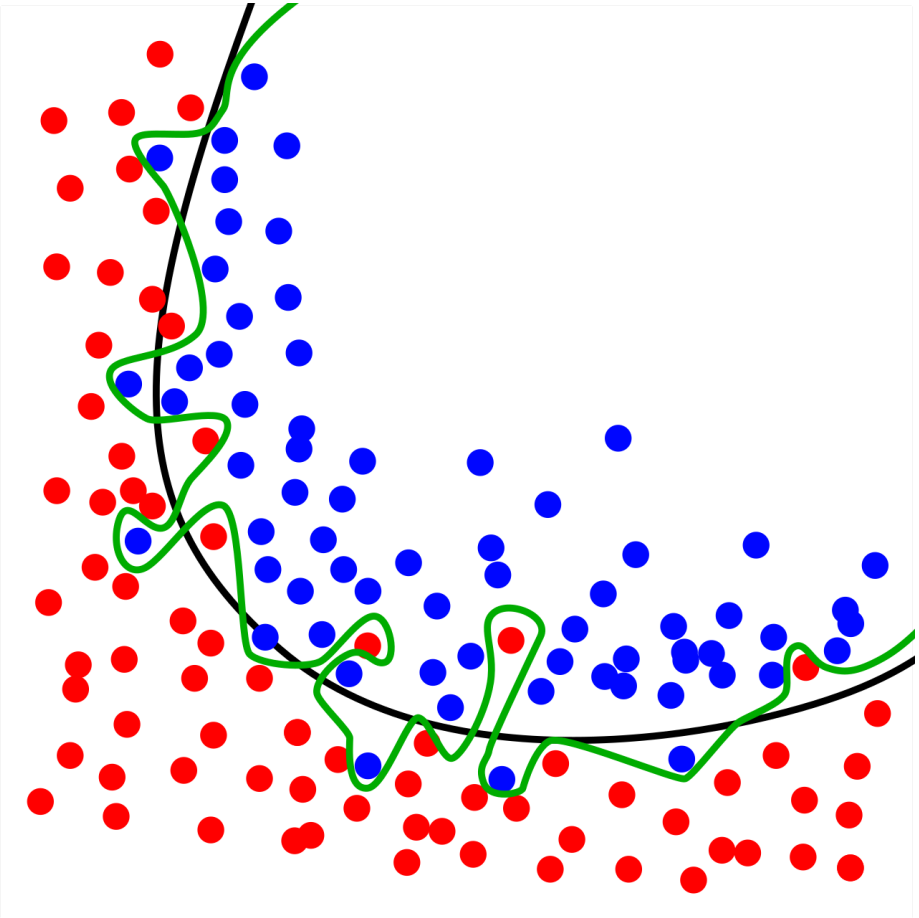
On generalisation and learning: Generalisation bounds for deep neural networks

Benjamin Guedj
https://bguedj.github.io
@bguedj

JdS
MALIA, June 9, 2021

Learning is to be able to generalise



[Credits: Wikipedia]

From examples, what can a system learn about the underlying phenomenon?

Memorising the already seen data is usually bad $\longrightarrow$ overfitting

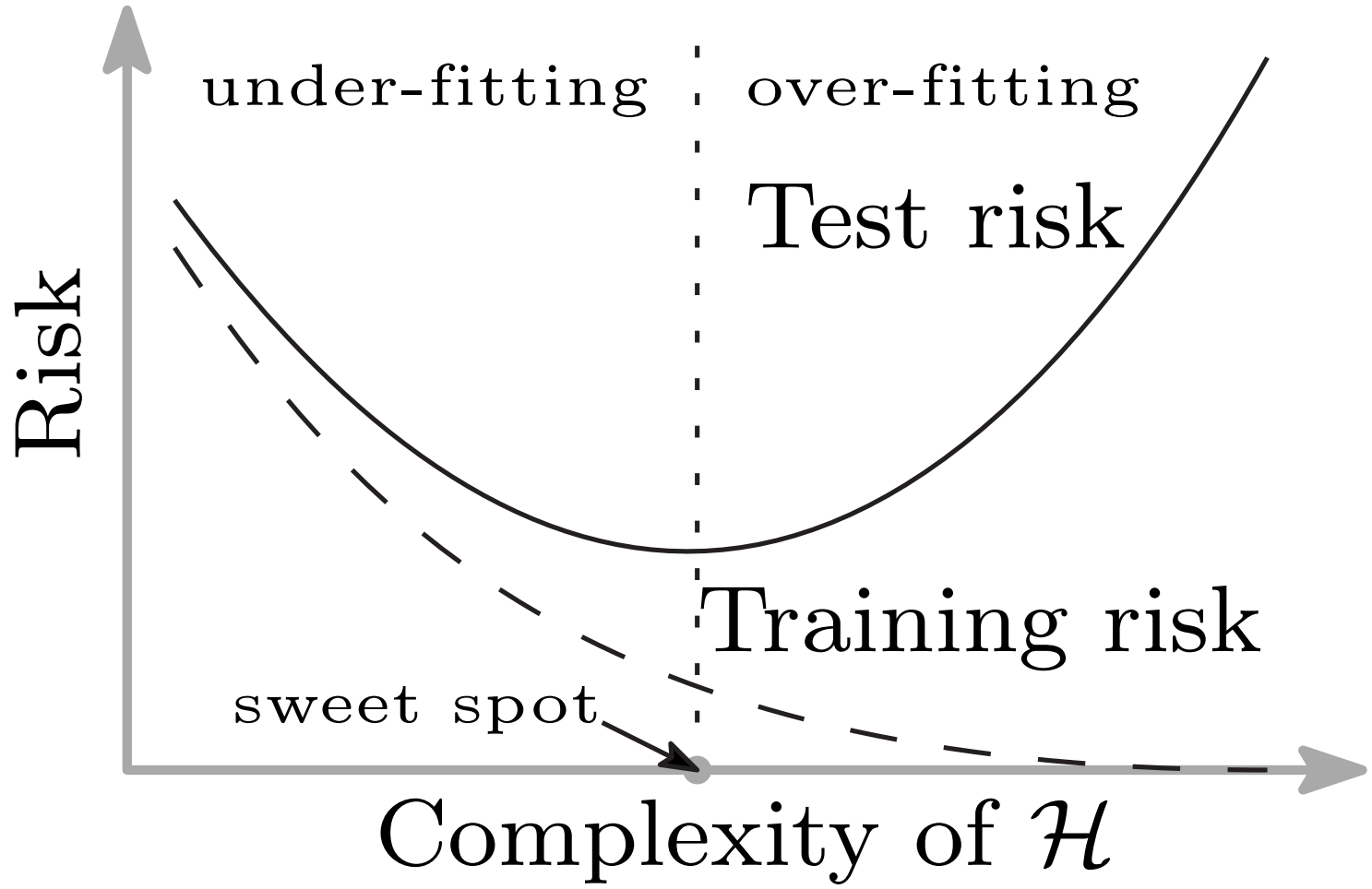
Generalisation is the ability to 'perform' well on unseen data.

Is deep learning breaking statistical learning theory?

Neural networks architectures trained on massive datasets achieve zero training error which does not bode well for their performance: this strongly suggests overfitting...

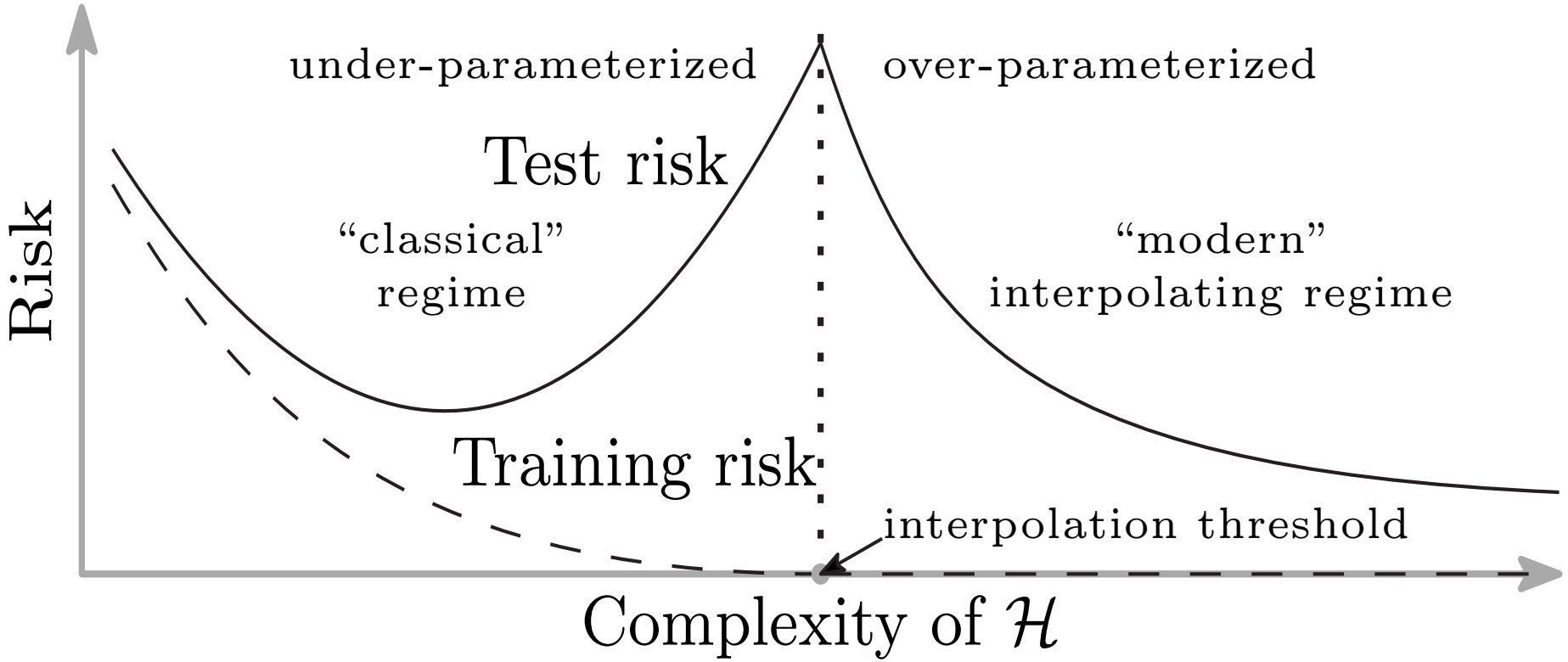
... yet they also achieve remarkably low errors on test sets!

A famous plot...



Belkin et al. (2019)

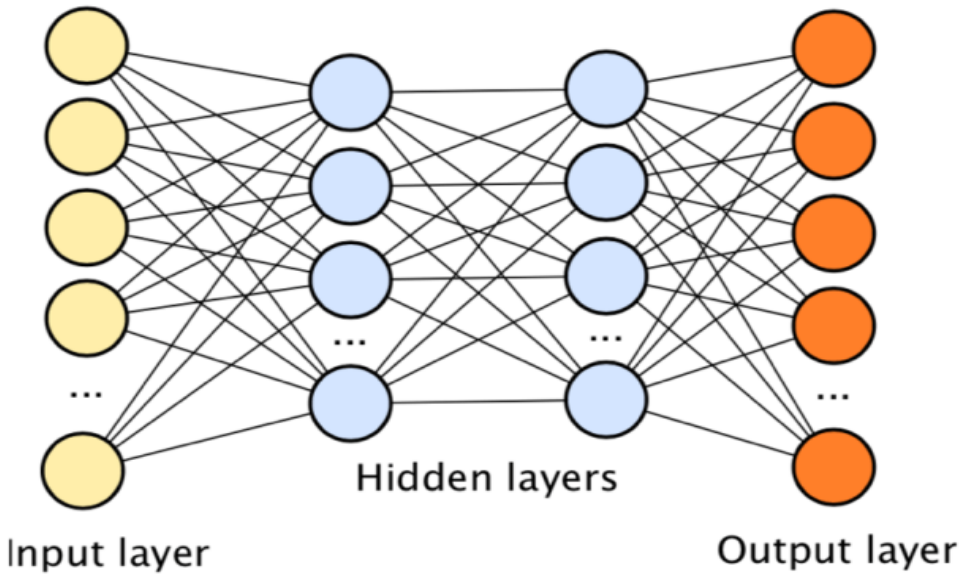
... which might just be half of the picture



Belkin et al. (2019)

A tale of two learners

On our left: a deep neural network



Typically identifies a specific item (say, a horse) in an image with accuracy $>$ 99%.
Training samples: millions of annotated images of horses – GPU-expensive training.

A tale of two learners

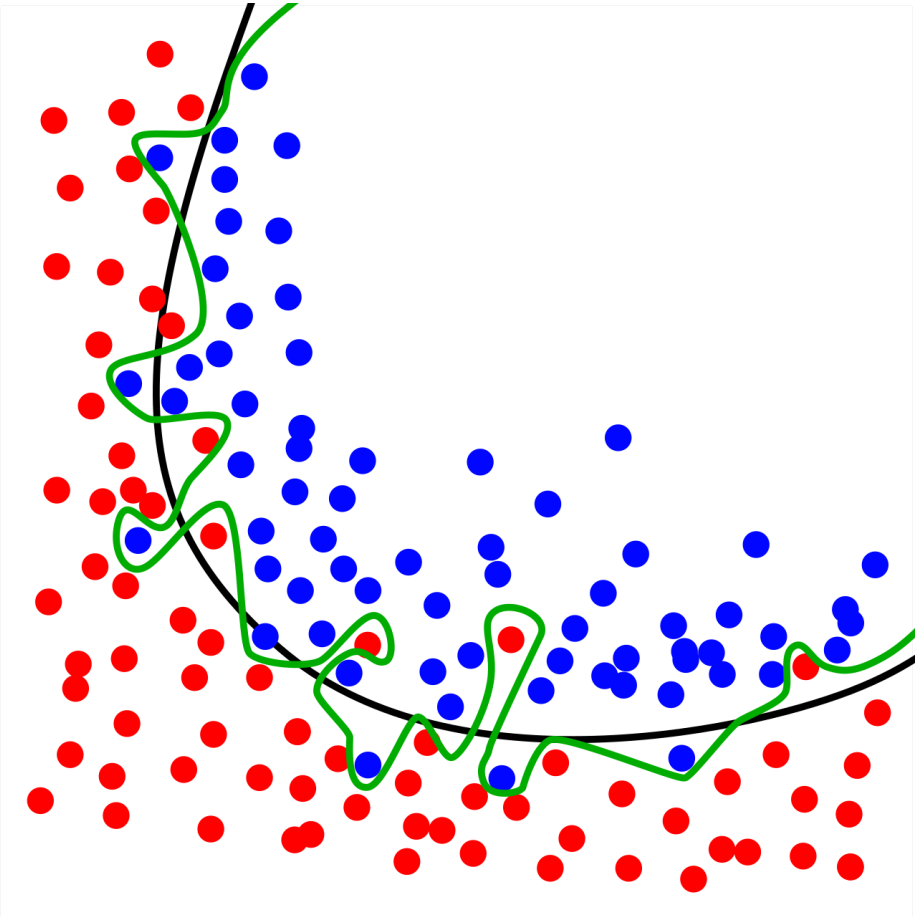
On our right: the next generation

Identify horses with 100% accuracy. Also very good at transferring to *e.g.* zebras

Training samples: a handful of children books, bedtime stories and (poorly executed) drawings.

Also expensive training.

Learning is to be able to generalise...



... but not from scratch! Tackling each learning task as a fresh draw unlikely to be efficient – must not be blind to context.

Need to incorporate structure / semantic information / implicit representations of the ”sensible” world.

Should lead to better algorithms design (more ”intelligent”, frugal / resources-efficient, etc.)

ICML 2019 Tutorial

A Primer on PAC-Bayesian Learning

https://bguedj.github.io/icml2019/index.html

Generalisation

Loss function $\ell(h(X), Y)$ to measure the discrepancy between a predicted output $h(X)$ and the true output Y.

Empirical risk: $R_{\text{in}}(h) = \frac{1}{m} \sum_{i=1}^{m} \ell(h(X_i), Y_i)$
(in-sample)

Theoretical risk: $R_{\text{out}}(h) = \mathbb{E}\left[\ell(h(X), Y)\right]$
(out-of-sample)

If predictor h does well on the in-sample (X, Y) pairs...
...will it still do well on out-of-sample pairs?

Generalisation gap: $\Delta(h) = R_{\text{out}}(h) - R_{\text{in}}(h)$

Upper bounds: with high probability $\Delta(h) \leqslant \epsilon(m, \delta)$

- $R_{\text{out}}(h) \leqslant R_{\text{in}}(h) + \epsilon(m, \delta)$

Flavours:

- distribution-free
- distribution-dependent
- algorithm-free
- algorithm-dependent

The PAC (Probably Approximately Correct) framework

In a nutshell: with high probability, the generalisation error of an hypothesis h is at most something we can control and even compute. For any $\delta > 0$,

$$\mathbb{P}\left[R_{\text{out}}(h) \leqslant R_{\text{in}}(h) + \epsilon(m, \delta)\right] \geqslant 1 - \delta.$$

Think of $\epsilon(m, \delta)$ as $\text{Complexity} \times \frac{\log \frac{1}{\delta}}{\sqrt{m}}$.

This is about high confidence statements on the tail of the distribution of test errors (compare to a statistical test at level $1 - \delta$).

PAC-Bayes is about PAC generalisation bounds for ***distributions over hypotheses***.

Why should I care about generalisation?

Generalisation bounds are a safety check: they give a theoretical guarantee on the performance of a learning algorithm on any unseen data.

Generalisation bounds:

- provide a computable control on the error on any unseen data with prespecified confidence
- explain why some specific learning algorithms actually work
- and even lead to designing new algorithms which scale to more complex settings

A classical PAC-Bayesian bound

Pre-history: PAC analysis of Bayesian estimators

Shawe-Taylor and Williamson (1997)

Birth: PAC-Bayesian bound

McAllester (1998, 1999)

Prototypical bound

For any prior P, any $\delta \in (0, 1]$, we have

$$\mathbb{P}^m\left(\forall\, Q \text{ on } \mathcal{H}\colon\ R_{\text{out}}(Q) \leqslant R_{\text{in}}(Q) + \sqrt{\frac{\text{KL}(Q\|P) + \ln \frac{2\sqrt{m}}{\delta}}{2m}}\right) \geqslant 1 - \delta\,,$$

with $R_{\text{in}}(Q) \equiv \int_{\mathcal{H}} R_{\text{in}}(h)\, dQ(h)$, $R_{\text{out}}(Q) \equiv \int_{\mathcal{H}} R_{\text{out}}(h)\, dQ(h)$,
$\text{KL}(Q\|P) = \mathop{\mathbf{E}}_{h\sim Q} \ln \frac{Q(h)}{P(h)}$.

Want to know more? *Guedj (2019)* + `https://bguedj.github.io/talks/`

PAC-Bayes-driven learning algorithms

With an arbitrarily high probability and for any posterior distribution Q,

$$\text{Error on unseen data} \leqslant \text{Error on sample} + \text{complexity term}$$

$$R_{\text{out}}(Q) \leqslant R_{\text{in}}(Q) + F(Q, \cdot)$$

This defines a principled strategy to obtain new learning algorithms:

$$h \sim Q^\star$$

$$Q^\star \in \arg\inf_{Q \ll P} \left\{ R_{\text{in}}(Q) + F(Q, \cdot) \right\}$$

(optimisation problem which can be solved or approximated by [stochastic] gradient descent-flavoured methods, Monte Carlo Markov Chain, variational inference...)

SVMs, KL-regularized Adaboost, exponential weights are all minimisers of PAC-Bayes bounds.

Generalisation guarantees for Binary activated DNNs

Letarte, Germain, Guedj and Laviolette (2019). Dichotomize and generalize: PAC-Bayesian binary activated deep neural networks, NeurIPS 2019.

Standard Neural Networks

Classification setting:

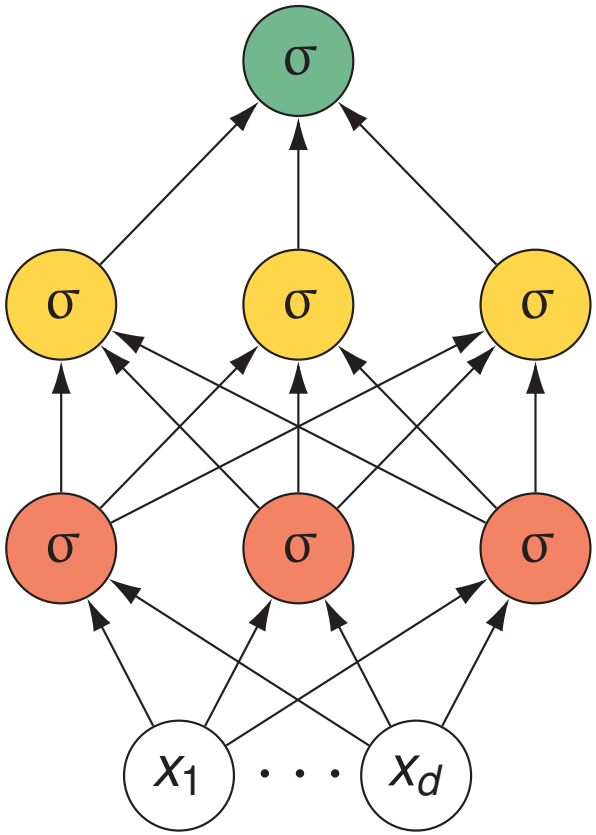
- $\mathbf{x} \in \mathbb{R}^{d_0}$
- $y \in \{-1, 1\}$

Architecture:

- *L fully connected* layers
- d_k denotes the number of neurons of the k^{th} layer
- $\sigma : \mathbb{R} \to \mathbb{R}$ is the *activation function*

Parameters:

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



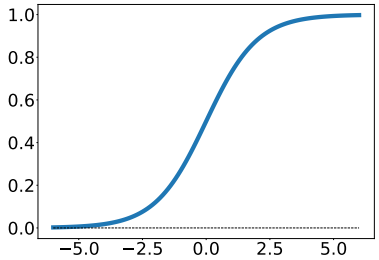
Prediction

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

PAC-Bayesian bounds for Stochastic NN

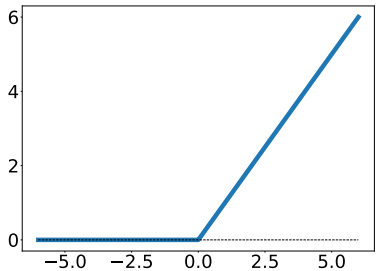
Langford and Caruana (2001)

- Shallow networks ($L = 2$)
- Sigmoid activation functions



Dziugaite and Roy (2017)

- Deep networks ($L > 2$)
- ReLU activation functions



Idea: Bound the expected loss of the network under a Gaussian perturbation of the weights

Empirical loss: $\underset{\theta' \sim \mathcal{N}(\theta, \Sigma)}{\mathbf{E}} R_{\text{in}}(f_{\theta'})$ $\longrightarrow$ estimated by sampling

Complexity term: $\mathrm{KL}(\mathcal{N}(\theta, \Sigma) \| \mathcal{N}(\theta_0, \Sigma_0))$ $\longrightarrow$ closed form

Binary Activated Neural Networks

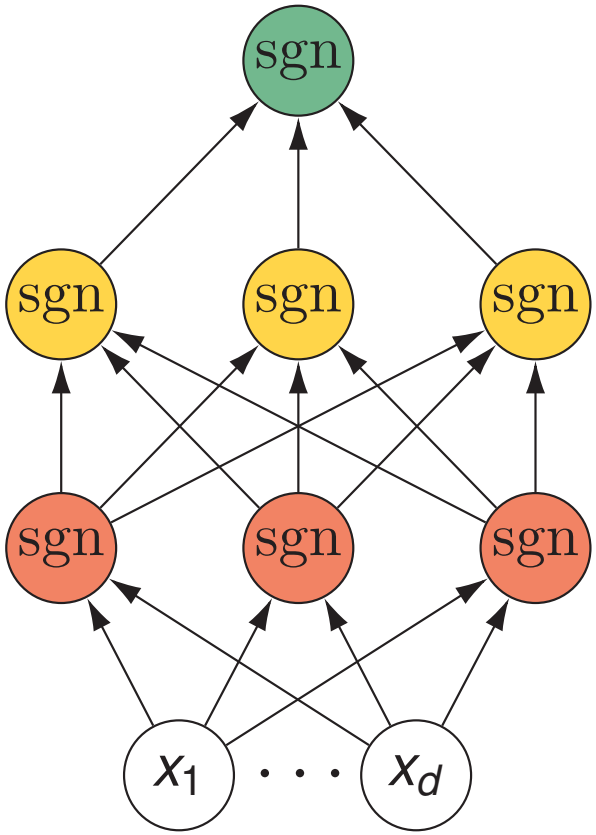
- $\mathbf{x} \in \mathbb{R}^{d_0}$
- $y \in \{-1, 1\}$

Architecture:

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

Parameters:

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



Prediction

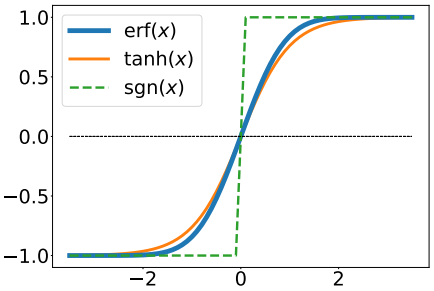
$$f_\theta(\mathbf{x}) = \operatorname{sgn}\left(\mathbf{w}_L \operatorname{sgn}\left(\mathbf{W}_{L-1} \operatorname{sgn}\left(\ldots \operatorname{sgn}\left(\mathbf{W}_1 \mathbf{x}\right)\right)\right)\right),$$

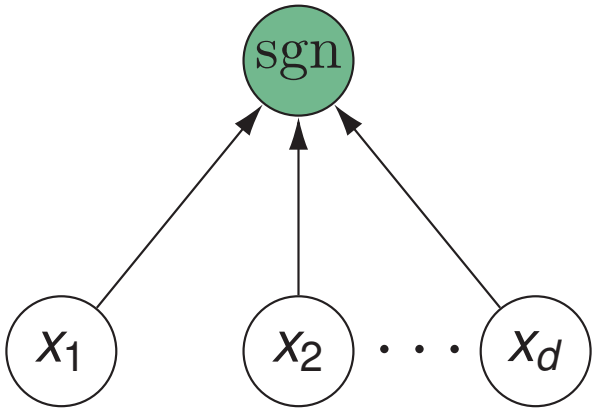
Building block: one layer (aka linear predictor)

Letarte et al. (2019)

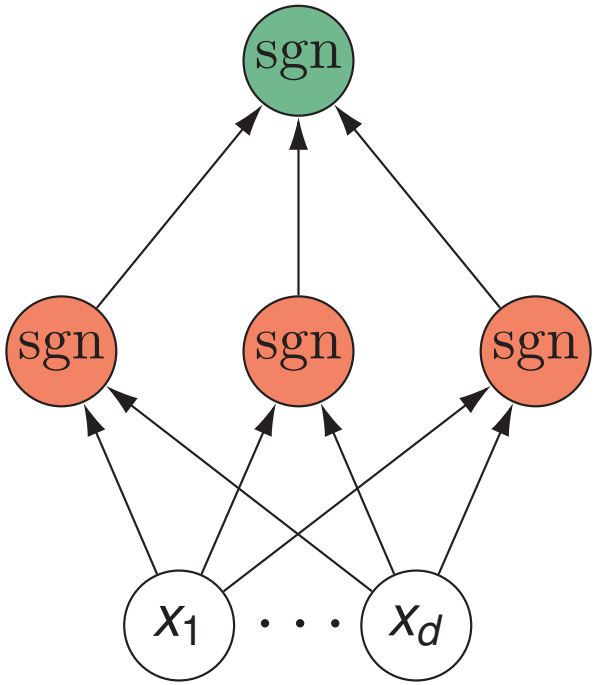
Model $f_{\mathbf{w}}(\mathbf{x}) \stackrel{\text{def}}{=} \operatorname{sgn}(\mathbf{w} \cdot \mathbf{x})$, with $\mathbf{w} \in \mathbb{R}^d$.

- Linear classifiers $\mathcal{F}_d \stackrel{\text{def}}{=} \{f_{\mathbf{v}} | \mathbf{v} \in \mathbb{R}^d\}$
- Predictor $F_{\mathbf{w}}(\mathbf{x}) \stackrel{\text{def}}{=} \mathbf{E}_{\mathbf{v} \sim Q_{\mathbf{w}}} f_{\mathbf{v}}(\mathbf{x}) = \operatorname{erf}\left(\frac{\mathbf{w} \cdot \mathbf{x}}{\sqrt{d}\|\mathbf{x}\|}\right)$
- Sampling + closed form of the KL + a few other tricks + extension to an arbitrary number of layers





Two Layers (shallow network)



Two Layers (shallow network)

Posterior $Q_\theta = \mathcal{N}(\theta, I_D)$, over the family of all networks $\mathcal{F}_D = \{f_{\tilde{\theta}} \mid \tilde{\theta} \in \mathbb{R}^D\}$, where

$$f_\theta(\mathbf{x}) = \operatorname{sgn}\left(\mathbf{w}_2 \cdot \operatorname{sgn}(\mathbf{W}_1 \mathbf{x})\right).$$

$$\begin{aligned}
F_\theta(\mathbf{x}) &= \mathop{\mathbf{E}}_{\tilde{\theta} \sim Q_\theta} f_{\tilde{\theta}(\mathbf{x})} \\
&= \int_{\mathbb{R}^{d_1 \times d_0}} Q_1(\mathbf{V}_1) \int_{\mathbb{R}^{d_1}} Q_2(\mathbf{v}_2) \operatorname{sgn}(\mathbf{v}_2 \cdot \operatorname{sgn}(\mathbf{V}_1 \mathbf{x})) d\mathbf{v}_2 d\mathbf{V}_1 \\
&= \int_{\mathbb{R}^{d_1 \times d_0}} Q_1(\mathbf{V}_1) \operatorname{erf}\left(\frac{\mathbf{w}_2 \cdot \operatorname{sgn}(\mathbf{V}_1 \mathbf{x})}{\sqrt{2}\|\operatorname{sgn}(\mathbf{V}_1 \mathbf{x})\|}\right) d\mathbf{V}_1 \\
&= \sum_{\mathbf{s} \in \{-1,1\}^{d_1}} \operatorname{erf}\left(\frac{\mathbf{w}_2 \cdot \mathbf{s}}{\sqrt{2 d_1}}\right) \int_{\mathbb{R}^{d_1 \times d_0}} \mathbb{1}[\mathbf{s} = \operatorname{sgn}(\mathbf{V}_1 \mathbf{x})] Q_1(\mathbf{V}_1)\, d\mathbf{V}_1 \\
&= \sum_{\mathbf{s} \in \{-1,1\}^{d_1}} \underbrace{\operatorname{erf}\left(\frac{\mathbf{w}_2 \cdot \mathbf{s}}{\sqrt{2 d_1}}\right)}_{F_{\mathbf{w}_2}(\mathbf{s})} \underbrace{\prod_{i=1}^{d_1} \left[\frac{1}{2} + \frac{s_i}{2} \operatorname{erf}\left(\frac{\mathbf{w}_1^i \cdot \mathbf{x}}{\sqrt{2}\,\|\mathbf{x}\|}\right)\right]}_{\Pr(\mathbf{s} \mid \mathbf{x}, \mathbf{W}_1)}.
\end{aligned}$$

Generalisation bound

Let F_θ denote the network with parameter θ. With probability at least $1-\delta$, for any $\theta \in \mathbb{R}^D$

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

Numerical experiments

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$_{\text{pre}}$					
– pretrain	linear loss (20 epochs)	50%	-	-	random init
– final	PAC-Bayes bound	50%	-	PAC-Bayes bound	pretrain

Dataset	MLP–tanh		PBGNet$_\ell$		PBGNet			PBGNet$_{\text{pre}}$		
	R_{in}	R_{out}	R_{in}	R_{out}	R_{in}	R_{out}	Bound	R_{in}	R_{out}	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

Quest for generalisation guarantees (about half *via* PAC-Bayes)

Directions:

- Generic bounds (relaxing assumptions such as iid or boundedness, new concentration inequalities, . . .)
- Tight bounds for specific algorithms (deep neural networks, NMF, . . .)
- Towards new measures of performance (CVaR, ranking, contrastive losses, . . .)
- Coupling theory and implemented algorithms: bound-driven algorithms
- Applications (providing guidelines to machine learning users, sustainable / frugal machine learning)

The Inria-London Programme

UCL Inria

References I

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019. ISSN 0027-8424. doi: 10.1073/pnas.1903070116. URL https://www.pnas.org/content/116/32/15849.

G. K. Dziugaite and D. M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, 2017.

B. Guedj. A Primer on PAC-Bayesian Learning. In *Proceedings of the second congress of the French Mathematical Society*, 2019. URL https://arxiv.org/abs/1901.05353.

J. Langford and R. Caruana. (Not) Bounding the True Error. In *NIPS*, pages 809–816. MIT Press, 2001.

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

D. McAllester. Some PAC-Bayesian theorems. In *Proceedings of the International Conference on Computational Learning Theory (COLT)*, 1998.

D. McAllester. Some PAC-Bayesian theorems. *Machine Learning*, 37, 1999.

J. Shawe-Taylor and R. C. Williamson. A PAC analysis of a Bayes estimator. In *Proceedings of the 10th annual conference on Computational Learning Theory*, pages 2–9. ACM, 1997. doi: 10.1145/267460.267466.