + \mathcal{O}\left(\sqrt{\frac{B^2 d^{l-1} l^2 h \log(lh) \prod_{i=1}^{l} ||W_i||_2^2 \sum_{i=1}^{l} (\frac{||W_i||_F^2}{||W_i||_2^2} + \log\frac{nl}{\delta})}{\gamma^2 n}}\right), \quad (4)$$

where W_i is the weight matrix of the j-th layer, $B > 0$ is a constant bounding the magnitude of features, $l > 1$ is the number of layers, h is the maximum hidden dimension across all layers, and $d-1$ is the maximum node degree.

4.2 Binary Neural Networks

Letarte et al. (2019) uses PAC-Bayes bound to give a theoretically-driven way of training binary-activated neural networks, which are less computational and memory heavy as compared to the currently mostly popular real-value activated neural networks.

Interesting, they are able to derive a bound to directly optimize binary neural networks with non-vacuous guarantees, despite the fact that binary neural nets are non-differentiable. The bound is based on the results of Catoni (2007). Their modified bound is,

$$\mathbb{E}[R(f)] \leq \inf_{C>0} \left\{ \frac{1}{1-e^{-C}} \left(1 - \exp\left(\mathbb{E}_Q[\hat{R}(f)] - \frac{1}{n}[\mathbb{KL}(Q,P) + \ln\frac{2\sqrt{n}}{\delta}] \right) \right) \right\}, \quad (5)$$

with constant C.

4.3 Probabilistic Neural Networks

Pérez-Ortiz et al. (2021) is interested in training probabilistic neural networks directly with PAC-bayes bounds as to achieve solutions with guarantees, introducing a family of "PAC-Bayes with

Backprop" methods, which connects to the "Bayes-by-Backprop" method of Blundell et al. (2015) which optimizes probabilistic neural networks through a KL bound (but without generalization guarantees).

They start with the classic results of Dziugaite and Roy (2017), which follows from a relaxation of the 0-1 loss with the bounded cross-entropy loss,

$$f_{\text{classic}}(Q) = \hat{L}_S^{x-e}(Q) + \sqrt{\frac{\mathbb{KL}(Q,P) + \log(\frac{2\sqrt{n}}{\delta})}{2n}}\,, \tag{6}$$

where $\hat{L}_S^{x-e}(Q) = \frac{1}{n}\sum_{i=1}^{n} \hat{R}(f(X_i), Y_i)$ is a bounded empirical cross entropy loss, with neural network function f. The authors go on to introduce two new objectives which achieves tighter bounds,

$$f_{\text{quad}}(Q) = \left(\sqrt{\hat{L}_S^{x-e}(Q) + \frac{\mathbb{KL}(Q,P) + \log(\frac{2\sqrt{n}}{\delta})}{2n}} + \sqrt{\frac{\mathbb{KL}(Q,P) + \log(\frac{2\sqrt{n}}{\delta})}{2n}} \right)^2 , \tag{7}$$

$$f_{\text{lambda}}(Q,\lambda) = \frac{\hat{L}_S^{x-e}(Q)}{1-\frac{\lambda}{2}} + \frac{\mathbb{KL}(Q,P) + \log(\frac{2\sqrt{n}}{\delta})}{n\lambda(1-\frac{\lambda}{2})}\,. \tag{8}$$

Generally, neural networks trained with this method is "self-certified" in that they can be trained on the full training dataset, while giving a bound for generalization errors on unseen data from the same distribution.

4.4 Reinforcement Learning

We now turn to a brief survey the application of PAC-Bayes analysis in the reinforcement learning (RL) setting. While PAC analysis is not uncommon in RL (Strehl et al., 2009), PAC-*bayes* analysis is more sparsely found. Notably, Milani Fard and Pineau (2010) provides the first PAC-bayes analysis for the RL setting, for the case of discrete, finite state and action spaces.

Concretely, given a finite set of states $\mathcal{S}$ and actions $\mathcal{A}$, the object of interest for this set of analysis is the *value function* $Q^{\pi^*} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. This is the value function (denoted by Q) of the best policy (denoted π^*), meaning it has the highest value (sum of future rewards) for all states $s \in \mathcal{S}$. If one can learn Q^{π^*}, then one can act optimally (i.e. in a way that maximizes total reward) by simply taking the greedy action: $\arg\max_a Q^{\pi^*}(s,a), \forall s \in \mathcal{S}$. However, the challenge of RL arises from the fact that one cannot learn Q^{π} by minimizing empirical risk from i.i.d. sampled data (as is the case for supervised learning). We do not focus on methods for solving Q^{π^*} here (for a good reference see Sutton and Barto (2018)), but we can still reason about PAC-bayes bounds in the RL setting by either lower-bounding the true optimal value function Q^{π^*}, or upper bounding the error of an approximate value function to Q^{π^*}.

Model based, discrete finite states One way to solve for Q^{π^*} is via "model-based" RL. Here, a transition model of the environment is learned, then Q^{π^*} can be solved through dynamic programming. To avoid abuse of notation, consider the prior ρ_0 and posterior ρ, here defined over the space of transition models. Milani Fard and Pineau (2010); Milani Fard (2014) lower-bounds Q^{π^*}, as a function of the error in learning such a transition model T, for discrete state and action spaces,

$$\mathbb{E}_{T\sim\rho}[Q^{\pi_T^*}(s,a)] \geq \mathbb{E}_{T\sim\rho}[\hat{Q}^{\pi_T^*}(s,a)] - \sqrt{\frac{\mathbb{KL}(\rho||\rho_0) - \ln\delta + |\mathcal{S}|\ln 2 + \ln|\mathcal{S}| + \ln n_{\text{min}}}{\frac{1}{2}(n_{\text{min}}-1)k^2}}\,, \tag{9}$$

with $|\mathcal{S}|$ denoting the size of the (finite) state space, and constants $n_{\text{min}} = \min_{s,a} n_{s,a}$ denoting the number of samples collected for each state-action pairs (note this space has size $|\mathcal{S}| \times |\mathcal{A}|$), and $k \geq (1-\gamma)^2/\gamma$ and $\gamma \in [0,1)$ being RL-specific constants that bounds the value function error as a function of the transition model error.

Model free, discrete finite states Another method for obtaining Q^{π^*} is to solve for it directly by minimizing $||Q - Q^{\pi^*}||_\infty$ for all states and actions. While this cannot be done directly (as one does not have samples from Q^{π^*}), this can be done by minimizing a surrogate "temporal difference" error l_{TD}.[†] In the parlance of supervised learning, one can treat $||Q - Q^{\pi^*}||_\infty$ as the true risk, and l_{TD} as a (surrogate) empirical risk. Similarly, one can derive a PAC-bayes bound by defining the posterior distribution ρ over value functions Q,

$$\mathbb{E}_{Q\sim\rho}[||Q - Q^{\pi^*}||_\infty] \leq \frac{\mathbb{E}_{Q\sim\rho}[l_{\text{TD}}]}{1-\gamma} + \sqrt{\frac{\mathbb{KL}(\rho,\rho_0) - \ln\delta + \ln|\mathcal{S}| + \ln|\mathcal{A}| + \ln(2n_{\text{min}})}{2(n_{\text{min}}-1)C}}\,. \quad (10)$$

Notably, the above bounds are not quite "generalization" bounds in the supervised learning setting. Instead, they bound the quality of behaviour of the agent (in terms of the value function Q^{π^*} which estimate how much total reward the agent will get) as a function of the number of interactions from the environment. Milani Fard and Pineau (2010) also optimizes the above bounds directly to take into account better prior information to find a good policy.

Model free, continuous states One can extend the above analysis to the space of continuous states—therefore continuous value functions, as was done in Milani Fard et al. (2012). However, Milani Fard et al. (2012) only shows this in the case of *policy evaluation*—finding Q for some given policy π—rather than the previous result of finding Q for the*best* policy π^*. This bound is,

$$\mathbb{E}_{Q\sim\rho}[||Q - Q^{\pi}||] \leq \frac{1}{(1-\gamma)^2}\left(\mathbb{E}_{Q\sim\rho}[J^{\pi}_{\text{TD}}(Q)] + \sqrt{\frac{\mathbb{KL}(\rho,\rho_0) + \ln\frac{n}{\delta}}{(n-1)/C}} - \mathbb{E}_{Q\sim\rho}[\Lambda^{\pi}_{TD}(Q)]\right)\,, \quad (11)$$

where $J^{\pi}_{\text{TD}}(Q)$ is the empirical temporal difference loss under policy π (akin to "empirical risk"), and $C = 2\tau V^4_{\text{max}}$ are constants, and $\Lambda^{\pi}_{TD}(V)$ is a term relating to the variance of each state.

This is useful for when we have informative prior about the value of a policy and wish to quickly re-evaluate its value in a new environment. All in all, much is left to be developed in applying PAC-Bayes analysis to the RL setting, although the lack of work here likely results from the difficulty in bounding the non i.i.d. RL objective.

5 Computational aspects of PAC-Bayes

Advancements in modern machine learning has lead to developments in provably efficient learning algorithms. While broader questions of whether deep neural networks are provably efficient remain unanswered, the study of shallow networks with common activation functions has shown reasonable success, when permitting for various assumptions of the weight structure of the network and distribution of the training data. While this framework limits each study to a limited set of learners and various constraints, recent success has resulted in the development of a handful of provably efficient learning algorithms in a variety of cases. In this section, we highlight a few examples of recent PAC-learnable algorithms, particularly those applicable to neural networks.

PAC Learning In proving efficient PAC learning algorithms, we typically define a target function $f \in \mathcal{F}$ and a data distribution $D \in \mathbb{R}^d$ noisy data generator (called a *noisy example oracle*, $\text{EX}^{\text{noise}}(f, \mathcal{F})$ that returns data pairs $(\mathbf{x}, y)$ such that $\mathbf{x} \sim \mathcal{D}$ and $y = f(\mathbf{x}) + \xi$, where ξ is a zero-mean subgaussian noise variable with standard deviation σ. For a given number of samples, and confidence δ, the goal of the learner is to produce a hypothesis $\hat{f}$ that is ϵ-close to f so that:

$$P\left[\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\ell(f(\mathbf{x}), \hat{f}(\mathbf{x}))^2\right] \leq \epsilon^2\left(\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[f^2(\mathbf{x})\right] + \sigma^2\right)\right] \geq 1 - \delta \quad (12)$$

This might look familiar, as a non-Bayesian form of 1. While much research has shown success in recognizing efficient learning algorithms for shallow/simple neural networks, fewer have specifically proven PAC learnability. Learnable algorithms have focused on *parameter estimation*, the task of recovering coefficients and the corresponding weight matrix of the data generation distribution,

[†] See appendix for a more precise definition of l_{TD}

which contrasts with PAC learning, which assumes no structure of the weight process, as the learned solution can be ϵ-close while having an entirely different weight structure.

5.1 Polynomial-time Learning for One-Hidden-Layer ReLU Networks

Diakonikolas et al. (2020) recently derived a PAC learning algorithm that's solvable in polynomial time, the first algorithm for the particular class of algorithms explored in the paper.

One-hidden-layer ReLU networks To ensure a class of networks without strong restrictions on the weights, this study focused on a relatively simple architecture of an input, hidden layer with k units, and output, using ReLU activations, denoted as $\mathcal{C}_k$. A particular instance of this class $f_{\alpha,\mathbf{W}} \in \mathcal{C}_k$ with a vector of coefficients α and weight matrix $\mathbf{W}$ would then be defined as:

$$f_{\alpha,\mathbf{W}} = \sum_{i}^{k} \alpha_i \phi(\langle \mathbf{w}^{(i)}, \mathbf{x} \rangle) \tag{13}$$

where $\phi(t) = \max\{0, t\}$ (a ReLU activation). Furthermore, they constrain this study to the subset of $\mathcal{C}_k$ such that all coefficients $\alpha_i > 0$, the positive coefficient class of single-hidden layer ReLU networks $\mathcal{C}_k^+ \subset \mathcal{C}_k$.

Techniques The crux of the approach by Diakonikolas et al. (2020) is reducing the problem from being d-dimensional to being k-dimensional by the following steps. Knowing our target function f will take the form $f = \sum_i^k \alpha_i \phi(\langle \mathbf{w}^{(i)}, \mathbf{x} \rangle)$, we know that there exists a subspace $\mathcal{V}$ spanned by the vectors of the weight matrix $\mathbf{w}^{(i)}$. If we learn this subspace, we can solve our target function in k dimensions, as it can be written as $f(\mathbf{x}) = f(\text{proj}_{\mathcal{V}}(x))$.

To build our approximation of this unknown subspace, $\mathcal{V}'$, the authors note that we can use the Chow-2 parameters[‡] of the target function, $f(x)(xx^T - \mathbf{I})$, showing that the moments of f are positive in directions along $\mathcal{V}$ and 0 in orthogonal directions. This notion can be approximated by noting that weight vectors $\mathbf{w}^{(i)}$ with large corresponding coefficients α_i will have large second moments, and those with small α_i will likely be orthogonal to $\mathcal{V}$. By building $\mathcal{V}'$ using this approximation of the second moment matrix by learning $\mathbf{W}$, it can be shown that $f(\mathbf{x})$ is provably close to $\hat{f}(\text{proj}_{\mathcal{V}'}(\mathbf{x}))$.

Statistical Query Lower Bound To formally define a lower bound for PAC learning the single-hidden-layer ReLU class, the authors define a correlational Statistical Query (SQ) model under the assumption of Gaussian-distributed data. This model has access to an oracle that can take a query function $q : \mathbb{R}^d \rightarrow [-1, 1]$ and accuracy parameter $\tau > 0$ and learns an estimate of $\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[f(\mathbf{x})g(\mathbf{x})]$ within some accuracy. This general result fits for the algorithm chosen, and also guarantees that error of order $\epsilon = \Omega(1)$ in polynomial time, dependent on the number of hidden units k. This bound is established for the specific form shown in 13, and provides the result for a solution in polynomial time when $k \leq \tilde{O}(\sqrt{\log d})$.

5.2 Further Examples of PAC Learning Algorithms for Neural Nets

PAC Learning of Single-Layer Neural Nets Vempala and Wilmes (2018) identify a PAC learning algorithm for the class of single-layer neural networks with any nonlinearity ϕ and a single output for inputs drawn from the uniform distribution on the sphere $S^{n-1} \subset \mathbb{R}^n$. By loosening the restrictions on the nonlinearity, they show, by a similar SQ framework to show complexity in exponential time, even when constraining to a single ReLU network, as done in Diakonikolas et al. (2020).

PAC Learning for 2-Layer Networks Goel and Klivans (2018) extends the work done by Vempala and Wilmes (2018) by developing a learning algorithm *Alphatron* that is provably efficient for two-layer neural networks in polynomial time. It similarly examines data drawn from the unit ball, but of any distribution over the ball. This is the provably PAC learning algorithm that is able to handle a multi-layer model, highlighting the difficulty of extending provably efficient algorithms for NNs of any meaningful depth.

[‡] Degree-2 Fourier Coefficients

References

Biggs, F. and Guedj, B. (2022). On Margins and Derandomisation in PAC-Bayes. *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR. 7

Catoni, O. (2007). PAC-Bayesian Supervised Classification: the Thermodynamics of Statistical Learning. *Institute of Mathematical Statistics Lecture Notes*. 2, 4, 5, 6

Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J., Sagun, L., and Zecchina, R. (2019). Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018. 5

Cheng, Y., Wang, D., Zhou, P., and Zhang, T. (2018). Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1):126–136. 5

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee. 2

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142. 5

Diakonikolas, I., Kane, D. M., Kontonis, V., and Zarifis, N. (2020). Algorithms and SQ Lower Bounds for PAC Learning One-Hidden-Layer ReLU Networks. *33rd Annual Conference on Learning Theory (COLT)*. 2, 9

Dziugaite, G. K., Hsu, K., Gharbieh, W., and Roy, D. M. (2020). On the role of data in pac-bayes bounds. *arXiv preprint arXiv:2006.10929*. 5, 6

Dziugaite, G. K. and Roy, D. M. (2017). Computing Nonvacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data. *Conference on Uncertainty in Artificial Intelligence (UAI)*. 2, 4, 5, 6, 7

Farid, A. and Majumdar, A. (2021). Generalization Bounds for Meta-Learning via PAC-Bayes and Uniform Stability. *35th Conference on Neural Information Processing Systems (NeurIPS)*. 4

Germain, P., Lacasse, A., Laviolette, F., and Marchand, M. (2009). PAC-Bayesian Learning of Linear Classifiers. *Proceedings of the 26th9 International Conference on Machine Learning (ICML)*. 4

Goel, S. and Klivans, A. R. (2018). Learning depth-three neural networks in polynomial time. *CoRR*, abs/1709.06010. 2, 9

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. 5

Hinton, G. E. and Van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13. 5

Hochreiter, S. and Schmidhuber, J. (1997). Flat minima. *Neural computation*, 9(1):1–42. 5

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*. 5

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. 5

Langford, J. (2002). *Quantitatively tight sample complexity bounds*. PhD thesis, Carnegie Mellon University. 5

Langford, J. and Caruana, R. (2001). (not) bounding the true error. *Advances in Neural Information Processing Systems*, 14. 5

Langford, J. and Seeger, M. (2001). Bounds for Averaging Classifiers. *Tech. rep CMU-CS-01-102, Carnegie Mellon University*. 2, 3

Letarte, G., Germain, P., Guedj, B., and Laviolette, F. (2019). Dichotomize and Generalize: PAC-Bayesian Binary Activated Deep Neural Networks. *33rd Conference on Neural Information Processing Systems (NeurIPS)*. 2, 6

Liao, R., Urtasun, R., and Zemel, R. (2021). A PAC-Bayesian Approach to Generalization Bounds for Graph Neural Networks. *9th International Conference on Learning Representations (ICLR)*. 2, 6

Maurer, A. (2004). A Note on the PAC Bayesian Theorem. *Preprint arXiv 041099v1*. 3

McAllester, D. (2013). A pac-bayesian tutorial with a dropout bound. *arXiv preprint arXiv:1307.2118*. 5

McAllester, D. A. (1999). PAC-Bayesian Model Averaging. *Proceedings of the 12th Annual Conference on Learning Theory (COLT)*, pages 164–170. 1, 3, 5

Milani Fard, M. (2014). *Regularized reinforcement learning with performance guarantees*. PhD thesis, McGill University. 7

Milani Fard, M. and Pineau, J. (2010). Pac-bayesian model selection for reinforcement learning. *Advances in Neural Information Processing Systems*, 23. 7, 8, 12

Milani Fard, M., Pineau, J., and Szepesvári, C. (2012). Pac-bayesian policy evaluation for reinforcement learning. *arXiv preprint arXiv:1202.3717*. 8

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. (2017a). Exploring Generalization in Deep Learning. *31st Conference on Neural Information Processing Systems (NeurIPS)*. 2, 6

Neyshabur, B., Bhojanapalli, S., and Srebro, N. (2017b). A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1707.09564*. 5

Pérez-Ortiz, M., Rivasplata, O., Shawe-Taylor, J., and Szepersvári, C. (2021). Tighter Risk Certificates for Neural Networks. *Journal of Machine Learning Research 22 (2021) 1-40*. 2, 5, 6

Rivasplata, O., Tankasali, V. M., and Szepesvári, C. (2019). Pac-bayes with backprop. *arXiv preprint arXiv:1908.07380*. 5, 6

Shawe-Taylor, J. and Williamson, R. C. (1997). A PAC Analysis of a Bayesian Estimator. *Proceedings of the 10th Annual Conference on Learning Theory (COLT)*. 1, 3

Strehl, A. L., Li, L., and Littman, M. L. (2009). Reinforcement learning in finite mdps: Pac analysis. *Journal of Machine Learning Research*, 10(11). 7

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press. 7

Thiemann, N., Igel, C., Wintenberger, O., and Seldin, Y. (2017). A Strongly Quasiconvex PAC-Bayesian Bound. *28th Annual Conference on Learning Theory (COLT)*. 4

Vempala, S. S. and Wilmes, J. (2018). Polynomial convergence of gradient descent for training one-hidden-layer neural networks. *CoRR*, abs/1805.02677. 2, 9

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*. 5

Zhou, W., Veitch, V., Austern, M., Adams, R. P., and Orbanz, P. (2019). Non-Vacuous Generalization Bounds at the ImageNet Scale: A PAC-Bayesian Compression Approach. *7th International Conference on Learning Representations (ICLR)*. 1, 2, 4, 5, 6

A Appendix: Reinforcement Learning

We follow the notation of Milani Fard and Pineau (2010) and define the empirical Bellman optimality operator, $\hat{\mathcal{B}}$. The operator acts to transform the the current value function $Q(s, a)$ in the following way,

$$\hat{\mathcal{B}}Q(s,a) = \frac{1}{n_{s,a}} \sum_{(s,a,r,s')\in U} \left(r + \gamma \max_{a'} Q(s', a') \right) , \tag{14}$$

where (s, a, r, s') is a single "experience sample" from some sampling distribution U (e.g. by acting in the environment, or past experiences from the environment) containing current state, current action, immediate reward, and next-step state, respectively. In practice, $\hat{\mathcal{B}}Q(s, a)$ can be used as a learning target for the current value function $Q(s, a)$. Learning is done by minimizing some distance between the two functions, e.g. here defined as a norm over all state-action pairs,

$$l_{\text{TD}} = ||Q - \hat{\mathcal{B}}Q||_\infty . \tag{15}$$