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

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

Memorising the already seen data is usually bad → 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.)
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:** $R_{\text{out}}(h) = \mathbb{E}[\ell(h(X), Y)]$ (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 $\Delta(h) \leq \epsilon(m, \delta)$

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

**Flavours:**
- distribution-free
- algorithm-free
- distribution-dependent
- 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 \( \delta > 0 \),

\[
P \left[ R_{\text{out}}(h) \leq R_{\text{in}}(h) + \epsilon(m, \delta) \right] \geq 1 - \delta.
\]

Think of \( \epsilon(m, \delta) \) as 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

$$\Pr^m \left( \forall Q \text{ on } \mathcal{H}: R_{out}(Q) \leq R_{in}(Q) + \sqrt{\frac{\text{KL}(Q\|P) + \ln \frac{2\sqrt{m}}{\delta}}{2m}} \right) \geq 1 - \delta,$$

with $R_{in}(Q) \equiv \int_{\mathcal{H}} R_{in}(h) \, dQ(h)$, $R_{out}(Q) \equiv \int_{\mathcal{H}} R_{out}(h) \, dQ(h)$,

$$\text{KL}(Q\|P) = \mathbb{E}_{h \sim Q} \ln \frac{Q(h)}{P(h)}.$$

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 \( Q \),

\[
\text{Error on unseen data} \leq \text{Error on sample} + \text{complexity term}
\]
\[
R_{\text{out}}(Q) \leq R_{\text{in}}(Q) + F(Q, \cdot)
\]

This defines a principled strategy to obtain new learning algorithms:

\[
h \sim Q^* \\
Q^* \in \text{arg inf}_{Q \ll P} \left\{ R_{\text{in}}(Q) + F(Q, \cdot) \right\}
\]

(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

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 \( k^{\text{th}} \) layer
- \( \sigma : \mathbb{R} \rightarrow \mathbb{R} \) 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

\[
f_\theta (\mathbf{x}) = \sigma (\mathbf{w}_L \sigma (\mathbf{w}_{L-1} \sigma (\ldots \sigma (\mathbf{w}_1 \mathbf{x})))).
\]
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: $\mathbb{E}_{\theta' \sim \mathcal{N}(\theta, \Sigma)} R_{in}(f_{\theta'}) \rightarrow$ estimated by sampling

Complexity term: $\text{KL}(\mathcal{N}(\theta, \Sigma) \| \mathcal{N}(\theta_0, \Sigma_0)) \rightarrow$ 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
- \( \text{sgn}(a) = 1 \) if \( a > 0 \) and \( \text{sgn}(a) = -1 \) otherwise

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

Prediction

\[
f_{\theta}(\mathbf{x}) = \text{sgn}(\mathbf{w}_L \text{sgn}(\mathbf{W}_{L-1} \text{sgn}(\ldots \text{sgn}(\mathbf{W}_1 \mathbf{x}))))) ,
\]
Building block: one layer (aka linear predictor)

Letarte et al. (2019)

Model \( f_w(x) \overset{\text{def}}{=} \text{sgn}(w \cdot x) \), with \( w \in \mathbb{R}^d \).
- Linear classifiers \( \mathcal{F}_d \overset{\text{def}}{=} \{ f_v \mid v \in \mathbb{R}^d \} \)
- Predictor \( F_w(x) \overset{\text{def}}{=} E_{v \sim Q_w} f_v(x) = \text{erf} \left( \frac{w \cdot x}{\sqrt{d} \| x \|} \right) \)
- 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 $\mathcal{F}_D = \{f_\tilde{\theta} | \tilde{\theta} \in \mathbb{R}^D \}$, where

$$f_\theta(x) = \text{sgn}(w_2 \cdot \text{sgn}(W_1 x)) .$$

$$F_\theta(x) = \mathbb{E}_{\tilde{\theta} \sim Q_\theta} f_\tilde{\theta}(x)$$

$$= \int_{\mathbb{R}^{d_1} \times d_0} Q_1(V_1) \int_{\mathbb{R}^{d_1}} Q_2(v_2) \text{sgn}(v_2 \cdot \text{sgn}(V_1 x)) dv_2 dV_1$$

$$= \int_{\mathbb{R}^{d_1} \times d_0} Q_1(V_1) \text{erf} \left( \frac{w_2 \cdot \text{sgn}(V_1 x)}{\sqrt{2} \| \text{sgn}(V_1 x) \|} \right) dV_1$$

$$= \sum_{s \in \{-1, 1\}^{d_1}} \text{erf} \left( \frac{w_2 \cdot s}{\sqrt{2}d_1} \right) \int_{\mathbb{R}^{d_1} \times d_0} \mathbf{1}[s = \text{sgn}(V_1 x)] Q_1(V_1) dV_1$$

$$= \sum_{s \in \{-1, 1\}^{d_1}} \text{erf} \left( \frac{w_2 \cdot s}{\sqrt{2}d_1} \right) \prod_{i=1}^{d_1} \left[ \frac{1}{2} + \frac{S_i}{2} \text{erf} \left( \frac{w_i \cdot x}{\sqrt{2} \| x \|} \right) \right].$$
Generalisation bound

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

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

<table>
<thead>
<tr>
<th>Model name</th>
<th>Cost function</th>
<th>Train split</th>
<th>Valid split</th>
<th>Model selection</th>
<th>Prior</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP–tanh</td>
<td>linear loss, L2 regularized</td>
<td>80%</td>
<td>20%</td>
<td>valid linear loss</td>
<td>-</td>
</tr>
<tr>
<td>PBGNet(_\ell)</td>
<td>linear loss, L2 regularized</td>
<td>80%</td>
<td>20%</td>
<td>valid linear loss</td>
<td>random init</td>
</tr>
<tr>
<td>PBGNet</td>
<td>PAC-Bayes bound</td>
<td>100%</td>
<td>-</td>
<td>PAC-Bayes bound</td>
<td>random init</td>
</tr>
<tr>
<td>PBGNet(_{pre})</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>– pretrain</td>
<td>linear loss (20 epochs)</td>
<td>50%</td>
<td>-</td>
<td>-</td>
<td>random init</td>
</tr>
<tr>
<td>– final</td>
<td>PAC-Bayes bound</td>
<td>50%</td>
<td>-</td>
<td>PAC-Bayes bound</td>
<td>pretrain</td>
</tr>
</tbody>
</table>

| Dataset     | MLP–tanh |  | PBGNet\(_\ell\) |  | PBGNet |  | PBGNet\(_{pre}\) |  |
|-------------|----------|--------------------------------|----------------|----------|------------------------------|----------------|----------------|--------------|----------------|----------------|----------------|
|             | \(R_{in}\) | \(R_{out}\) | \(R_{in}\) | \(R_{out}\) | \(R_{in}\) | \(R_{out}\) | \(R_{in}\) | \(R_{out}\) | \(Bound\) | \(R_{in}\) | \(R_{out}\) | \(Bound\) |
| 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)
References I


