Bias-Variance vs Approximation-Estimation

Christopher Yau
University of Oxford

22 October 2025

Question Suppose you have a dataset for sentiment analysis with limited labeled data (e.g., 100 text examples with 0-1 sentiment label). You have three models:

Model A Regression trained from scratch.

Model B A deep neural network trained from scratch.

Model C A large pretrained Language Model (e.g., GPT-3), fine-tuned on your 100 labeled examples.

How do you assess which one is better?

Example data

```
0.9 Absolutely outstanding performance.

0.8 The project exceeded all expectations.

0.1 A disappointing result despite strong effort.

0.7 The presentation was clear and articulate.

0.8 A deeply emotional and inspiring speech.

0.2 A poor execution of an ambitious plan.
```

The eternal pursuit of the modern AI PhD student:

Dataset	My Model	Your Model 1	Your Model 2
MNIST	**99**	67	76
CIFAR-10	**97**	54	81
CelebA	**95**	34	46

Table: A column of boldness leads to the path of NeurIPS Glory

You should also have as many rows and columns as possible to further highlight the **boldness** of your model.

- ✗ Never benchmark with data that is actually relevant to your problem.
- ✗ Never hyperparameter optimise other models (just use the settings on Github or the paper)
- ✗ Never admit that, in any meaningful way, the linear/logistic model is basically just as good.
- ✓ Always use the metric(s) which best highlights your **boldness**
- ✓ Keep changing the random number generator seed until your **boldness** is at its greatest
- ✓ Apply **early stopping** if other models will overtake yours.
- ✓ Keep **adding regularisation** terms to your loss function until your model lies at the top of the **boldness** mountain.

"Leaderboard mentality" is a challenge for AI.

Impacting on standards and *recognition* of rigour.

Leads to:

- Flawed research directions and insights,
- Real-world translation / generalisability,
- Bad research practice and culture.

NPCL: Neural Processes for Uncertainty-Aware Continual Learning

Saurav Jha
UNSW Sydney
saurav.jha@unsw.edu.au

Dong Gong*
UNSW Sydney
dong.gong@unsw.edu.au

He Zhao
CSIRO's Data61
he.zhao@ieee.org

Lina Yao
CSIRO's Data61, UNSW Sydney
lina.yao@data61.csiro.au

Method	**S-CIFAR-10** *Class-IL*		**S-CIFAR-100** *Class-IL*		**S-Tiny-ImageNet** *Class-IL*		**P-MNIST** *Domain-IL*		**R-MNIST** *Domain-I*	
Joint ResNet	92.2 ±0.15		70.44		59.99±0.19		94.33±0.17		95.76±0.0	
Joint NP	91.66±0.11		70.58±0.24		59.83±0.17		95.02±0.21		95.37±0.0	
Joint ANP	91.26±0.16		70.77±0.21		60.14±0.17		95.39±0.18		95.85±0.0	
Joint NPCL	92.74±0.12		71.46±0.20		60.18±0.22		95.97±0.14		96.11±0.0	
Multitask NPCL	69.15±0.09		53.6±0.21		35.53±0.13		87.40±0.10		89.21±0.0	
oEWC [44]	19.49±0.12		-		7.58±0.10		75.79±2.25		77.35±5.7	
SI [54]	19.48 ±0.17		-		6.58 ±0.31		65.86±1.57		71.91 ±5.3	
LwF [34]	19.61±0.05		-		8.46 ±0.22		-		-	
$\mathcal{M}_{\text{size}}$	200	500	500	2000	200	500	200	500	200	
ER [43]	44.79±1.86	57.74±0.27	22.10	38.58	8.49±0.16	9.99±0.29	72.37±0.87	80.6±0.86	85.01±1.90	88.9
iCaRL [42]	49.02 ±3.20	47.55 ±3.95	46.52	49.82	7.53±0.79	9.38±1.53	-	-	-	
FDR [2]	30.91±2.74	28.71±3.23	-	-	8.70±0.19	10.54±0.21	74.77±0.83	83.18 ±0.53	85.22±3.35	89.6
RPC [41]	-	-	22.34	38.33	-	-	-	-	-	
DER [4]	61.93±1.79	70.51±1.67	36.6	51.89	11.87 ±0.78	17.75±1.14	81.74±1.07	87.29±0.46	90.04±2.61	92.2
NP [14]	46.1±3.44	59.3±2.76	22.92	38.70	8.32±0.62	10.2±0.34	70.02±1.44	79.44±0.81	85.03±2.7	88.1
ANP [27]	46.67±1.23	58.77±0.65	23.2	39.06	8.81±0.93	9.75±0.90	73.55±0.66	80.98±0.57	85.70±1.39	89.2
ST-NPCL (w/ only per-task latent)	54.6±2.14	65.22±1.89	28.45	42.1	10.92±1.03	13.7±1.35	76.4±1.62	82.06±0.92	86.99±3.07	89.6
Naive NPCL (w/o task head inf.)	19.54±3.44	20.71±3.09	18.27	18.90	7.19±1.02	8.48±0.90	68.37±1.58	73.3±0.81	81.13±2.91	83.6
NPCL (ours)	**63.78**±1.70	**71.34**±1.48	**37.43**	**46.71**	**12.44**±0.59	**15.29**±1.02	**83.11**±0.90	**86.52**±0.77	**91.48**±1.79	**92.0**

Type	Method	iCIFAR10 *CIL*	iCIFAR100 *CIL*	S-TinyImagenet *CIL*	MNIST *TIL*
(a)	ER	0.577	0.221	0.099	0.972
	DER	0.705	0.366	0.178	0.983
	NPCL[3]	0.702	0.198	0.135	-
	CSReL-LODE	0.398	0.420	0.228	-
(b)	EWC	0.182	0.092	0.056	0.441
	oEWC-MACL	0.206	0.088	0.079	-
	LwF	0.196	0.145	0.094	0.472
	SSRE	0.355	0.179	0.110	-
	DS-AL	0.408	0.205	**0.122**	-
	MeRGAN	0.251	0.122	0.078	0.670
	FeTriL	0.417	<u>0.208</u>	0.115	-
	NICE	**0.551**	0.203	<u>0.120</u>	-
	PRER	0.412	0.197	0.108	-
	ReReLRP	0.399	0.058	0.033	-
	Boo-VAE	-	-	-	<u>0.892</u>
	CL-BRUNO$_{\alpha_2=0}$	0.257	0.146	0.056	0.734
	CL-BRUNO	<u>0.421</u>	**0.212**	0.117	**0.947**

NCPL claimed 0.374 and 0.153!

1. Confusion around Bias/Variance and Approximation/Estimation
2. What it actually means to be Bayesian
3. Correlation vs Causation
4. Are you sure that you are calculating your metrics correctly?

Today's lecture:

1. What is classical learning theory?
2. Why deep learning broke it?
3. And why modern learning theory isn't a one-size-fits-all.

Condensing 80 years of learning theory into less than 2 hours!

Note: No proofs.

These terms are pervasive in the ML literature:

- **Bias**
- **Variance**
- **Approximation error**
- **Estimation error**

Often used interchangeably but actually mean fundamentally different things.

$\mathbf{X}$ is a random variable denoting an **input** (use lower case $\mathbf{x}$ for observed)

$\mathbf{Y}$ is a random variable denoting an **output** (use lower case $\mathbf{y}$ for observed)

$P(\mathbf{X}, \mathbf{Y})$ is a joint probability distribution for $\mathbf{X}$ and $\mathbf{Y}$

Use shorthand $P(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}) = P(\mathbf{x}, \mathbf{y})$

$f(\mathbf{x})$ is a function or **model** applied to $\mathbf{x}$ where $f \in \mathcal{F}$

$l(\mathbf{y}, f(\mathbf{x}))$ is a **loss function** which measures the discrepancy between $f(\mathbf{x})$ and an observed $\mathbf{y}$

Given a model $f(\mathbf{x})$ we want a way of measuring how well it performs or alternatively the **risk** associated with using the model.

We formalise this idea using the notion of **expected loss, or risk**.

The **population risk** of a model is defined as:

$$R(f) := \mathbb{E}_{\mathbf{x},\mathbf{y}}[l(\mathbf{y}, f(\mathbf{x}))] = \int l(\mathbf{y}, f(\mathbf{x}))dP(\mathbf{x}, \mathbf{y})$$

This is the expected loss with respect to the *true* probability distribution $P(\mathbf{x}, \mathbf{y})$.

Note: In general, we can never know this risk since we generally will not know $P(\mathbf{x}, \mathbf{y})$ (or the DGP - data generating process).

The **Bayes model** f^0 is the (possibility not unique) function which minimises the population risk:

$$f^0 \in \arg\inf_{f \in \mathcal{F}} R(f)$$

where $\mathcal{F}$ denotes the set of possible candidate models

The quality of each model is measured by its risk, $R(f)$

The notation $\arg\inf_{f \in \mathcal{F}}$ indicates choosing the best possible model from all the candidates in the set where "best" means having the lowest possible expected risk.

If f comes from a restricted family of functions $\mathcal{F}'$ then the **best-in-family** model f^*:

$$f^* \in \arg \inf_{f \in \mathcal{F}'} R(f)$$

It is possible that $f^* \neq f^0$ if the family of functions does not contain the Bayes model.

Examples:

- $\mathcal{F}'$ is the set of all linear models but Bayes model is a polynomial function.

- $\mathcal{F}'$ is the set of all ReLU deep net models but Bayes model is differentiable everywhere.

Data: $D = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n\} \sim_{iid} P(\mathbf{x}, \mathbf{y})$ (unknown)

Empirical Risk:

$$\overline{R}(f) = \frac{1}{n} \sum_{i=1}^{n} l(\mathbf{y}_i, f(\mathbf{x}_i))$$

This is an **empirical approximation** to the true risk based on the n observed data pairs.

Empirical Risk Minimiser (ERM):

$$\hat{f} \in \arg \inf_{f \in \mathcal{F}'} \overline{R}(f)$$

In general $\hat{f} \neq f^* \neq f^0$

Empirical Risk Minimisation (ERM) is the standard framework for supervised learning.

ERM derived functions are **random quantities** since they are a function of the sample data they are derived from.

(In Statistics, you would say that the ERM has a *sampling distribution*).

If we were to draw a new sample from the population and re-train, we would generally obtain a different ERM estimator.

Quick Aside

The randomness arises due to sampling, i.e. the possibility that you could have observed a different dataset, this is a *frequentist* perspective.

f^0 - Bayes model, best model, population data distribution

f^* - Best-in-class model, best model under restricted family, population data distribution

$\hat{f}$ - ERM model, best model under restricted family, sample data

$$R(\hat{f}) \geq R(f^*) \geq R(f^0)$$

Each model leads to different risks with the latter two having additional risk due to model and data constraints.

Concerns the differences in risk between models:

$$\underbrace{R(\hat{f}) - R(f^0)}_{\text{Excess Risk}} = \underbrace{R(\hat{f}) - R(f^*)}_{\text{Estimation Error}} + \underbrace{R(f^*) - R(f^0)}_{\text{Approximation Error}}$$

Excess Risk is the difference in risk between your ERM model compared to the Bayes model

Estimation Error is the difference in risk between your model and the Best in class model

Approximation Error is the difference in risk between the Best in class model and the Bayes model

Question: Suppose you have a dataset for sentiment analysis with limited labeled data (e.g., 100 labeled examples and corresponding sentiment score from 0-1).

You have three models:

Model A Linear regression trained from scratch.

Model B A deep neural network trained from scratch.

Model C A large pretrained Language Model (e.g., GPT-3), fine-tuned on your 100 labeled examples.

Note: Assume suitable feature extractors are available for models A and B.

Question: 100 labeled examples and corresponding sentiment score from 0-1.

Model A Linear regression trained from scratch.

Model B A deep neural network trained from scratch.

Model C A large pretrained Language Model (e.g., GPT-3), fine-tuned on your 100 labeled examples.

Answer these questions in groups:

- How do the approximation and estimation errors differ among Models A, B, and C?
- What additional sources of error could affect Models B and C?

Hint: Think about model complexity, data efficiency, and prior knowledge.

- **Approximation Error:** $R(f^*) - R(f^0)$
  - This is error due to *restricted model class*
  - Model B has lower approximation error than Model A because it can represent a richer set of functions.
- **Estimation Error:** $R(\hat{f}) - R(f^*)$
  - This is error due to *finite sample size*.
  - With few data and many parameters, Model B typically has higher estimation error than Model A (risk of overfitting).
  - However, in some overparameterised regimes, optimisation and regularisation may reduce effective estimation error (double descent).

There are two additional error types to be considered:

Optimisation Error: $R(\tilde{f}) - R(\hat{f})$

- This error arises from imperfect optimisation e.g., the algorithm fails to find the global (or good local) minimum of the loss
- Could be due to, e.g., non-convexity, poor initialisation, or limited training.

Model specification error

- Arises when model class includes non-admissible or physically impossible functions, leading to unrealistic or invalid approximations
- Particularly relevant for deep networks due to non-convex loss surfaces.

The application of Approximation-Estimation Error analysis to foundation models is not so straightforward.

Foundation Models (FMs):

- Pretrained on massive and diverse external datasets.
- Exhibit extremely high representational capacity.

Implications:

- **Low approximation error** from expressive architectures which can approximate a wide range of functions.
- **Low estimation error** and fine-tuning requires only limited task-specific data.
- BUT: Emergence of a new type of error...

Target Data: $D = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n\} \sim_{\text{iid}} P(\mathbf{x}, \mathbf{y})$

Pre-training data: $D_{\text{FM}} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m\} \sim_{\text{iid}} P_{\text{FM}}(\mathbf{x}, \mathbf{y})$

Empirical Risk for the foundation model:

$$\overline{R}(f_{\text{FM}}) = \frac{1}{n} \sum_{i=1}^{n} l(\mathbf{y}_i, f_{\text{FM}}(\mathbf{x}_i))$$

but pretraining data $(\mathbf{x}, \mathbf{y})$ drawn from the pre-training distribution not the new data distribution.

If $P(\mathbf{x}, \mathbf{y})$ is not the same as $P_{\text{FM}}(\mathbf{x}, \mathbf{y})$ ...

We introduce a third form of error: $R_P(\hat{f}_{\text{FM}}) - R_{P_{\text{FM}}}(\hat{f}_{\text{FM}})$:

$$\text{Risk}(\hat{f}_{\text{FM}}) - R(f^0) = \text{Estimation} + \text{Approximation} + \text{Transfer Error}$$

- Arises from a mismatch between pretraining and target task distributions.
- Produces outputs that are plausible but systematically incorrect.

Foundation models succeed when transfer error is small but, when it's large, **alignment** becomes the key challenge.

In practice, these error types can often be distinguished by how model performance changes with more data or different inputs.

Error Type	**Indicator**
Approximation	Fails even with lots of data
Estimation	Improves with more labeled data
Transfer/Alignment	Fails despite low training error

Tip: Probe the model with OOD inputs, contradictory prompts, or synthetic edge cases.

Example code implementing the three models is available here.

Github: `https://github.com/cwcyau/foai-cdt-module1`

Note: Regression with a good feature extractor does surprisingly well.

BREAK HERE

Empirical Risk Minimisation (ERM) picks the model with the lowest loss on your sample.

But: *How reliable is this performance on new data?*

We need a theory of learning that generalises — one that's *Probably Approximately Correct*.

Key Idea:

- PAC learning defines when an algorithm's training performance guarantees true performance.

  "Probably (with high confidence) $\approx$ Approximately (within small error) Correct."

Takeaway:

- PAC formalises the intuition behind **estimation error** — turning "how much data is enough?" into a mathematical question.

An Informal-Formal Definition

A learning algorithm is **PAC-learnable** if, for all $\varepsilon > 0$ and $\delta > 0$, there exists a sample size $n(\varepsilon, \delta)$ such that:

$$\Pr\Big(R(\hat{f}) - R(f^*) \leq \varepsilon\Big) \geq 1 - \delta$$

- $R(\hat{f})$: risk of the learned model
- $R(f^*)$: minimum achievable risk in the hypothesis class
- ε ("approximately"): allowed accuracy gap
- δ ("probably"): confidence level

Formal Definition (informal form)

A learning algorithm is **PAC-learnable** if, for all $\varepsilon > 0$ and $\delta > 0$, there exists a sample size $n(\varepsilon, \delta)$ such that:

$$\Pr\Big(R(\hat{f}) - R(f^*) \leq \varepsilon\Big) \geq 1 - \delta$$

Interpretation:

With probability $\geq 1 - \delta$, the learner's true error is within ε of optimal.

Suppose $\mathcal{F}$ denotes the *hypothesis class* - the set of all candidate models or functions the learner can choose from.

A **finite hypothesis class** means the learner can choose from a **finite set of hypotheses**:

$$|\mathcal{F}| < \infty$$

That is, there are only finitely many distinct functions

$$f : \mathcal{X} \to \mathcal{Y}$$

in the class $\mathcal{F}$.

Having finitely many parameters $\neq$ finite hypothesis class.

Example

Linear model:

$$f_w(x) = w^\top x, \quad w \in \mathbb{R}^d$$

Even though w has finitely many components, it can take infinitely many values in $\mathbb{R}^d$. Hence, $\{f_w : w \in \mathbb{R}^d\}$ is *uncountably infinite*.

Neural networks and polynomial models with real-valued weights are also **infinite** hypothesis classes.

Examples of Hypothesis Classes

Hypothesis class $\mathcal{F}$	**Finite?**	**Explanation**
$\{f_1, f_2, f_3\}$	Yes	Only 3 total hypotheses.
$\{f_w(x) = w^\top x : w \in \mathbb{R}^2\}$	No	Continuous range of w.
{Decision stumps on a finite feature set}	Yes	Finitely many threshold splits
{All polynomials of degree ≤ 2}	No	Real-valued coefficients.

Theorem

PAC Bound for Finite $\mathcal{F}$:
If $|\mathcal{F}| < \infty$, then for any $\varepsilon, \delta > 0$, with probability at least $1 - \delta$:

$$R(\hat{f}) - R(f^*) \leq \sqrt{\frac{1}{2n}\left(\log|\mathcal{F}| + \log\frac{2}{\delta}\right)}$$

Interpretation:

- Larger hypothesis class $\Rightarrow$ larger $|\mathcal{F}|$ $\Rightarrow$ need more samples n.
- Higher confidence (smaller δ) $\Rightarrow$ need more data.
- Gives concrete **sample complexity:** $n = O\big(\frac{1}{\varepsilon^2}\left(\log|\mathcal{F}| + \log\frac{1}{\delta}\right)\big)$

Sample complexity:

$$n = O\left(\frac{1}{\varepsilon^2}\left(\log|\mathcal{F}| + \log\frac{1}{\delta}\right)\right)$$

Intuition:

- More hypotheses $\Rightarrow$ higher risk of overfitting.
- PAC bound quantifies how much data ensures reliable generalisation.
- Finite hypothesis classes are always PAC-learnable.

Proofs: See
`https://vatsalsharan.github.io/lecture_notes/lec2_final.pdf`.

Motivation:

- The PAC bound for finite $\mathcal{F}$ depends on $\log|\mathcal{F}|$.
- But what if $\mathcal{F}$ is **infinite**? (e.g., all linear classifiers, neural networks)
- We need a new way to measure the **capacity** or **complexity** of a hypothesis class.

Idea: Replace $\log|\mathcal{F}|$ with a notion of *effective size* — how many distinct labellings a class $\mathcal{F}$ can represent on finite samples.

Definition (VC Dimension):

- The **Vapnik–Chervonenkis (VC) dimension** of $\mathcal{F}$, denoted $\mathrm{VC}(\mathcal{F})$, is the size of the largest set of points that $\mathcal{F}$ can *shatter*.
- "Shatter" = correctly realise all 2^n possible labelings of n points.
- Applies to binary classification (generalisations: Natrajan, pseudo-dimensions)

Interpretation:

- VC dimension generalises the notion of model complexity to infinite function classes.
- Finite VC dimension $\Rightarrow$ PAC learnability still holds.

Definition: A hypothesis class $\mathcal{F}$ **shatters** a set of points $S = \{x_1, x_2, \ldots, x_n\}$ if, for *every possible labeling* $y_1, \ldots, y_n \in \{0, 1\}$, there exists a function $f \in \mathcal{F}$ such that:

$$f(x_i) = y_i \quad \text{for all } i = 1, \ldots, n.$$

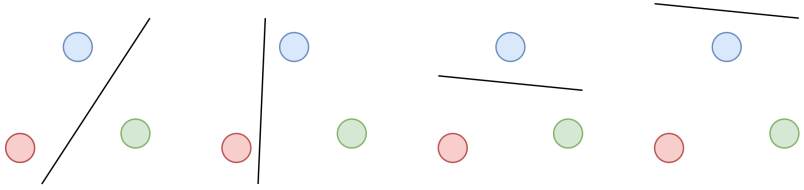
Intuition:

- "Shattering" measures how **expressive** a hypothesis class is.
- If $\mathcal{F}$ can realise every possible labeling of n points, it can **separate** or "shatter" that set.
- The largest n such that $\mathcal{F}$ can shatter some set of n points defines the **VC dimension**.

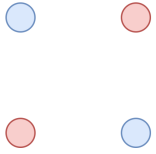
Example: Linear classifiers in 2D

- Any 3 non-collinear points can be separated in all $2^3 = 8$ ways.
- No single line can realise all labelings of 4 points (e.g., XOR pattern).
- $\Rightarrow$ VCdim(2D linear classifiers) $= 3$.

Example: Linear classifiers in 2D



With 3 points, there are four ways of splitting the points (plus four if we flip the assignment)



XOR Pattern - no single linear decision boundary can split this pattern

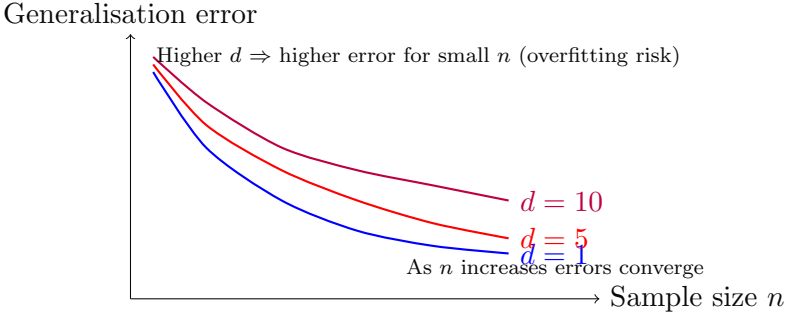
Theorem

PAC Learnability via VC Dimension: If a hypothesis class $\mathcal{F}$ has finite VC dimension $d = \text{VCdim}(\mathcal{F})$, then $\mathcal{F}$ is **PAC-learnable**. Moreover, for any $\varepsilon, \delta > 0$, with probability at least $1 - \delta$:

$$R(\hat{f}) - R(f^*) \leq O\left(\sqrt{\frac{d\log(n/d) + \log(1/\delta)}{n}}\right)$$

- The **VC dimension** d determines how much data n is required to generalise.
- Finite $d \Rightarrow$ learnable; infinite $d \Rightarrow$ no uniform generalisation guarantee.
- Larger d increases the risk of **overfitting** unless n grows accordingly.

The generalisation gap $R(\hat{f}) - R(f^*)$ decreases with more data (n) but increases with model complexity (VC dimension d).



There is a trade-off: complex models can fit more patterns, but need more data to generalise reliably.

Classical View (PAC–VC):

- Finite VC dimension ⇒ learnable.
- Increasing VC dimension ⇒ higher risk of overfitting.
- Generalisation guaranteed only when model capacity is controlled wrt to sample size and hypothesis class size (personal error tolerance).

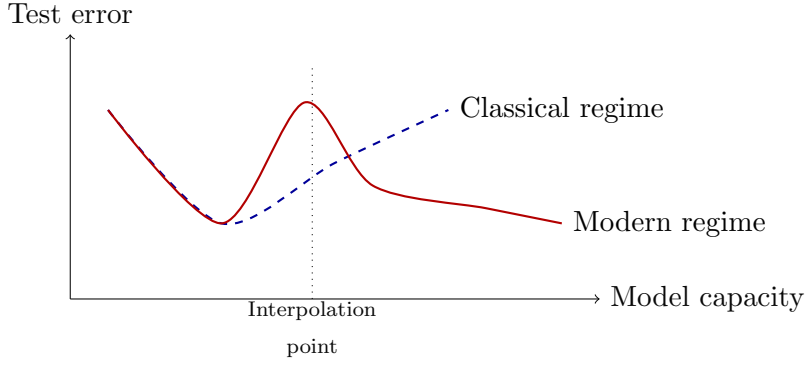
Modern Reality: Deep and Overparameterised Networks

- Neural networks often have **very large VC dimensions**.
- Yet they **generalise well** in practice — even when they can fit random labels.
- Indicates that VC dimension alone cannot explain modern generalisation behaviour.

Takeaway: Classical VC theory formalised when learning *can* generalise.

Modern theory asks *why* large models often *do*.

As capacity increases *beyond the interpolation point* (where training error ≈ 0), test error often **decreases again**.



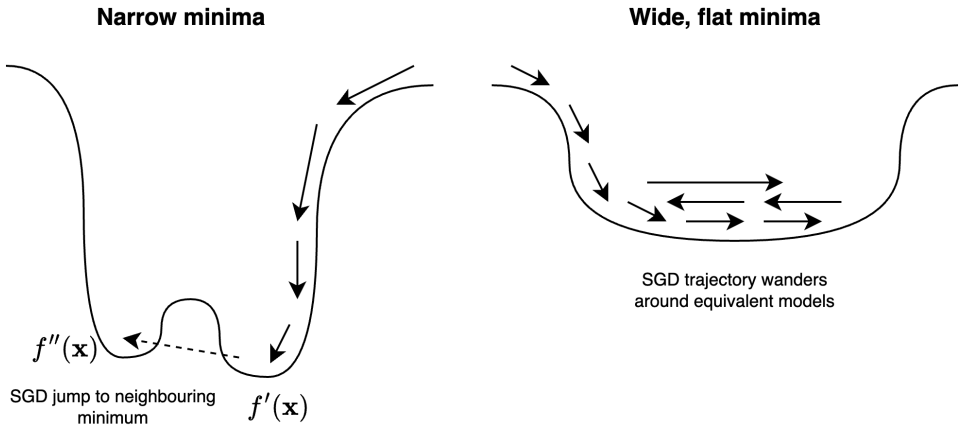
This behaviour is known as the **double descent** curve.

Q: Why does this occur?

In general, no universal explanation exists at present.

One explanation is given by **implicit regularisation** — the learning algorithm itself biases the solution towards simpler functions.

- **Optimization bias:** Gradient descent and SGD tend to find low-complexity (flat-minima) solutions among many possible interpolating ones.
- **Overparameterisation:** High-dimensional parameter spaces often contain many "benign" minima with similar training loss but different generalisation behaviour.
- **Effective capacity control:** Even with large VC dimension, only a subset of hypotheses are reached by the optimiser.


Narrow minima
Wide, flat minima
$f''(\mathbf{x})$
SGD jump to neighbouring minimum
$f'(\mathbf{x})$
SGD trajectory wanders around equivalent models

Other explanations include:

- After the interpolation point, additional parameters can help align model behaviour with data structure rather than memorising noise.
- This leads to the second descent in test error.

Takeaway: Generalisation in deep learning is governed not only by model capacity, but influenced by the **implicit biases of the optimisation process**.

PAC Learning vs Foundation Models: Broken Assumptions

Classical PAC Setting	**Foundation / Pre-trained Models**
Data are i.i.d. from a fixed distribution $P(x, y)$	Pretraining data drawn from diverse, heterogeneous sources
Fixed and limited hypothesis class $\mathcal{H}$	Extremely large or effectively unbounded model class (massive capacity)
Empirical risk minimisation from scratch	Fine-tuning or prompting on top of pretrained parameters
Same training and deployment objective	Objectives differ: pretraining vs. downstream alignment or prompting
Explicit generalisation guarantees via sample complexity	No direct PAC-style guarantees; rely on empirical scaling and robustness

Result: Classical PAC assumptions break, but the *intuition*—balancing capacity, data, and distribution alignment—still guides modern learning theory.

- **Sample complexity → data efficiency**: pretraining amortises sample cost across many tasks.
- **Capacity control** shifts from explicit (VC bounds) to implicit (optimiser dynamics, early stopping, layer freezing).
- **Estimation error** replaced by **transfer / alignment error**:

$$R(\hat{f}_{\mathrm{FM}}) - R(f^*) = \text{Approx.} + \text{Estimation} + \text{Transfer Error}.$$

- Domain adaptation bounds (Ben-David et al., 2010):

$$R_T(f) \leq R_S(f) + \text{divergence}(P_S, P_T) + \lambda.$$

Concept	Classical View	Modern View
PAC Learning	Guarantees low generalisation error with enough i.i.d. data; assumes fixed objective and distribution.	Pretraining and alignment break i.i.d. assumption; goal is to recover PAC-like reliability under distribution shift.
VC Dimension	Measures model capacity combinatorially; controls sample complexity.	Capacity effectively unbounded; generalisation controlled implicitly by optimisation, architecture, and data diversity.
Rademacher Complexity	Empirical measure of ability to fit noise on given data; tightens generalisation bounds.	Analogous ideas used via regularisation, weight decay, and stability analysis of pre-trained representations.
Alignment / Transfer Error	Not part of classical theory; assumes same $P(x, y)$ for train/test.	Central issue: mismatch between pre-training and deployment distributions causes "alignment error" in outputs.

BREAK HERE

VC bounds depend only on the **size or capacity** of the hypothesis class. They do not capture:

- The effect of **priors or biases** over hypotheses.
- The **data-dependent structure** of modern learning algorithms.
- How stochastic training (e.g. SGD) or random initialisation affect generalisation.

Idea of PAC–Bayes:

- Instead of analysing a single hypothesis $\hat{f}$, we reason about a **distribution over hypotheses**.
- Combine PAC learning's **"probably approximately correct"** framework with Bayesian-style reasoning using a **prior** and **posterior**.
- Leads to tighter, data-dependent generalisation bounds — especially powerful for **stochastic or overparameterised models**.

In short: PAC–Bayes provides a bridge between classical learning theory and modern probabilistic deep learning.

Goal: To bound the **true risk** of a stochastic predictor that samples hypotheses from a learned distribution.

Setting:

- Let $\mathcal{F}$ be a hypothesis class (e.g., neural networks).
- Training data $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ drawn i.i.d. from $P(\mathbf{x}, \mathbf{y})$.
- Define a **prior distribution** $P(f)$ over hypotheses — independent of the data.
- After seeing data, the learner produces a **posterior distribution** $Q(f|D)$.
- Prediction is made by sampling $f \sim Q$ and using $f(\mathbf{x})$.

Risks:

$$\text{Empirical risk: } \overline{R}(Q) = \mathbb{E}_{f\sim Q}\left[\frac{1}{n}\sum_{i=1}^{n}\ell(f(\mathbf{x}_i),\mathbf{y}_i)\right]$$

$$\text{True risk: } R(Q) = \mathbb{E}_{f\sim Q}\left[\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim P}\,\ell(f(\mathbf{x}),\mathbf{y})\right]$$

Goal of PAC–Bayes:

- Bound $R(Q)$ in terms of $\overline{R}(Q)$, the complexity term $\mathrm{KL}(Q\,\|\,P)$, and the sample size n.

Theorem

Theorem (McAllester, 1999):

- Let $\mathcal{F}$ be any hypothesis class, and P a prior distribution over $\mathcal{F}$ independent of data.
- For any $\delta \in (0, 1)$, with probability at least $1 - \delta$ over samples $D \sim P(x, y)^n$, the following holds for all posteriors Q:

$$R(Q) \leq \overline{R}(Q) + \sqrt{\frac{\mathrm{KL}(Q \,\|\, P) + \log \frac{2\sqrt{n}}{\delta}}{2n}}$$

Proofs: Pierre Alquier, `https://arxiv.org/pdf/2110.11216`

$$R(Q) \leq \hat{R}(Q) + \sqrt{\frac{\mathrm{KL}(Q \,\|\, P) + \log \frac{2\sqrt{n}}{\delta}}{2n}}$$

- $R(Q)$ — true (population) risk — what we want to bound
- $\overline{R}(Q)$ — empirical (training) risk — what we can compute
- $\mathrm{KL}(Q \,\|\, P)$ — complexity term — penalises deviation from the prior

Intuition:

- A posterior Q that stays close to the prior P (low KL) generalises better.
- The bound holds for all Q, so it applies to **any learning algorithm**.
- Captures both **data fit** and **model complexity** in one expression.

PAC–Bayes View:

- The term $\mathrm{KL}(Q \,\|\, P)$ measures how much the posterior Q (after training) diverges from the prior P (before seeing data).
- It captures how much **information about the data** the learner encodes in its model.

Interpretation:

- Small $\mathrm{KL}(Q \,\|\, P) \rightarrow$ posterior close to prior $\rightarrow$ less data memorisation $\rightarrow$ **stronger generalisation**.
- Large $\mathrm{KL}(Q \,\|\, P) \rightarrow$ posterior far from prior $\rightarrow$ model has overfit $\rightarrow$ **weaker generalisation**.

Back to our observation:

- In deep networks, different parameter settings can achieve near-zero training loss.
- Yet models found by stochastic optimisation (e.g., SGD) often generalise better.

PAC–Bayes Perspective:

- A **flat minimum** corresponds to a region in parameter space where many nearby weights perform well.
- A posterior Q concentrated over such a wide region stays **close to the prior** $\rightarrow$ smaller $\mathrm{KL}(Q \,\|\, P)$.
- Hence, flat minima $\Rightarrow$ better PAC–Bayes generalisation.

Implicit Regularisation:

- SGD noise and finite learning rates favour wider, smoother minima.
- This acts as an **implicit prior** that keeps Q low in complexity.

Note: This is a looser definition of "prior" than what Bayesian Statistics would define.

This alternative characterisation:

- Provides a **data-dependent** view of generalisation — the bound adapts to the posterior Q found by the training algorithm.
- Works naturally with **stochastic models or training** (e.g. dropout, SGD noise).
- Bonus: It turns out to offer interpretable terms: empirical loss + complexity penalty (KL term).

1. **Classical PAC Learning:**
   - Guarantees generalisation when empirical risk is low and hypothesis class is simple.
   - Complexity measured by **VC dimension** or finite class size.
2. **Limitations of VC Theory:**
   - Cannot explain good generalisation in **overparameterised models**.
   - Ignores data-dependent and algorithmic effects (e.g. SGD noise, implicit bias).

A general comment is that PAC theory only works by assuming *worse-case scenarios* which provides *strong* mathematical guarantees but *weak* practical insight.

3. PAC–Bayes Framework:

- Introduces **priors and posteriors** over hypotheses.
- Balances **empirical fit** ($\hat{R}(Q)$) with **complexity** ($\mathrm{KL}(Q\|P)$).
- Naturally extends to stochastic or Bayesian learning settings.

Takeaway:

- PAC–Bayes starts to unify statistical learning theory and modern deep learning intuition: good generalisation arises when training finds **simple posteriors that fit the data**.

1. Computational Intractability

- Computing the true posterior $Q(f|D)$ or optimising the PAC–Bayes bound is often intractable for complex models.
- Requires approximations (e.g., Gaussian, low-rank, variational) that may not capture the real training dynamics.

Note: Stochastic equivalents of approximation, estimation and optimisation error re-appear.

2. Choice of Prior

- The bound depends on a **data-independent prior** $P(f)$.
- Selecting a realistic yet tractable prior is difficult in practice.
- "Data-dependent priors" can tighten bounds, but complicate theory and may break assumptions.

Note: Diffusion Models are one solution to this. Sampling from generative models.

3. Loose or Asymptotic Bounds

- The constants in PAC–Bayes bounds can be loose, making them **qualitatively informative** but not numerically tight.
- Bounds are asymptotic in nature — they often require large n to be meaningful.

4. Interpretability in Deep Networks

- Mapping SGD dynamics to an explicit Q or P is still an open challenge.
- Difficult to connect theoretical posteriors to real training trajectories.

Takeaway: PAC–Bayes provides deep insight into generalisation, but practical implementation and precise interpretation remain areas of active research.

1. Data–Dependent Priors

- Replace fixed priors $P(f)$ with priors informed by data or model structure.
- Example: using a small subset of training data or pretrained weights to define P.
- Enables **tighter, more realistic** generalisation bounds.

2. Approximate and Variational Posteriors

- Learn tractable approximations to $Q(f|D)$ (e.g., Gaussian or low-rank).
- Leads to practical **optimisable PAC–Bayes objectives** for neural networks.
- Example: PAC–Bayes training via stochastic gradient variational inference.

3. Connections to Flatness and Compression

- Recent work links PAC–Bayes bounds to **flat minima** and **model compression**.
- Flat posteriors $\Rightarrow$ low $\mathrm{KL}(Q\|P)$; compressible models $\Rightarrow$ low description length.

4. Towards Practical Bounds for Deep Learning

- Scalable empirical PAC–Bayes bounds for modern architectures.
- Combining with information–theoretic or algorithmic stability approaches.

Recent example: Omar Rivasplata "PAC-Bayesian Reinforcement Learning Trains Generalizable Policies" `https://arxiv.org/pdf/2510.10544`, pg. 3

2.2 Probably Approximately Correct (PAC) Learning Bounds

In a supervised learning task, the instance space is taken to be the product $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} \subseteq \mathbb{R}^d$ is the feature space and $\mathcal{Y}$ the label space; commonly $\mathcal{Y} \subseteq \mathbb{R}$ for regression, and $\mathcal{Y} \subseteq \mathbb{N}$ for classification problems. In this setting, a learning algorithm is a function that takes in a training sample $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ and returns a prediction function $f_\theta : \mathcal{X} \to \mathcal{Y}$, also referred to as a hypothesis, parametrized by $\theta \in \Theta$, where $\Theta \subset \mathbb{R}^p$ denotes the set of all admissible parameter vectors. An unknown data distribution $\mathcal{D}$ over $\mathcal{Z}$ is postulated, with $\mathcal{D}_\mathcal{X}$ denoting the marginal distribution on $\mathcal{X}$, and training samples $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ are such that each pair $(\boldsymbol{x}_i, y_i) \in \mathcal{Z}$ is an independent and identically distributed (i.i.d.) random draw from $\mathcal{D}$, that is, $S \sim \mathcal{D}^{\otimes m} := \mathcal{D} \otimes \cdots \otimes \mathcal{D}$ (m copies). The "quality" of a hypothesis f_θ is typically assessed through a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$, which quantifies the discrepancy between predicted and true outputs (labels). The performance of a given hypothesis is measured by its *true risk*, and its *empirical risk* on the training sample S, given by

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{(\boldsymbol{x},y)\sim\mathcal{D}}\big[\ell\big(f_\theta(\boldsymbol{x}), y\big)\big], \qquad \hat{\mathcal{L}}_S(\theta) = \frac{1}{m}\sum_{i=1}^{m} \ell\big(f_\theta(\boldsymbol{x}_i), y_i\big).$$

In supervised machine learning, the goal is to learn a hypothesis f_θ that accurately predicts labels $y \in \mathcal{Y}$ for given inputs $\boldsymbol{x} \in \mathcal{X}$. Since training is based on a dataset $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$, a central question is: how can we ensure that the learned function f_θ will perform well on unseen data?

Probably Approximately Correct (PAC) learning answers this question via probability inequalities saying that, under suitable restrictions, for a chosen level δ within $(0, 1)$ we have

$$\Pr_{S\sim\mathcal{D}^m}\big\{\mathcal{L}(\theta) \le \hat{\mathcal{L}}_S(\theta) + \epsilon\big\} \;\ge\; 1 - \delta.$$

Choosing a small δ then translates into a high-confidence bound for $\mathcal{L}(\theta)$. Concrete PAC bounds specify how large m must be (or how large the gap ϵ can be) in terms of properties of the hypothesis class, *e.g.* VC-dimension, Rademacher complexity, stability, compression, etc. All of the classical PAC bounds treated f_θ as a deterministic output of the algorithm.

2.3 PAC-Bayesian Bounds

The **PAC-Bayesian** framework (McAllester, 1999; Seeger, 2003; Guedj, 2019; Shalaeva et al., 2020; Alquier, 2024) extends the PAC learning paradigm to analyze distributions over hypothesis, rather than individual hypotheses. This framework has been useful in studying generalization for stochastic learning algorithms and prediction rules based on randomizing or averaging. Let Θ denote the set of parameters defining a family of prediction functions $\{f_\theta : \mathcal{X} \to \mathcal{Y}\}_{\theta\in\Theta}$. Prior to observing data, a *prior* distribution $\mu \in \mathcal{P}(\Theta)$ is specified over Θ. Upon receiving a training sample $S \sim \mathcal{D}^{\otimes m}$, the learning algorithm then selects a *posterior* distribution $\rho \in \mathcal{P}(\Theta)$. PAC-Bayesian theory provides high-confidence bounds on the population risk $\mathbb{E}_{\theta\sim\rho}[\mathcal{L}(\theta)]$ in terms of the empirical risk $\mathbb{E}_{\theta\sim\rho}[\hat{\mathcal{L}}_S(\theta)]$ and an additional term that effectively constrains the complexity of the posterior distribution ρ via an information quantity measuring the discrepancy between posterior ρ and prior μ, typically the Kullback-Leibler divergence $\mathrm{KL}(\rho\|\mu)$, but there are other choices. Formally, for any $\kappa > 0$ and chosen level $\delta \in (0, 1)$, the following inequality holds with probability of at least $1 - \delta$ over the random draw of the training sample S, simultaneously for all distributions ρ:

$$\mathop{\mathbb{E}}_{\theta\sim\rho}[\mathcal{L}(\theta)] \;\le\; \mathop{\mathbb{E}}_{\theta\sim\rho}[\hat{\mathcal{L}}_S(\theta)] \;+\; \frac{1}{\kappa}\Big(\mathrm{KL}(\rho\|\mu) + \ln\tfrac{1}{\delta} + \Psi_{\ell,\mu}(\kappa, n)\Big) \tag{3}$$

with

$$\Psi_{\ell,\mu}(\kappa, n) = \ln \mathop{\mathbb{E}}_{\theta\sim\mu}\Big[\exp\big(\kappa\,\big(\mathcal{L}(\theta) - \hat{\mathcal{L}}_S(\theta)\big)\big)\Big].$$

Compared with classical PAC guarantees, PAC-Bayes offers two ad
reinforcement learning: (1) *Data-dependent priors (Parrado-Hernánd*

Recent example: "PAC-Bayesian Reinforcement Learning Trains Generalizable Policies" https://arxiv.org/pdf/2510.10544

Potential supervisors:

1. Patrick Rebeschini (Stats) https://www.stats.ox.ac.uk/~rebeschi/teaching/AFoL/22/index.html
2. Varune Kanade (CS) https://www.cs.ox.ac.uk/people/varun.kanade/teaching/CLT-MT2022/lectures/CLT.pdf
3. Jared Tanner (Maths)
4. Coralia Cartis (Maths)

BREAK HERE

This assessment exercise will look at:

Is Bias-Variance the same as Approximation-Estimation?

In machine literature, these terms are often used interchangeably, but is it correct to do this?

Working in groups of 3-4:

1. Read the paper "Bias/Variance is not the same as Approximation/Estimation" by Brown and Ali (2024) *JMLR*.
2. Develop a 15 minute presentation that will:
   - 2.1 Summarise the difference between B/V and A/E (Section 3).
   - 2.2 Include a numerical example illustrating the concept (use the published examples to build upon).
   - 2.3 Technical level: As if I was to add another section to this lecture!

Due: 7 Nov: One hour of presentations, followed by a lecture.

Note: see Canvas for full details including group assignments.