A Novel Analysis on Learning Theory

Bui Gia Khanh, Luong Van Tam, Dang Trung
Independent Researchers

July 30, 2025

Abstract

The analysis of learning action, machine learning and related practices in the field theoretically has been made by utilizing Computational Learning Theory (CoLT) Valiant [1984] and Statistical Learning Theory (SLT) Vapnik [1999]. These two theories, while overlapped, provides a general framework in analysing and justifying learning actions and learning model constructions, aiding in the formation of modern practices. One of the famous insight using such framework is the *bias-variance tradeoff* Geman et al. [1992], which states that model complexity and generality inversely affect each other, thus guarantee the need for a safe bracket between them. However, recent literatures, Belkin et al. [2019] has indicated the fallout of such dilemma by the new phenomenon called *double descent*, where there exists an interpolation threshold that renders the current statistical justification for bias-variance inconsequential to a given region. Various 'anomaly' has also been detected similarly, with varying degree of sophistication and potential. In this paper, we analyse the classical learning framework, investigating aspects and concepts related to the formation of the insight of bias-variance dilemma, double descent, and give a separated interpretation and explanation of the learning theory as well as double descent. Furthermore, we also interject with one of the particular experimental result on GNN - a special case where the observed double descent does not occur at all, yet.

1 Introduction

Machine learning and its modern practice has been developed and researched, of a substantial portion by empirical and heuristic approach, either by advancing new practices, architectures or by try-and-test modification. From its early onset of a regression estimator $\theta(x, y)$ on the linear regression problem, machine learning has developed substantially. Certain model concept with great successes includes regression-classification model, Bayesian modelling, generative learning model, support vector machine, Gaussian processes, and more. Of all such, the more formal and complex model architecture created, is the concept of a *neural network*. With increasingly sophisticated architecture, heuristic approach become popular, the fast-paced advancement of the field comes with new method, new results, new observations, and its far-reaching application which led to even bigger and larger scale deployment, there has been questions about the formation and status of a theoretical ground, a rigorous matter on the side of **theoretical machine learning**.

While rigorous and well-formulated in a sense, classical and theoretical machine learning was dwarfed by the modern advancement of machine learning as a whole, leading to several anecdotal problems regarding the interpretation of phenomena, the reevaluation of the theory to fit the more updated analysis, and explanation to more sophisticatedly designed system. This and many more, plus as present, many of such advancements and improvements are heuristic, and the general theory and conceptual understanding remain limited, led to the choice to often opt for analogies and empirical workaround. Ultimately, this resulted in multiple 'failures' in explaining, interpreting, and predicting behaviours of new phenomena appeared in the modern landscape of machine learning. Thereby, we suggest a fix.

2 Main focus

Our main focus is on the topic of the umbrella term classical learning theory, and the recent phenomena observed named 'double descent'. Statistical learning theory (SLT) and computational learning theory (CLT) Vapnik [1999], Mohri et al. [2012], Shalev-Shwartz and Ben-David [2014], Hajek and Raginsky [2021], Bousquet et al. [2020] has been prominent in constructing a well-rounded formal theory surrounding learning problems, models, and machine learners analysis. Valuable insights have been dissected from treatment of statistical theory and mathematical modelling on models, including the *bias-variance tradeoff* Geman et al. [1992], Domingos [2000a], which serves as a bound for efficient learning and model configuration (or complexity). However, recently there has been observations of *double descent* Belkin et al. [2019], Schaeffer et al. [2023], Nakkiran et al. [2019], Lafon and Thomas [2024] which refute the famous tradeoff assumption, and hence brings question to the establishment of the theory, as well as several assumptions and insight in the framework. Further events and phenomena observed also includes grokking and triple, to n-descent Davies et al. [2023], d' Ascoli et al. [2020]. Furthermore, many problems of defining and formalizing notions used in designing and implementing machine learning models are inconclusive, such as, for example, *model complexity* and others.

The phenomena *double descent* itself has been investigated somewhat thoroughly, firstly introduced by Belkin et al. [2019]. Further analysis was made by several literatures, particularly conjectured the existence of double descent to the concept of model complexity and inductive bias. Nakkiran et al. [2019] expanded the phenomena into deep neural network models. Their conclusion is reached by considering the perturbation of a learning procedure $\mathcal{T}$ on the effective model complexity $\mathrm{EMC}_{\mathcal{D},\epsilon}(\mathcal{T})$ defined in the paper, separating the eventual phenomena into regions of observations, thereby in one way or another, predicting the tendency of double descent. This is also the first, arguably, "concise" definition of double descent, even though its nature is an empirical definition, including the notion of model complexity. Preliminary works are done by Lafon and Thomas [2024], Schaeffer et al. [2023], and Liu and Flanigan [2023] on the role of optimization in double descent. Davies et al. [2023] attempted to unifying grokking with double descent, theorized a possibility of similarity during the generalization phase transition of the inference period, and Olmin and Lindsten [2024] attempted to explain epoch-wise double descent within the model of two-layer linear network. However, it is mentioned that it can also be expanded into deep nonlinear networks.

3 Contribution

1. **Analysis on (statistical) learning theory and the theoretical treatment of model selection in learner setting**: We analyse and discuss of the characteristics, structures of the learning theory and model selections to identify patterns, assumptions, notions, different narrative or system dynamic regarding the problem of model selection at large, and double descent on its own. One of the main problem in rationalizing double descent is the absent of well-formulated theoretical justification for bias-variance tradeoff, the principle which it breaks, and different consideration between researchers and authoritative sources.
2. **Expansion on fundamental experiments on bias-variance and double descent**: Aside from already conducted experiments and studies by Sharma and Aiken [2014], Schaeffer et al. [2023], Nakkiran et al. [2019], Belkin et al. [2019], Geman et al. [1992], uni, Fortmann [2012], Neal [2019], we expand on a variety of others experiment with carefully crafted setting and control. This includes models of which can be regarded as test models, such as polynomial regression, multi-logistic regression, radial basis model, hyperplane models (precursor of modern neural network), support vector machine (Cristianini and Shawe-Taylor [2000]), and classical multilayer-perceptron from Goodfellow et al. [2016].
3. **Hypothesis forming and conjectures**: During the study, we also propose plenty of conjectures and

hypotheses in development, most typically of concern with interpreting results and behaviours, as well as the analysis taken afterward on the problem. Furthermore, we also link up with a physical interpretation in the sense of statistical mechanics.

4. **Expansion on practical experiments on bias-variance and double descent**: We also present the additional analysis on a more practical problem, of the **graph neural network** (GNN) from Hamilton, Scarselli et al. [2009], Lopushanskyy and Shi [2024], Tanis et al. [2024], Bronstein et al. [2021], Veličković [2023]. One particular fact pre-experiment is some reports deliberately recorded an absence of double descent in several of their neural network model on graph.

4 Preliminary

Most of the paper will argue about different features of statistical learning theory (Sterkenburg [2024], Vapnik [1999], Hajek and Raginsky [2021]). It is then imperative to discuss the background setting of such. We are given the observations, or dataset of the form $\mathcal{S} = (\mathcal{X}, \mathcal{Y}) \subset \mathbb{R}^n \times \mathbb{R}^m$, of all 2-tuple pairs, assumed to be sampled or observed and governed by a distribution $\mathcal{D}$.

$$\mathcal{S} = \{(x_1, y_1), (x_2, y_2), \dots (x_n, y_n)\} \subset \mathcal{X} \times \mathcal{Y}$$

The dataset is assumed to be i.i.d. sampling according to $\mathcal{D}$, which is unknown by the priori.

The learning problem is then formulated as followed. Given the machine learning model expressed a hypothesis h of the hypothesis class $\mathcal{H}$, the learning theory aims for creating a procedure to learn either elements of the concept class $\mathcal{C}$ of all concepts $c : \mathcal{X} \to \mathcal{X}$, or the function class $\mathcal{F}$ of all functions $f : \mathcal{X} \to \mathcal{Y}$, which is usually set to $\{0, 1\}$ or $[0, 1]$. This distinction is trivial, hence, if it is clear, we will talk about the concept class $\mathcal{C}$ only as representative form. The learner $\mathcal{L}(h)$ consider the set of possible hypothesis $\mathcal{H}$, in which might not coincide with $\mathcal{C}$. It receives a partial image of sample $S = (x_1, \dots, x_2, \dots, x_n)$ drawn i.i.d. according to $\mathcal{D}$ as well as the label $(c(x_1), \dots, c(x_n))$. This constitutes the dataset $\mathcal{S}$, which are based on specific concept $c \in \mathcal{C}$ for the hypothesis to learn. The task is then to use (or *extract*) meaningful information to select a hypothesis $h_S \in \mathcal{H}$ that accurately mimic c, with marginal error Θ. The notation h_S stands for all hypothesis that can be inferred from the range of the dataset (the first argument, $\mathcal{X}$).

The marginal error Θ is considered of two parameters, the empirical error $\hat{R}(h)$ and the generalization error $R(h)$. We give the following definition of them.

Definition 4.1 (Empirical risk). *Given a hypothesis $h \in \mathcal{H}$, a target concept $c \in \mathcal{C}$, and a sample $S = (x_1, \dots, x_m)$. For some particular $\epsilon > 0$, the* ***empirical error*** *or empirical risk of h is defined by*

$$\hat{R}_S(h) = \mathop{\mathbb{P}}_{x \in S \sim \mathcal{D}}[\ell\{h(x), c(x)\} \geq \epsilon] = \frac{1}{m} \sum_{i=1}^{m} \ell\{h(x_i), c(x_i)\} \tag{1}$$

Definition 4.2 (Generalization risk). *Given a hypothesis $h \in \mathcal{H}$, a target concept $c \in \mathcal{C}$, and an underlying distribution $\mathcal{D}$ on $\mathcal{X}$. For some particular $\epsilon > 0$, the generalization error or risk of h is defined by*

$$R(h) = \mathop{\mathbb{P}}_{x \sim \mathcal{D}}[\ell\{h(x), c(x)\} \geq \epsilon] = \mathop{\mathbb{E}}_{x \sim \mathcal{D}}[\ell\{h(x), c(x)\}] = \int_{x \in \mathcal{D}} \ell\{h(x), c(x)\} \, dP(x) \tag{2}$$

In essence, the empirical risk measure the hypothesis-to-observation error, and the generalization risk captures the difference of the hypothesis to the actual concept. Notice that by doing this, we have implicitly assumed that the observations and the concept is not the same thing, under the majority of scenarios. For the observations to be *approximately equal* or sufficiently captures the concept, there then would have to be various criteria to fulfil. This can be illustrated as Illustration 1.

For fixed $\mathcal{H}$, for fixed and sufficiently large $\mathcal{S}$, and no observation (data) errors, the empirical risk is the generalization risk. These two measures between h and c constitute the learning problem, which can also be separated into both cases - either empirical learning or generalization learning, one to minimize $\hat{R}(h)$, and one to minimize $R(h)$.

Definition 4.3 (Empirical learning problem). *We present the formal form of the empirical learning. Suppose we have a* ***target****, $c \in \mathcal{C}$, where $\mathcal{C}$ is an arbitrary concept class that captures targets of the same type. Suppose we are*

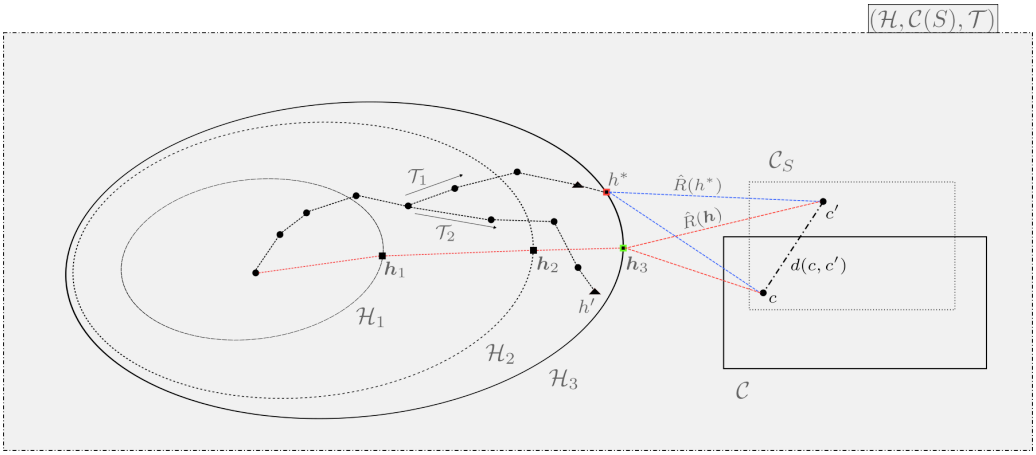


Figure 1: **Illustrative dynamic of the learning problem**. For an incremental hypothesis space sequence $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3$, we aim to obtain the procedure $\mathcal{T}$ that would either reach the *generalization solution* $\boldsymbol{h}$, or the *empirical solution* h^* for a particular hypothesis class, with respect to the concept c and its observed concept c'. Model selection hence dictates how an algorithm or procedure than choose the best possible hypothesis to approximate the generalization solution from the empirical solution set. Do notice that here we explicitly state that the data would create a **proxy concept** that overlaps with the concept class and of some arbitrary 'distance' from the true concept by $d(c, c')$. In case the two classes overlap, there still exists the arbitrary distance. Also, we can also observe the intuitive notion of increasing hypothesis class - while it indeed can help in getting closer to the concept class, the somewhat intrinsic property of current learning theory being the relative random procedure class make it probabilistically unstable.

provided a set of observations $\mathcal{S}$. The problem is to use certain algorithm $\mathcal{A}$ using $\mathcal{S}$, to obtain a hypothesis h^* for a fixed $\mathcal{H}$ such that:

$$R(h^*) = \min_{h \in \mathcal{H}} \hat{R}(h) = \min_{h \in \mathcal{H}} \mathop{\mathbb{E}}_{x \sim \mathcal{D}, x \in \mathcal{S}} \ell\{h(x), c(x)\} \tag{3}$$

The generalization best $\boldsymbol{h}$ is also called in literature as the *Bayes model*, such that it reduces the Bayes risk that is the infimum of the generalization risk, $R^* = \inf_{h \in \mathcal{H}} R(h)$. The hypothesis h^* is often called the *empirical best*, for it being the minimal, finite hypothesis of the lowest loss evaluation on the entire observation space $\mathcal{S}$. There exists no certified assumption regarding whether h^* aligns with the minimal generalization error.

Definition 4.4 (Generalization learning problem). *We present the formal form of the generalization learning problem. Suppose we have a* ***target****, $c \in \mathcal{C}$, where $\mathcal{C}$ is an arbitrary concept class that captures targets of the same type. Suppose we are provided a set of observations $\mathcal{S}$. Supposed we have an algorithm $\mathcal{A}$ that for fixed hypothesis space $\mathcal{H}$, equation 3 holds true. The problem is to use certain algorithm $\mathcal{A}'$ such that, under limited availability, to obtain $\boldsymbol{h}$, satisfies:*

$$R(\boldsymbol{h}) = \min_{h \in \mathcal{H}} R(\boldsymbol{h}) \leq \{\epsilon\}, \quad \epsilon > 0 \tag{4}$$

For a set of risk bounds ϵ. If the setting is deterministic, then there exists $\epsilon = 0$.

Then, we can partially say that generalization problem is an advancement from empirical solution. As such, empirical solution imposes the observed concept, and not the concept itself. This is reflected in

modern machine learning landscape, by modelling the concept as distributions, and then using notions such as KL-divergence to measure such relative disparity.

These two stages of learning procedure comes of as the intrinsic design of machine learning setting. As opposed to interpolation, where empirical learning problem would have to be put into first priority to achieve the best possible point-through model, machine learning leans on approximation more and a separated criterion that is only visible in the learning setting - the fact that we are using c' to extract c. This disparity is exactly the **generalization problem**, of which is then also the goal of machine learning, to approximate the general solution using the observed solution. One then can ask what is the difference between the two notions, and when we can guarantee that both will 'converge' to the same point. The process of **model selection** is then the procedure that will optimize $h \in \mathcal{H}$ to a given target fixture, such as the empirical best.

One of the major assumption or observation from the point of the hypothesis class, is that the hypothesis cannot know the generalization error. Thereby, the more efficient and often used procedure/algorithm in training, is then the *empirical risk minimization* (ERM) method, minimizing only the empirical risk to the empirical best; then, certain measures to 'generalize' this heuristically is apprehended on the hypothesis. The process of model selection is also the place that we get the concept of **bias-variance tradeoff**.

Suppose that for a concept class $\mathcal{H}$, we can then partition it to $\mathcal{H} = \bigcup_{k\in\mathbb{N}} \mathcal{H}_k$ for finite k hypotheses. Assume that they satisfy the uniform convergence (a very big assumption):

$$\forall k \in \mathbb{N}, \quad \mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}\left(\sup_{h\in\mathcal{H}_k} R(h) - \hat{R}_S(h) \leq \epsilon_k(m,\delta)\right) \geq 1-\delta \tag{5}$$

for functions ϵ_k satisfies that for all k and $\delta \in (0,1)$, $\lim_{m\to\infty} \epsilon_k = 0$. One of the major assumption or observation from the point of the hypothesis class, is that the hypothesis cannot know the generalization error. Thereby, the more efficient and often used procedure/algorithm in training, is then the *empirical risk minimization* (ERM) method, minimizing only the empirical risk to the empirical best; then, certain measures to 'generalize' this heuristically is apprehended on the hypothesis:

$$\mathsf{ERM}(h') = \arg\min_{h\in\mathcal{H}} \hat{R}(h) \tag{6}$$

Another way is to use the *structural risk minimization* (SRM) method, which add a regularizer $r(\cdot)$ after the empirical error,

$$\mathsf{SRM}(h') = \arg\min_{h\in\mathcal{H}} \hat{R}(h) + \lambda r(h) \tag{7}$$

The regularizer term $r(\cdot)$ takes of its argument the model, and calculate based of certain measure of its complexity. Thereby, the goal is to penalize structures with high complexity and favour simpler designs, in principle.

In this research, we would like to presume the ERM method in question as the famous *gradient descent*. We refer to Achlioptas, Ruder [2017] for general view of the algorithm, and Zhang [2019] for a source on particularly gradient descent algorithm on deep learning models. Nominally, for the nth output sequence w_n, it updates the model to a given path of optimization for w_{n+1} such that

$$w_{n+1} = w_n - \alpha_n \nabla_{w_n}(\mathcal{L}(h(x), y)) \tag{8}$$

for $\alpha_n \geq 0$ the step size of the iteration, and with a (potentially) convex function $\mathcal{L}(h(x), y)$ of the hypothesis class $\mathcal{H}$.

4.1 Sample set partitions

Previously, we established that at least in said formal setting, generalization errors cannot be known beforehand. Thereby, all of our operations, procedures, processes has to be conducted using the observed data, or the sample space $\mathcal{S}$. A solution to utilize that is to consider the generalization region as the region of *new, unseen observations* that while still is of the same concept c, but it is indeed, unseen if the model is never provided of such instances. Hence, it prompts the utilization of limited resources by then to partition the sample space into various subspaces, that would be then fed into the learning process as observational space. This is simply called **sample set partitioning**. Denoted the number of partition (discrete) as k, then for $k = 2$, then this is called the **train-test partition** (or dataset). This contains a dataset $\mathbf{x}_1$ and $\mathbf{x}_2$ of the same original sample set, such that $\mathbf{x}_1/\mathbf{x}_2 \geq 1$ (one is bigger than another, usually). The ratio does indeed affect the overall running procedure, accordingly.

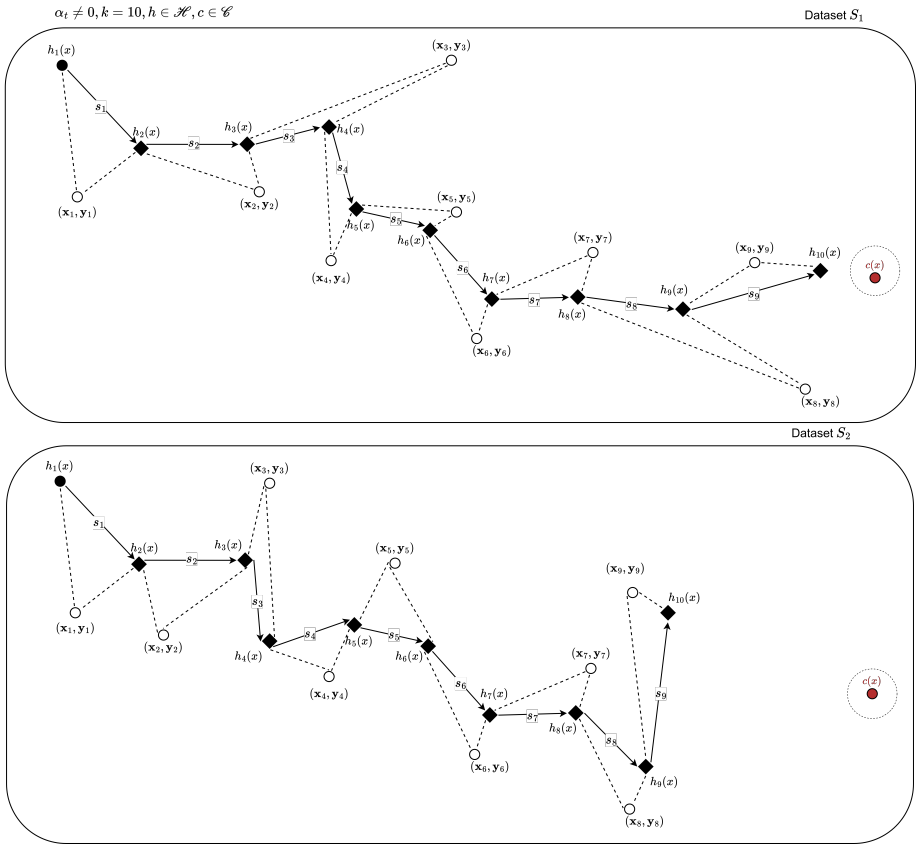


Figure 2: **Conceptual representation of the sample set partitions and its effect on iterative process**. (1), for S_1, and of the specified ordering, h is able to make it to the optimal point compared to the actual concept c. The bubble around c is what we call irreducible error, intrinsic of the observational space. (2) for S_2, of the changing dataset, while of the same partition but also changing order, gives different volatile path, and perhaps suboptimal performance compared to the first dataset case. Note that they are the supposed *optimal path* of both S_1, S_2. If randomization is introduced, may suboptimal path will occur, and the result will differ.

If $k > 3$ and above, we call it the nth partitioning. If we further consider the choice of observational spaces being provided, then if $k > 3$, and subspaces are chosen randomly, it is called **cross-validating partition**. The effect of them are fairly well-understood, and we might as well refer to authoritative sources on such issue.

Based on the above conclusion, the empirical error and generalized error can be re-calculated to fit the general scheme. For $|\mathcal{S}| = m$ and partition $k + p = m$ for train and test partition respectively,

$$\hat{R}_k(h) = \frac{1}{k}\sum_{i=1}^{k} \ell\{h(x_i), c(x_i)\}, \quad \hat{R}_p(h_{\mathcal{A}}) = \frac{1}{k}\sum_{i=1}^{p} \ell\{h_{\mathcal{A}}(x_i), c(x_i)\} \tag{9}$$

where $h_{\mathcal{A}}$ is the result from the learning algorithm (procedure), for $h^* \neq h_{\mathcal{A}}$. Then, the empirical learning seeks to optimize the first term, that is, for the objective

$$\arg\min_{h\in\mathcal{H}} \hat{R}_k(h) = \arg\min_{h\in\mathcal{H}} \frac{1}{k}\sum_{i=1}^{k} \ell\{h(x_i), c(x_i)\} \tag{10}$$

and the generalization learning as:

$$\begin{aligned} \arg\min_{h\in\mathcal{H}} \hat{R}_p(h_{\mathcal{A}}) &= \hat{R}_k(h_{\mathcal{A}}) + \arg\min_{h\in\mathcal{H}} \hat{R}_p(h_{\mathcal{A}}) \\ &= \arg\min_{h\in\mathcal{H}} \left[\frac{1}{k}\sum_{i=1}^{k} \ell\{h_{\mathcal{A}}(x_i), c(x_i)\} + \frac{1}{k}\sum_{i=1}^{p} \ell\{h_{\mathcal{A}}(x_i), c(x_i)\}\right] \\ &= \arg\min_{h\in\mathcal{H}} \left[\epsilon + \frac{1}{k}\sum_{i=1}^{p} \ell\{h_{\mathcal{A}}(x_i), c(x_i)\}\right] \end{aligned} \tag{11}$$

By the definition order, the generalization error can only be calculated post-$\mathcal{A}$. Hence, we often called the empirical, or ERM-generalizer case to be dynamic, and the solution for ERM is tested statically on generalization set. This partition of order of computation though, brings up the problem of updating factors and others contributing potential that will damage or diffuse the measure.

For cross-validation and other randomize partition, the formulation requires more care in the notion by itself. The structural risk minimization scheme dilemma can be understood from the above consideration naturally. We have effectively defined the two proxies, **heuristic empirical risk** and **heuristic generalization risk**.

While they are effective as a pair, just as Figure 2 illustrated, overly optimizing the path can lead to it in general, be fairly ineffective. On the flip side, not sufficiently optimizing the path will lead to sub-average result, leading to adverse relation. Furthermore, since $\mathcal{S}$ might not, in general, be representative of the population that depicts c, the form c' created from observations of c-generated space (we call this the **observed concept** to the true concept) might be skewed, leading to adverse effect. From there, we gain the notion of **overfitting** and **underfitting** under the constraints of limited knowledge and reduced concept space to the observation space, relative to the true concept itself. It is also from here that the proxy concepts of **bias** and **variance** are introduced to construct the bias-variance tradeoff, and subsequently, the phenomena of double descent.

Another point to note is also the way of partitioning and location of the data. As we have suggested, data comes in a variety of range, of which we can directly infer from using the supremum set and the infimum element, given particular structure and encoding space. Denoted by $\mathbf{r}_{\mathcal{S}}$, it is responsible for

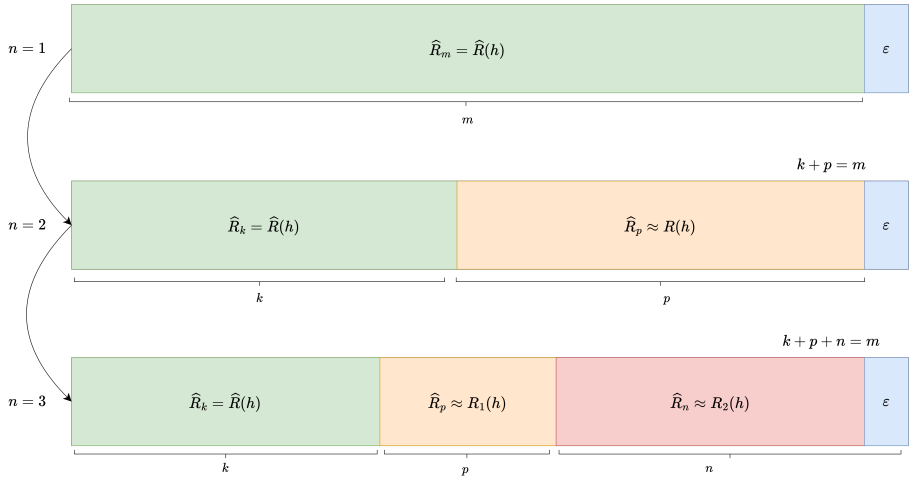


Figure 3: Partitioning process and its error potential consideration. We assume each partition includes the irreducible error ϵ accompanied by the n partition, belongs to the furthest partitioning set. Within every increasing partition, for supposed distributed data (unordered data), the generalization risk is further decomposed.

the partitioning strength of the dataset on its own. Then, partitioning must also take the randomization chance, or the distribution and distinctive ordering of the dataset. That is, the metadata related to the dataset must also be configured, as suggestibly, there would be some observable behaviours typically available with, for example in permutation or linear ordering of data by magnitude and expression range. If that is the case, and the data is discretely fragmented, but not diffused in the manner that spread out the missing information all around the dataset, the result would be entirely different, and it would influence the generalization error, heuristically, far more than what is realized. Thereby, there then exists two fundamental polarity: either it is the *totally ordered set*, or it is *randomized uniform aberration*, for any given observation space.

4.2 Datasets and its potential effects

The effect that data has on the model should also be taken into account, as of fact. As far as we are concerned of, the shape, practical system that the dataset gives the setting, the factors running into the dataset, the choice of the dataset being sampled, selected in partitions, and other factors must also be analysed. Without such, we leave 'gaps' in which sensitive, potentially chaotic behaviours might occur frequently and would not be identified, or being classified into irreducible errors, static of the learning setting. As such, we might as well list a few potential points of interest.

1. **Data distribution**: the distribution of the dataset itself affects how the concept is represented in the space. If the representation space created by using the data is large, but sparse, then the condition would make for some behaviours to occur more easily. This is one degree of freedom of the system setting. If the dataset spread is low, yet the data is very volatile, then it will lead to inconclusive picture of the actual concept, hence introducing new behaviours on the now unknown region of the distribution. As such, the majority of those unknown spaces will treat every point there as a potential outlier instead.

2. **Data complexity**: The complexity of the data itself is of interest. In a controlled setting of response-inducing evaluation, identifying behaviours of the model with a glimpse on the data complexity -

how the actual data works with the internal mechanics accompanied - is of specific interest, as it can affect some of the majorative effects and chaotic behaviours that occur in the learning process, especially stochastic process and probabilistic optimization patterns.

3. **Data partitions**: The partitioning of data in the previously mentioned sample partitioning section might give rise to previously observed behaviours, as well as explaining particular volatility when changing the test partitioning set. The ordering of the dataset also helps in such regard.

It is also worth to note that since we emphasize the use of dataset partitioning to replicate the empirical-generalization guarantee, it is also then indicative that such partitioning would have its separate way of calculating test error on itself.

4.3 Overfitting and underfitting

Under such guise, one understanding about the concept of underfitting and overfitting James et al. [2013] can be reached, using the notion of partitioning. We can define the heuristic for better definition, as followed.

Definition 4.5 (Heuristic empirical risk). *Given a dataset $\mathcal{S}$ of size m, supposedly of the concept $c \in \mathcal{C}$, and the hypothesis $h \in \mathcal{H}$, then the **heuristic empirical risk** take a partitioning $k < m$ with $k/m \geq 1/2$, hence defining the heuristic empirical loss on k, denoted $\hat{R}_k(h)$.*

Definition 4.6 (Heuristical generalization risk). *Given a dataset $\mathcal{S}$ of size m, supposedly of the concept $c \in \mathcal{C}$, and the hypothesis $h \in \mathcal{H}$. Fixed $\hat{R}_k(h)$, for the final optimization of algorithm $\mathcal{A}$, then the **heuristic generalization risk** take the remaining 2-partition of portion $p = k-m$, hence defining the heuristic generalization loss on p, denoted $\hat{R}_p(h_\mathcal{A})$.*

The two heuristic definition however, can be noted to not be equal to the true generalization risk - empirical risk pair in reality. Nevertheless, one can then define overfitting and underfitting in such manner.

Definition 4.7 (Underfitting). *A model h is called **underfitting** if for a constant $\epsilon > 0$ chosen for the setting, then $\hat{R}_k(h) > \epsilon$, $\hat{R}_p(h_\alpha) > \epsilon$ over the entire observation space.*

Usually, underfitting is defined to be such that test error is smaller than training error partition, however, we do not think that is a good definition. Especially since, at least in this setting, the empirical error under algorithm $\mathcal{A}$ is reduced to the representative smallest error that $\mathcal{A}$ can achieve on that particular model, within particular setting of h, then come the heuristic generalization error. Considering also that $k/m \geq 1/2$, that is, at least the training partition is half of the observation set, then only under very specific case will $\hat{R}_p(h_\alpha)$ be smaller than $\hat{R}_k(h)$, statistically speaking. Thence this line of thought also reinforces the idea that both of them will be larger than the threshold ϵ. Usually, heuristically, this threshold of error is 0.3, normalized. Next, we define the notion of overfitting.

Definition 4.8 (Overfitting). *A model h is called **overfitting** if there exists a specific constant $\epsilon > 0$ chosen for the setting, such that $\hat{R}_k(h) < \epsilon$, $\hat{R}_p(h_\alpha) > \epsilon$ over the entire observation space.*

Namely, this mean that the static learning makes the generalization error worse. This draws the analogy of the student - studying explicitly for the past paper tests, yet unaware of the larger margin of questions in the large set of the test paper pool, leading to subpar performance. Similarly, overfitting is theorized to be fitting too fit to the reduced observation space - hence when it comes to the generalization observation space, it fails within such heuristic. The notion works on the mask of the risks, $\hat{R}_k$ and $\hat{R}_p(h_\alpha)$.

5 Bias-variance tradeoff

Bias-variance tradeoff comes off from a statistical approach, predate machine learning by itself. The theory that bias and variance often come in to conjunction is **Estimation theory**, of which there exists an estimator $\hat{\theta}$ that use a set of observations, to estimate the concept, or the underlying mechanics of certain system that outputted the observation set, by the **probabilistic perspective** - that is, it is governed by appearance by an independently and identically distributed process of parameterized probability $\theta \in \Theta$. Here, parameterized means being expressed by a set, often finite, of parameters, by E. L. Lehmann [1998], Paninski [2005].

5.1 Statistical origin

As we have said, the notion comes from statistical analysis. In such, there are two important functions associated with any estimator $\hat{\theta}$ that are useful as a thumbnail sketch on how well the estimator is doing (Paninski [2005]), the *bias* and the *variance*. Classically, estimation theory is concerned of the problem of data sample estimation on continuous real line $\mathbb{R}$. As such, the ultimate goal of classical estimation theory is the minimization of the estimator mean square error (MSE), that is given as

$$\mathsf{MSE}(\hat{\theta}) \triangleq \mathbb{E}_y ||\hat{\theta} - \theta||^2 = \mathbb{E}_y ||z(y) - \theta||^2 \tag{12}$$

where y is the observation set by notation, and $\hat{\theta} = z(y)$ is the estimator of interest. Then, the bias and variance is defined to be

$$\mathsf{Bs}(\hat{\theta}) = ||\mathbb{E}_y\{\hat{\theta} - \theta\}||, \quad \mathsf{Var}(\hat{\theta}) = \mathbb{E}_y ||\hat{\theta} - \mathbb{E}_y\{\hat{\theta}\}||^2 \tag{13}$$

This prompted a tradeoff, in which for the mean squared error the **minimum mean squared error** (MMSE) estimator would consider the bias and variance for every value of the parameterized descriptor θ. The best strategy potentially can be employed is then to reduce the variance of the model. This is rationalized as because for estimation theory in general, the bias term is unrealizable because it depends on the measure of the true concept. Based on the probabilistic interpretation, the true concept, or its true parameters governing the probabilistic sampling process cannot be learned, and hence the term bias can be at best approximated, with varied degree of accuracy. The limitation (Piera and Javier [2005], Kay [1993]) suggests to focus uniquely on unbiased estimators holding that $\mathsf{Bs}(\hat{\theta}) = 0$. Thus, the estimator mean square measure is equal to its variance, and the resulting estimator is then referred to as the minimum variance unbiased (MVU) estimator by Kay [1993].

In modern literature, the analysis of machine learning model and the introduction of such bias-variance concept for model selection criterion first came from Geman et al. [1992]. Of a later date, there are many interpretations, and similar notions of such tradeoff comes from both statistical learning theory and his original paper, which is used extensively among modern machine learning practicians.

5.2 Precursor (Geman et al., 1992)

The original definition of bias-variance tradeoff by Geman et al. [1992] is first constructed using the means-square error, which is regarded as a normal measure in the real encoding space. Their approach is to justify bias-variance via decomposition of the loss function ℓ, for such to find an alternative reasonable form of such loss landscape. Suppose of a regression problem to construct a hypothesis function $f(x)$ from $(x_1, y_1, \ldots, x_N, y_N)$ for the purpose of generalization - that is, predicting unseen variational values for different pair $(x_j, ?)$ such that $? = y_j + \epsilon$ for a conceivable implicit error. To be explicit about the relation of this problem, or f on the given data $\mathcal{D} = \{(x_i, y_i) \mid i \leq N\}$, denote $f(x; \mathcal{D})$ instead of f, the natural mean-square measure as a predictor is:

$$\mathcal{M}(f, y) = \mathbb{E}\left[((y - f(x; \mathcal{D})))^2 \mid x, \mathcal{D}\right] \tag{14}$$

for $\mathbb{E}[\cdot]$ the expectation wrt to a distribution P. Decomposing the right-hand side, we have:

$$\mathcal{M}(f,y) = \mathbb{E}\left[((y - f(x;\mathcal{D})))^2 \mid x, \mathcal{D}\right] = \mathbb{E}\left[(y - \mathbb{E}[y \mid x])^2 \mid x, \mathcal{D}\right] + (f(x;\mathcal{D}) - \mathbb{E}[y \mid x])^2 \tag{15}$$

Here, $\mathbb{E}\left[(y - \mathbb{E}[y \mid x])^2 \mid x, \mathcal{D}\right]$ does not depend on $\mathcal{D}$, but simply the statistical variance of y given x. The term $(f(x;\mathcal{D}) - \mathbb{E}[y \mid x])^2$ is considered a natural measure of effectiveness on $\mathbb{R}^n$ as a singular predictor of y. Now, for $\mathbb{E}_{\mathcal{D}}\left[(f(x;\mathcal{D}) - \mathbb{E}[y \mid x])^2\right]$ which depends on the training set $\mathcal{D}$ in its computation, is decomposed into the form of *bias-variance decomposition* terms, by derivation:

$$\mathbb{E}_{\mathcal{D}}\left[(f(x;\mathcal{D}) - \mathbb{E}[y \mid x])^2\right] = \underbrace{\left\{\mathbb{E}_{\mathcal{D}}[f(x;\mathcal{D})] - \mathbb{E}[y \mid x]\right\}^2}_{\text{bias term}} + \underbrace{\mathbb{E}_{\mathcal{D}}\left\{(f(x;\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[f(x;\mathcal{D})])^2\right\}}_{\text{variance term}} \tag{16}$$

We summarize this in the following statement.

Theorem 5.1 (Bias-variance decomposition). *Suppose the model $f(x;\mathcal{D})$ for the data $\mathcal{D} = (x_i, y_i)$ and its parameter x is defined. For y_i of the target concept's responses y, and consider a regression problem with the loss measure $\mathcal{M}(f,y)$ of mean squared risk, the following statement is true:*

$$\mathbb{E}\big[\mathcal{M}(f,y)\big] = \mathcal{B}(f,y) + \mathcal{V}(f,y) + \mathbb{E}\Big[\mathbb{E}\left[((y - f(x;\mathcal{D})))^2 \mid x, \mathcal{D}\right]\Big] \tag{17}$$

for $\mathbb{E}[\cdot \mid x, \mathcal{D}]$ any expression with dependencies on x and $\mathcal{D}$. The bias and variance term is subsequently expressed by

$$\mathcal{B}(f,y) = \underbrace{\left\{\mathbb{E}_{\mathcal{D}}[f(x;\mathcal{D})] - \mathbb{E}[y \mid x]\right\}^2}_{\textit{bias}}, \quad \mathcal{V}(f,y) = \underbrace{\mathbb{E}_{\mathcal{D}}\left\{(f(x;\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[f(x;\mathcal{D})])^2\right\}}_{\textit{variance}} \tag{18}$$

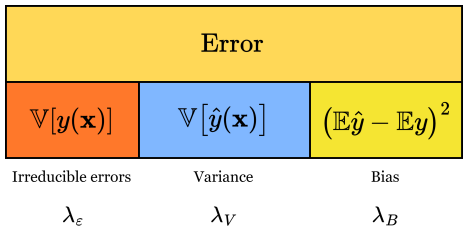


Figure 4: **Decomposition of the error term into 3 parts.** Respectively, the irreducible error $\mathbb{V}[y(\mathbf{x})]$, the variance (blue) and the bias (dark yellow). All of them are assumed to take up 100% of the error observed in such composition. The proportion is included using the three coefficient $\lambda_\epsilon, \lambda_V, \lambda_B$ where $\lambda_\epsilon + \lambda_V + \lambda_B = 1$.

The above decomposition principle is often expressed into a form where there exists the intrinsic noise Brown and Ali [2024]:

$$\begin{aligned}\mathbb{E}_D\left[\mathbb{E}_{xy}\left(y - \hat{f}(x)\right)^2\right] = \mathbb{E}_x\left[\left(y^* - \mathbb{E}_D[\hat{f}(x)]\right)^2\right] + \mathbb{E}_x\left[\mathbb{E}_D\left(\hat{f}(x) - \mathbb{E}_D[\hat{f}(x)]\right)^2\right] \\ + \mathbb{E}_{xy}\left[(y - y^*)^2\right]\end{aligned} \tag{19}$$

For simplicity of notation, we adopt the similar form in the standard case of Adlam and Pennington [2020]:

$$\mathbb{E}\left[\hat{y}(\mathbf{x}) - y(\mathbf{x})\right]^2 = \left(\mathbb{E}\hat{y}(\mathbf{x}) - \mathbb{E}y(\mathbf{x})\right)^2 + \mathbb{V}\left[\hat{y}(\mathbf{x})\right] + \mathbb{V}[y(\mathbf{x})] \tag{20}$$

A main common theme of criticism toward bias-variance tradeoff is the fact that the decomposition is much more general, and intrinsic for the class of *mean squared loss*. However, when considering the naturalness of an error measure and then its direct applicant, mean square error on real space of sufficient support and measure comes of as a very natural choice of an error measure, at least in consideration of the regression setting; for that, we then can also define classification as another top-layer above a regression's similar real continuous space, i.e. a decision layer. Furthermore, it can also be shown Brown and Ali [2024], Pfau [2013] that it also holds for the class of Bregman divergence measure.

In general, bias-variance is typically presented. In fact, one of the reason that it became the rule-of-thumb for ML practitioner, as well as generally statistical learning (Lafon and Thomas [2024] provides a quite rigorous treatment of bias-variance tradeoff in the section on statistical learning theory) solidify the trade-off as a particular model selection principle. Generally, this tradeoff can be summarized as followed:

Theorem 5.2 (Bias-variance tradeoff). *For the expected loss of any given hypothesis h, the bias $\mathcal{B}(f, y)$ and variance $\mathcal{V}(f, y)$ is inversely proportional, that is, $\mathcal{B}(f, y) \propto \lambda^{-1}\mathcal{V}(f, y)$ for some proportionality λ that may or may not be constant. In the most general case possible, $\lambda = -1$ on the entire error range.*

The tradeoff is then of inverse proportionality. Indeed, statistically, we have such tradeoff on a statistical framework in a more concrete sense. For the bias to increase, variance will increase, of which the criterion is inverse - we would like to have more bias but lower variance, according to such theory.

There are problems regarding such stance. The main problem is that generally, the bias-variance measure does not totally match the overall decomposition structure. Even with irreducible errors, there are factors which make it to incorporate other error factors in for the decomposition, of which is inexplainable. There are also works, for example, Domingos [2000b], of which decompose the error to

$$\begin{aligned}\text{Error} &= \lambda_1 \text{Bias}(f, y) + \lambda_2 \text{Var}(f, y) + \lambda_3 \epsilon(\mathcal{D}) \\ &= \lambda_1 \underbrace{\{\mathbb{E}_{\mathcal{D}}[f(x; \mathcal{D})], \mathbb{E}[y \mid x]\}}_{\text{bias term}} + \lambda_2 \underbrace{\mathbb{E}_{\mathcal{D}}\left\{(f(x; \mathcal{D}), \mathbb{E}_{\mathcal{D}}[f(x; \mathcal{D})])\right\}}_{\text{variance term}} + \underbrace{\lambda_3 \epsilon}_{\text{irreducible error}}\end{aligned}$$

of which the bias-variance-irreducible error are 'fit' into the total error by three coefficients $\lambda_1, \lambda_2, \lambda_3$ instead. This does not only scale up the B-V-IE triplet, but also leaves out the inexplainable gap between the scaling, and the actual normal measure.

The bias-variance decomposition and tradeoff is not general. Indeed, works, by Geman et al. [1992], Sharma and Aiken [2014], Domingos [2000b], Adlam and Pennington [2020], Yang et al. [2020] and more all attempted to reconstruct and reformulate bias-variance, and it did leave out a picture of uncertainty regarding the concept.

5.3 Statistical learning theory perspective

By statistical learning theory, we observed that there are plenty tradeoffs, most notably the empirical-generalization tradeoff, though we would have to clarify what those terms are. Furthermore, there is also what can be seen to be the analogue bias-variance tradeoff in such sense, is the *approximation-estimation tradeoff* of statistical learning theory, though it must be clarified that they are strictly not the same, as seen from Brown and Ali [2024].

Informally, the empirical-generalization tradeoff can be said to be the balancing act between achieving the true generalization and the empirical best. To achieve the generalization best, the empirical best, for finite hypothesis $\mathcal{H}$ (Mohri et al. [2012]),

$$\lim_{n\to\infty}\left[\mathop{\mathbb{E}}_{\mathcal{S}_n\sim\mathcal{D}^m}(\hat{R}_{\mathcal{S}(h)})\right] = R(h), \quad h\in\mathcal{H} \tag{21}$$

This follows by the law of large number, and the statistical-probabilistic interpretation of the learning setting. However, the tradeoff is not straightforward - it is rather considered between, for example, the time-complexity, sample-space size, and generalization best's reach of the model. That is, for large the dataset is, the easier it is to get a sample space that represents more and more of the actual concept. However, in exchange, time-complexity increases, and thereby, the ergonomic aspect must be also considered. It is also concerned of the fact that the empirical best is not the same as even the generalization best, thereby, one will want their hypothesis class to extend. By doing this, they extend the outreach of the hypothesis class, however make it unstable under random, iterative process typically used to train such model to target.

This is also noticed and formulated in the concept of approximation-estimation tradeoff. We refer to Mohri et al. [2012], Lafon and Thomas [2024] for some analysis and mentions.

Let $\mathcal{H}$ be a family of functions mapping $\mathcal{X}\to\{1,-1\}$. This is the particular case of **binary classification**, in which can be straightforwardly extended to different tasks and loss functions. The *excess error* of a hypothesis $h\in\mathcal{H}$, is the difference between its error $R(h)$ and the Bayes error R^*. This can be decomposed to be the following:

$$R(h)-R^* = \Big(R(h)-\inf_{h\in\mathcal{H}}R(h)\Big)+\Big(\inf_{h\in\mathcal{H}}R(h)-R^*\Big) \tag{22}$$

The first bracket contains the **estimation error**, and the second bracket contains what is called the **approximation error**. The estimation error depends on the hypothesis h selected. It measures the error of h with respect to the infimum of the error achieved by hypotheses in $\mathcal{H}$, or that of the best-in-class hypothesis h^* when that infimum is reached. The approximation error measures how well the Bayes error can be approximated using $\mathcal{H}$. It is a property of the hypothesis set $\mathcal{H}$, a measure of its richness.

5.4 Gradient descent on bias-variance decomposition

While the interpretation of what the bias-variance tradeoff, the usual training-versus-test curve, and statistical learning consideration, another interesting aspect to consider (as well for double descent), is how bias-variance decomposition acts on gradient descent and its components. This claim is perhaps reinforced by the work of Adlam and Pennington [2020] which indicates that different bias-variance decompositions effects differently on the total system mechanism.

The main term of a gradient descent algorithm is the gradient of the loss function $\mathcal{L}(h(x),y)$. For the hypothesis h, the number of instances supplied denoted by k has three main cases: $k=1$ for stochastic descent, $k\in(1,m)$ for batch gradient descent, and $k=m$ for standard gradient descent, for a given space of m samples. The gradient in such case is calculated using the empirical risk on m size, that is:

$$\mathcal{L}_k(h(x),y) = \hat{R}_{S[k]}(h) = \frac{1}{k}\sum_{i=1}^{k}\ell(h(x_i),y_i) \tag{23}$$

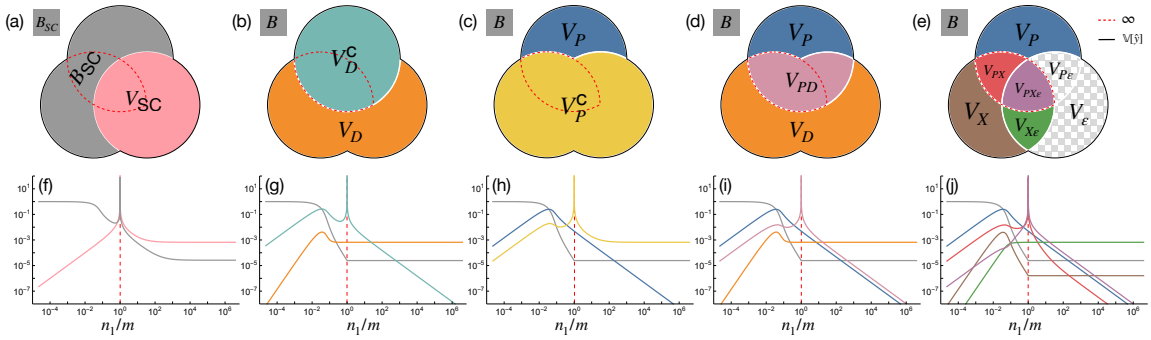


Figure 5: **(a–e) The different bias-variance decompositions.** (f–j) Corresponding theoretical predictions for $\gamma = 0$, $\phi = 1/16$ and $\sigma = \tanh$ with SNR $= 100$ as the model capacity varies across the interpolation threshold (dashed red). (a,f) The semi-classical decomposition of Hastie et al. [2019], Mei and Montanari [2019] has a nonmonotonic and divergent bias term, conflicting with standard definitions of the bias. (b,g) The decomposition of Neal et al. [2018] utilizing the law of total variance interprets the diverging term V_D^{C} as "variance due to optimization". (c,h) An alternative application of the law of total variance suggests the opposite, *i.e.* the diverging term V_P^{C} comes from "variance due to sampling". (d,i) A bivariate symmetric decomposition of the variance resolves this ambiguity and shows that the diverging term is actually V_{PD}, *i.e.* "the variance explained by the parameters and data together beyond what they explain individually." (e,j) A trivariate symmetric decomposition reveals that the divergence comes from two terms, V_{PX} and $V_{PX\varepsilon}$ (outlined in dashed red), and shows that label noise exacerbates but does not cause double descent. Since $V_\varepsilon = V_{P\varepsilon} = 0$, they are not shown in (j). Taken from Adlam and Pennington [2020]

We use the classical bias-variance tradeoff at first. Since the gradient is a linear operator, we have:

$$\begin{aligned}\nabla_k(\mathcal{L}) &= \nabla_k \left((\mathbb{E}\hat{y}(\mathbf{x}) - \mathbb{E}y(\mathbf{x}))^2 + \mathbb{V}\left[\hat{y}(\mathbf{x})\right] + \mathbb{V}[y(\mathbf{x})] \right) \\ &= \nabla_k \left(\mathbb{E}\left[\hat{y}(\mathbf{x})\right] - \mathbb{E}\left[y(\mathbf{x})\right] \right)^2 + \nabla_k \mathbb{V}\left[\hat{y}(\mathbf{x})\right] + \nabla_k \mathbb{V}\left[y(\mathbf{x})\right]\end{aligned} \tag{24}$$

This effectively decompose the gradient into three parts that influence the overall gradient calculation. Because $\nabla_k \mathbb{V}\left[y(\mathbf{x})\right]$ calculates the gradient of irreducible error, which is supposed to be constant of the sample space, it is then equal 0. Suppose a sample $\mathbf{x}_k$ of size k. Then, the loss function calculates:

6 Double descent

Nevertheless, of bias-variance and the analogous statistical learning theory concept, the target is the same. It is the dilemma of which is presented in **Occam's razor**, for choosing the sufficient model of good complexity, or bias, for tradeoff of its generalization ability, or variance. Then there must exist a sweet spot between the axis of bias and variance, since they are as exhibited above inversely proportional to each other. However, double descent seemingly broke the status quo, and insists on an interesting phenomenon - under the same setting, if we 'crank' the complexity high enough, we will then reach a point then called the **interpolation threshold**, such that the trend reverse and the error rate, instead of being theorized to go up, goes down to a certain line of lower bound.

The first identification of the double descent phenomena dated back to the paper of Belkin - Belkin et al. [2019], in which the title is literally "reconciling" modern machine learning practice and the bias-variance tradeoff. In modern machine learning practice, or state-of-the-art developments, models are now bigger than ever. If to notice, we will see that currently models are inherently large, for example, a normal large language model will have from 900 millions (900M) to a few billions, for example 10 billions (10B) parameters. That is not taking into account the overall dynamics and structure of the model, which dictates the operating range and efficiency of the model itself. These model, based on the neural network architecture are somewhat trained to exactly fit (or interpolate) the data, almost certainly so that it turn from a prediction setting to an estimation setting. By statistical learning theory, this would be considered overfitting, and yet, they often obtain very high accuracy on test data.

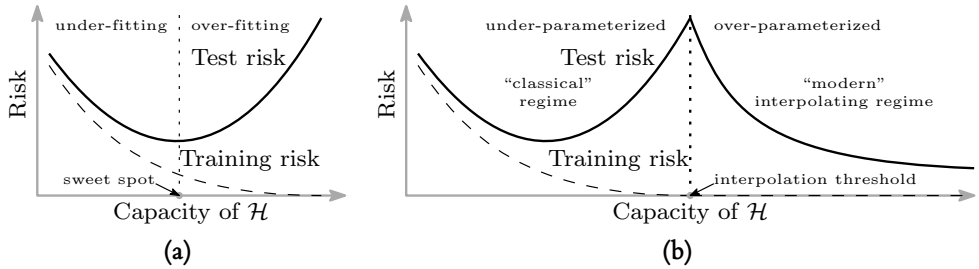


Figure 6: **Curves for training risk (dashed line) and test risk (solid line).** (**a**) The classical *U-shaped risk curve* arising from the bias-variance trade-off. (**b**) The *double descent risk curve*, which incorporates the U-shaped risk curve (i.e., the "classical" regime) together with the observed behaviour from using high capacity function classes (i.e., the "modern" interpolating regime), separated by the interpolation threshold. The predictors to the right of the interpolation threshold have zero training risk. Reproduced from Belkin et al. [2019].

The main finding that Belkin found is a pattern for how the apparent performance on unseen data depends on model capacity and the mechanism underlying the emergence of double descent. When function class capacity is below the "interpolation threshold", learned predictors exhibit the classical U-shaped curve from Figure 6. The 'modern' interpolating regime marks the opposite trend to the right, where the risk starts to decrease up to a lower bound, which then can be called the *optimal descent bound*.

Another prominent result to look at is Nakkiran et al. [2019], on the double descent of deep learning models. This is the first step toward identifying double descent to be perhaps, universal.

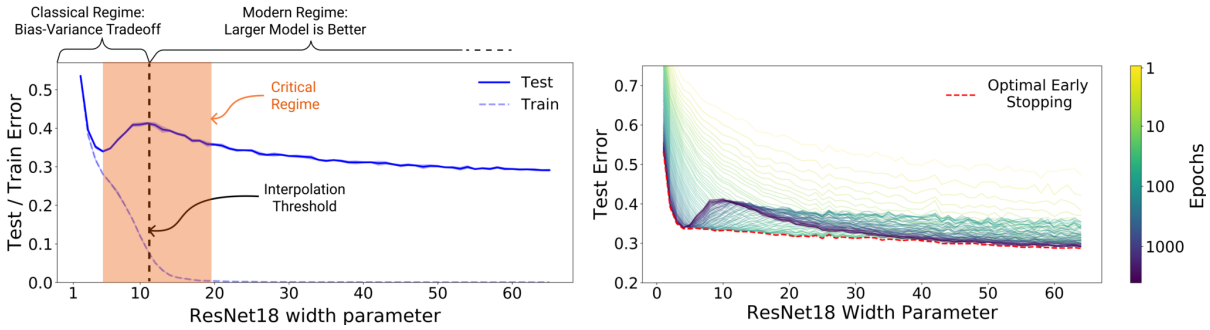


Figure 7: **Left**: Train and test error as a function of model size, for ResNet18s of varying width on CIFAR-10 with 15% label noise. **Right**: Test error, shown for varying train epochs. All models trained using Adam for 4K epochs. The largest model (width 64) corresponds to standard ResNet18. Resued from Nakkiran et al. [2019].

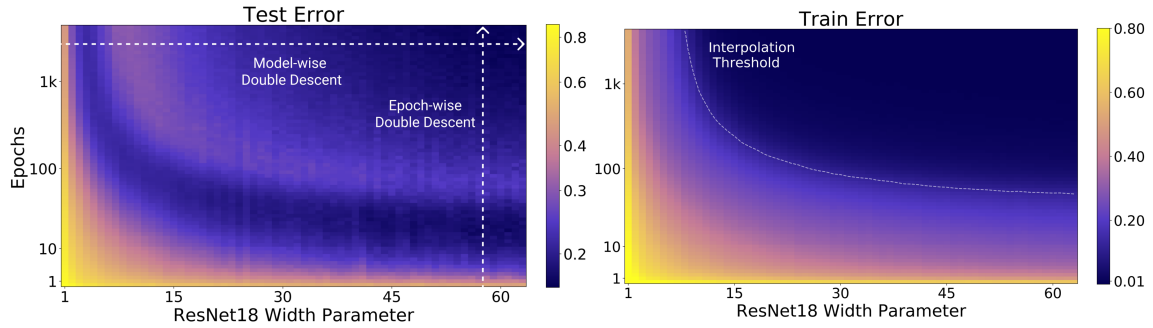


Figure 8: **Left**: Test error as a function of model size and train epochs. The horizontal line corresponds to model-wise double descent–varying model size while training for as long as possible. The vertical line corresponds to epoch-wise double descent, with test error undergoing double-descent as train time increases. **Right** Train error of the corresponding models. All models are Resnet18s trained on CIFAR-10 with 15% label noise, data-augmentation, and Adam for up to 4K epochs. Reused from Nakkiran et al. [2019]

They define *effective model complexity* of $\mathcal{T}$ (w.r.t. distribution $\mathcal{D}$) to be the maximum number of samples n on which $\mathcal{T}$ achieves on average ≈ 0 *training error*. This is an entirely empirical definition, similarly per definition as VC-dimension.

Definition 6.1 (Effective Model Complexity). *The* Effective Model Complexity *(EMC) of a training procedure $\mathcal{T}$, with respect to distribution $\mathcal{D}$ and parameter $\epsilon > 0$, is defined as:*

$$\mathrm{EMC}_{\mathcal{D},\epsilon}(\mathcal{T}) := \max\{n \mid \mathbb{E}_{S\sim\mathcal{D}^n}[\mathrm{Error}_S(\mathcal{T}(S))] \leq \epsilon\}$$

where $\mathrm{Error}_S(M)$ is the mean error of model M on train samples S.

Using this definition, their main hypothesis can then be stated as the following three-fold regions:

Hypothesis 6.1 (Generalized Double Descent hypothesis, informal). *For any natural data distribution $\mathcal{D}$, neural-network-based training procedure $\mathcal{T}$, and small $\epsilon > 0$, if we consider the task of predicting labels based on n samples from $\mathcal{D}$ then:*

Under-parameterized regime. *If $\mathrm{EMC}_{\mathcal{D},\epsilon}(\mathcal{T})$ is sufficiently smaller than n, any perturbation of $\mathcal{T}$ that increases its effective complexity will decrease the test error.*

Over-parameterized regime. *If $\mathrm{EMC}_{\mathcal{D},\epsilon}(\mathcal{T})$ is sufficiently larger than n, any perturbation of $\mathcal{T}$ that increases its effective complexity will decrease the test error.*

Critically parameterized regime. *If $\mathrm{EMC}_{\mathcal{D},\epsilon}(\mathcal{T}) \approx n$, then a perturbation of $\mathcal{T}$ that increases its effective complexity might decrease* **or increase** *the test error.*

It is to notice that this definition is also only observational. By that, it only outlines the specific region of interest where the paradigm shift is identified through experimental results. It has no effective predictive power, or rather descriptive power more than setting up hypothesis with respect to the effective model complexity, and some arbitrary perturbation. Nakkiran et al. [2019] also noticed this difficulty in providing a theoretical definition and theorem regarding such hypothesis, as said in their manuscript.

The behaviour itself has been particularly investigated, in certain settings. For example, before Belkin et al. [2019], Advani and Saxe [2017] investigated the generalization error in neural networks of high-dimensional measure. Of such, various behaviours that bear similarity to double descent can be observed. Belkin et al. [2018] expanded on his previous research on the particular subset of kernel learning models, providing a theoretical analysis of specifically the Laplacian kernel on standard neural network. Mei and Montanari [2020] investigated random feature regression in such regard, and also found similar results.

7 Theoretical analysis

We will attempt to analyse double descent in terms of classical, non-novel theory. First, the issue of bias-variance and double descent is related to multiple factors, but specifically between the notion of bias-variance and complexity-accuracy. However, their definition and usage in certain contexts are obscured, thus would require a fairly thorough consideration. Most of these results contains of bounds and often discussion of random processes, and also restricted itself to mathematically well-defined setting. For now, we would like to investigate their notions and applications. Though for starter, we have to discuss the following preliminary. The hypothesis class $\mathcal{H}$ has two types of classification: either infinite hypothesis class $\mathcal{H}^\infty$ or finite hypothesis class $\mathcal{H}^n$. Based on the details of the expression of representation strings (Kearns and Vazirani [1994]), an infinite hypothesis class contains infinitely many specifiers of the hypothesis structure. Note that this does not correlate to an increase in parameter, as we will see with the case of axis-aligned rectangle - for only four parameters (l_1, l_2, h_1, h_2) we can specify all axis-aligned rectangles with their value taken from the field $\mathbb{F} = \mathbb{R}$. Finite hypothesis class is then can be understood of the number of configuration, or in discrete form combinatorial pairs that can be presented of the hypothesis class, for example, a string of all n-length binary numbers $\{x_1, \ldots, x_n\}, x_i \in \{0, 1\}$. A subtlety that would then have to discover, is the increment in size of the hypothesis to accommodate either finite or infinite number of specifying parameters - one example can be sampled from is the class of all support vector machine (SVM, Vapnik [1999]), which for now we treat it as it is. The concept class can then also be treated similarly. Here, we assume a setting where unless specified, we have knowledge of the concept class structures, but not the hypothesis.

7.1 Complexity measures

Assume the working space is the real field $\mathbb{R}^n$. Let us recall a model $h \in \mathcal{H}$ of certain hypothesis class $\mathcal{H}$. Then, the **complexity** refers to the complexity of the hypothesis h itself; there is also the notion of class maximal complexity for $\mathcal{H}$. Complexity in this case refers to the mass of the model, or rather, to answer the following questions:

1. What is the effective expression range of a hypothesis?
2. What is the hypothesis structured in?
3. If the model is constructed by parameters, how many parameters are there?
4. How many components are there in specific model h, and how would another model $h' \in \mathcal{H}$ differs from h?

Those questions are not so easily answered. Specifically, analysis often puts the hypothesis class $\mathcal{H}$ as a class of functions. So, there exists the class $\mathcal{H}_n$ of n-th degree polynomials, or else. Effective complexity in such case means the range of expression a function can give - for example, for n-th polynomial class, we already have the Weierstrass's theorem that tells us that any continuous function of closed interval can be approximated by a polynomial of arbitrary order. These expressions depend on the field or space the hypothesis class lives in, for example, if the hypothesis is configured to work in the set of all functions in $\mathbb{F} = \mathbb{R}$ of algebraic field, then we call it the algebraic family. There are various hypothesis classes in such case, and usually, not a lot of them can be analysed and represented in the same way, and hence also the analysis of the 'size' of the hypothesis, also between different hypothesis classes. This prompted the usage of a more general notion, called **representation complexity**. Representation complexity refers to the number of components needed to express h and generally, any hypothesis in $\mathcal{H}$. This is often the parameters count, since the parameters themselves are used in constructing the function in algebraic system. Thus, again, an n-th degree polynomial might have $2(n+1)$ parameters, for $n+1$ variational input (or variable), and $n+1$ coefficients. Notice that we specifically at the beginning, use the notion of algebraic

structure $\mathbb{R}$ because of it. If we restrict ourselves to the expression, for example, for $\mathbb{B} = \{0, 1\}$, then it can be expressed in various ways in logic theory, such as conjunctive normal form (CNF), disjunctive normal form (DNF), and else. This forbid the analysis based on the effective process itself, but rather the number of components required, hence taking them as complexity of construction.

Furthermore, the above two measures only talk about the **general mass** (representation) and **general expression**. In a deeper analysis, we haven't talked about the general individual complexity of the model itself. This notion refers to the complexity computed or evaluated, by considering the process-dependent behaviours and range of a specific expression influence on the operational process itself. While expressive complexity can provide a sufficient bound of approximation strength, the individual strength that can make it happen (for example, the fact that n^k value range versus n^{k-1} value range is totally different by one power degree), the contribution and sum of individual components can only be discussed using the general individual complexity, by using **process complexity** term. Up until now, this has not been formulated as author's knowledge, hence the two standard term would then be used instead in the analysis. The above complexity, aside from sample space-related measure, can also be applied onto the concept

Measure	Purpose	Note
Representation complexity	Measuring the construction complexity — complexity of representing a model	Taken over the alphabet used by the model
Expressive complexity	Measuring the operational complexity of the model — the outer process observation	Taken of the entire model, belongs to the model measure
Structural complexity	Similar to expressive complexity, but measures the inner structure of the model — which is then decomposed into complexity measures of individual components	Belongs to the model measure, taken over the entire model, decomposable
Structural mass	Measures the total representation size of a given model, using representation complexity and representation space to aggregate	Taken over the entire model, belongs to the model measure
Sample complexity	The observation space complexity — number of sample points given (cardinality)	Belongs to the process measure, taken over the sample space
Sample space complexity	The complexity of the samples, including detailed and aggregated information about the sample space	Statistical-like measure, taken over the sample space
Optimization complexity	The complexity of the learning optimization algorithm	Taken over the learning process, very hard to define, hard to draw out conclusive relations

Table 1: **Description of various complexity measure** that can be taken over a given model, and the specific learning process.

class's members itself. We also take the class complexity in such case, as the argument that maximize those measure: that is, the largest value of the measure, taken on the class as the class's complexity.

The next complexity measures that can be taken of, comes from the learning process structure itself. Usually, in statistical learning theory (Hajek and Raginsky [2021], Mohri et al. [2012], Shalev-Shwartz and Ben-David [2014]), we often use the setting that is related to **supervised process**. This usually means

the process can be configured in a feedback loop, or induced response toward the model - often come from the observational space, where the concept that is set to be the target of the process - then called as learning process - is conjured.

The learning process assumes an operational space containing the structure, for example, $\mathbb{R}$-encoding space of real values, and a set of **samples** living in such space, denoted $\mathcal{S}$. Those samples are often referred to as the **observation set** of a particular objective of learning. As mentioned above, $\mathcal{S}$ belongs to the concept c, assumed to be of certain concept class $\mathcal{C}$. $\mathcal{S}$ is obtained by using an observing process to create or generate the result. This process can be determined or considered in multiple way, however, generally can be considered by having a deterministic interpretation (using an explicit function $c(x)$) or by probabilistic approach (modelling a distribution $\mathcal{D}$). Hence, we gain another type of complexity, with regard to the process called **sample complexity** - usually defined of cardinality - the amount of sample needed for particular purpose, for particular objective. For the complexity category up to the structure of the learning setting by itself, we call it the **sample space complexity**.

One final complexity that is worth mentioning is then the **optimization complexity**, measure on the action of the learning algorithm. By such, it investigates the construction of an optimization algorithm solely dependent on its size of the algorithm, and other non-process related measures. Optimization complexity can also be related or expressed, in situation of time complexity, which is typical in computing system.

7.2 Analysis

This part of the section will be focusing on analysing the classical complexity measures that are used in the past to validate and examine particular model for their complexity, given in various sense.

7.2.1 Structural complexity

7.3 Learning process

The second important aspect of the theoretical analysis is reserved for the relation of complexity to the *learning process*, or as it is called, the model selection phase. This is one of the main property that machine learning excels against traditional mathematical modelling, aside from the structuring principle of the model, is a way for a model to effectively 'learn' and improve on which explicit formulation is not required. One of the main contributing notion to gauging the learning process and performance of the model, is through the lens of error measures.

On the space of numerical (algebraic) encoding, for Euclidean real space, one of the measure that naturally arise is the **distance measure** between objects encoded into the space. The most natural distance measure of them all is then the L_2-norm. These distance measures affect how the learning process is proceeded, since they provide the disparity comparison with the output of both hypothesis and concept. Fix at L_2-norm, we also gain the typical bias-variance tradeoff, certainly from Geman et al. [1992]. Most of the learning process is taken into account using distance measures.

In the more discrete cases, the distance measure is reduced into discrete conceptual part, masked by a **categorization function** on top that measure the classification of discrete data points on another proximal criteria of categories. That is, for each $x \in \mathcal{R}^d$ of d-dimension representation space, there then exists such function $\mathcal{C}_{\text{cat}} : \mathcal{R}^d \to \{\phi_i\}$ for all $\phi_i \in \Phi$ of all categories. Then, any error measure on such space can be called **classification error measure**. Because of its delegate nature (categories masking), there then can be two types of error accompanied. Either taking into account of the discrete absolute categorization (for example, 1 for correct, 0 for wrong as binary classification, and varying 'score' on other multicategory sets), or the representation space error - factoring into the classifier's consideration of the input for misclassifying. Still, they would still act on a representation space of algebraic system, in practice.

Generally, the error measure varies between settings, models, and tasks. To identify them all requires a great effort to generalize all of such error measures, thus making it improbable of an effort. Nevertheless, we can still generalize all learning error measure, basing them on the space that they operate, and hence some of their intrinsic properties. In this theoretical analysis, we would also assume the static gradient descent-based algorithms for optimization. Such closed form solution like least square method can be used, especially in preliminary experiment, however, to base the analysis on a more practical ground, iterative optimization methods is used in majority instead.

Error measure directly, as was said, influences the learning process of which it is a component. Analytically, we define the *learning process*, by virtue of the action of learning, as followed.

Definition 7.1 (Learning process). *A procedure $\mathcal{P}$ is called a* ***learning process****, given two* ***constructs*** *or* ***model*** *$h \in \mathcal{H}$, $c \in \mathcal{C}$ if it follows an* ***error measure*** *or operator $\ell(\cdot, \cdot)$ such that, for $\mathcal{P}_i \in \mathcal{P}$ indicating the ith iteration of the procedure $\mathcal{P}$, h_i the ith hypothesis iteration of $\mathcal{P}$, minimize the operator that it chooses, that is,*

$$\exists n, i \in \mathbb{N}^*, n \geq i, \quad \ell(h_n, c) = \min_{h \in \mathcal{H}} \ell(h_i, c), \tag{25}$$

This definition then emphasizes learning process's error measure importance in the procedure $\mathcal{P}$. As we see, the standard learning process relies on some assumptions: that the iteration h_i is *stable*, $c \in \mathcal{C}$ is static, the procedure $\mathcal{P}$ provides adequate facilities that support such motion, and the hypothesis class $\mathcal{H}$ is also adequate to reach $\min_{h \in \mathcal{H}} \ell(h_i, c)$. In practical, this is impropable, as Mohri et al. [2012] indicated that in general the absolute minimum cannot be reach, then, we can change the condition such that there exists $\epsilon > 0$ such that

$$||\ell(h_n, c) - \min_{h \in \mathcal{H}} \ell(h_i, c)|| \leq \epsilon \tag{26}$$

It is also at this point that we begin to consider the meaning of the **probabilistic nature** of the learning setting. Where will probability fit in such scheme? Classically speaking, we believe that the *availability of knowledge*, or the assumption of missing information in the learning setting is detrimental to determine how 'probabilistic' a setting is.

7.3.1 PAC-learning procedure

The PAC-learning process (Shalev-Shwartz and Ben-David [2014], Mohri et al. [2012], Hajek and Raginsky [2021]) connects three complexity measure - the sample space complexity, representation complexity, and optimization complexity, to the evaluation of the statistical learning process. It refers to the categorization of various learning procedure that satisfies the Probably Approximately Correct criteria in its name, as such being followed.

We assume no structure of h and c. They can be functions, partial functions, relations, complex algorithms, or others. In a typically learning setting, we also have the argument of *preliminary knowledge*, presented in literatures of the term *inductive bias*. For example, if the concept class $\mathcal{C}$ is assumed, then we say the setting is **model-specific**. If there exists no hard assumption on $\mathcal{C}$, then the learning setting is said to be **model-free**. Then, the PAC-learning set for model-specific setting is defined as followed.

Definition 7.2 (PAC-learning). *A concept class $\mathcal{C}$ is said to be* ***PAC-learnable*** *if there exists an algorithm $\mathcal{A}$ and a polynomial function $poly(\cdot, \cdot, \cdot, \cdot)$ of 4-argument such that for any $\epsilon > 0$, $\delta > 0$, for all distribution $\mathcal{D}$ on $\mathcal{X}$ and for any target concept $c \in \mathcal{C}$, the following holds for any sample size $m \geq poly(1/\epsilon, 1/\delta, n, size(c))$:*

$$\Pr_{S \sim \mathcal{D}^m} [R(h(S)) \leq \epsilon] \geq 1 - \delta$$

for a given error measure $R(h_S)$. If $\mathcal{A}$ further runs in $poly(1/\epsilon, 1/\delta, n, size(c))$, then $\mathcal{C}$ is said to be efficiently PAC-learnable. When such algorithm exists, we call $\mathcal{A}$ a PAC-learning algorithm.

This can be easily extended into the model-free case, and further on, which is left in the appendix section. The reason for the argument $size(c)$ is as followed. Consider a class of concepts defined by the satisfying assignments of certain formulae. A concept from this class that satisfies such formulae, can be represented by a formula f, a truth table, or any given formulae that is tautologically equivalent formulae f' to f. PAC-learning bound argues in the sense of *computational complexities*, and *space complexities*. Specifically, the term $poly(1/\epsilon, 1/\delta, n, size(c))$ hopes to bound the required learning process to polynomial time, specified by 4 parameters. Here, n is the *input dimension*, $size(c)$ is the encoding complexity of the target concept, $\epsilon > 0$ is the *accuracy bound*, and δ is the confidence bound - the probability that h fails.

8 Problem

Preliminary sections and historical remark on the problem, both bias-variance and double descent hinted at a potential deadlock in resolving such it. Conflicting notions and reports aid into the confusion of such topic more than can be normally observed, and the anomalous behaviours in some researches (for example, in Shi et al. [2024], we received that in GNN, there exists no trace of double descent) makes it even harder. As such, both the status of bias-variance being false, albeit in a given range only, and double descent in masking certain behaviours of the modelling process, have taken a mysterious stance in the modern machine learning community.

While discovering the double descent phenomena, it also exposes theoretical concepts that are not organized or formalized, portions of the theory and empirical observations that are within conflict of each others, issues with definitions, theorems, and interpretation of them. Those problems would then ultimately hinder further research in such direction, as well as broader theoretical requirement of the theory of machine learning, and specifically to a larger and more complex system of deep learning.

8.1 Model complexity

The notion of model complexity can be said to be often not so well-defined. There are attempts by Hu et al. [2021], Luo et al. [2024], Barceló et al. [2020], Molnar et al. [2020], Janik and Witaszczyk [2021]. The problem of model complexity is not so apparent until the inherent complex nature of models, typically in deep learning models (based on the MLP architecture) is observed to cannot be treated in similar way as classical models. In Hu et al. [2021], most deep learning models are based on, and investigated of their complexity through measures like the expressive capacity, the effective capacity, and so on; aside from other teams and researchers investigated it by using computational complexity, or functional decomposition. One promising aspect of this is Molnar et al. [2020], where we are able to decompose the functional form a specific model into variations of its complexity.

Nevertheless, it is reasonable to say there are no conclusive measure or definition of a model's complexity. Considering that double descent also appears outside the range of deep learning model, and is somehow consistent in classical models (for example in models tested by Belkin et al. [2019]). This is particularly troublesome, as a theoretical analysis requires such definition, especially when the relation of interest is directly involved in such manner. This is a particular aspect needed to be resolved if there are to be progress made in this investigation. Additionally, the concept of overparameterization and underparameterization is also a problem, especially since its definition is not well-defined, hence become a problem in analysis by identifying the classification of the problem setting.

8.2 Model structure

In modern practice, the theoretical and formal treatment, rigorous consideration of different model structures and complexity is not realized. Partially, this is due to the bloom of machine learning and artificial intelligence from the early 2000s. Currently, there has been no conclusive theoretical definition or formulation about the structure of different models, typically can go under the name of mathematical modelling hypothesis, in a way that unify certain properties between architectures.

8.3 Setting consistency

The setting surrounding the learning theory, machine learning models is generally missing of its rigours and analytical properties. Most of the learning setting and model's training-testing setting are often poorly stated in theoretical sense, and there are a lot of diffusing details that might or might not affect the model's perturbation without knowing. This in a certain way leads to the complexity in analysing different problem setting and experimental results, since the architecture used, the setting considered, model in questions, configuration (either custom or on system side) are not consistent overall.

Furthermore, a lot of terms, definitions are often hand-waived in papers or in discussions. This also led to the point that in Nakkiran et al. [2019], they have to somehow 'reinvent' another type of model complexity itself, albeit unsatisfying of a definition, is still a new kind of definition per insufficiently of such. while it is welcome to introduce new definitions, the fact is that it is subjectively inconclusive of terms, definitions and thereof leads to the perception and situation where multiple studies can be found, but all of them are defined on different ground.

8.4 Classical analytical difficulty

Analysing statistical learning theory and overall landscape of statistical learning theory, and the learning theory as a whole, encountering new problems like double descent revealed its weakness, and ultimately, perhaps one of the reason why it is ineffective against such problem. First, there are simply too many assumptions made, too many formulations made during said process. Furthermore, there are also unclear notions and concepts, of which make it even harder to analyse or fully formalize. Secondly, there are inherent conflicts and uncertainty within those theories, formulations and notions by itself. For example, in one sense, the No-free-lunch theorem is considered to be representative and true, whilst also simultaneously being considered the opposite of such. And amidst abnormality behaviours of the old bias-variance formulation, we also find distinctive weakness in our theory, for example, the concerning difficulty in defining the notion of *model complexity* in various contextual ways. To analyse double descent, perhaps we also need a new theory or formulation to support it. [1]

[1] Furthermore, most of the general solution and bounds created by statistical learning theory is often in a very simplistic system. For example, if we are to utilize the Rademacher complexity measure (Mohri et al. [2012]), most of the time we will have to compute it through the growth function $\Pi_{\mathcal{H}}(m)$ for m points, on the standard finite hypothesis class $\mathcal{H}$ such that

$$\Pi_{\mathcal{H}}(m) = \max_{x_1,\dots,x_m \in \mathcal{X}} |\{(h(x_1),\dots,h(x_m)) \mid h \in \mathcal{H}\}| \tag{27}$$

Most of our problems resolve to binary classification, or rather, the problem of pattern recognizing discrete, reduced classification form. It is not so sure for now if all problems can be reduced to such way, so we cannot draw conclusive analysis that is not diluted of mathematical formulation for complex systems. That is not to count the computationally intensive operation required to calculate the supposedly classical measure, while not entirely of itself holds any substantial reasonable information about the internal dynamics of the model itself.

9 Hypothesis

In development of our theory, several hypotheses, most of the time observations regarded such, is formed. Hence, there are several hypotheses for what to look out for, as well as the formulation needed to interpret certain phenomena and observations. This ranges from topics of re-evaluating the statistical learning theory, estimation and refactor of assumptions (or *inductive bias* when designing models), to different interpretations and representations that can better explain, model, or gives certain insights.

9.1 Stability of bias-variance measure

Stability of bias–variance measure is questionable, as it is in literature (Domingos [2000b]), loosely defined by defining a 3-dimensional vector $\lambda = (\lambda_1, \lambda_2, \lambda_3)$ for such:

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}\left[d((f, \mathbb{E}[y \mid x])\right] &= \lambda_1 \mathrm{Bias}(f, y) + \lambda_2 \mathrm{Var}(f, y) + \lambda_3 \epsilon(\mathcal{D}) \\ &= \lambda_1 \underbrace{\{\mathbb{E}_{\mathcal{D}}[f(x; \mathcal{D})], \mathbb{E}[y \mid x]\}}_{\text{bias term}} + \lambda_2 \underbrace{\mathbb{E}_{\mathcal{D}}\left\{(f(x; \mathcal{D}), \mathbb{E}_{\mathcal{D}}[f(x; \mathcal{D})])\right\}}_{\text{variance term}} + \underbrace{\lambda_3 \epsilon}_{\text{irreducible error}}\end{aligned}$$

where $\epsilon(\mathcal{D})$ is the irreducible error of the system, depends (theoretically) on the intrinsic imperfection of the dataset $\mathcal{D}$. Assessment on stability of such decomposition is recommended.

9.2 Alternative measures

We believe that the bias–variance decomposition is a poor estimation and measure for controlling, validating and choosing model from, at least from the perspective of analysing missing details. Specifically, bias and variance term, as well as their supposed expression of decomposition in the standard loss measure of model effectiveness is not representative, nor expressive of the actual, underlying notion of *model complexity* $C(f)$ and its effectiveness. Furthermore, bias–variance decomposition is not straightforward of others type of loss functions, except only for some classes of loss function (loss on Bregman divergences and exponential class). This sentiment is shared in Brown and Ali [2024].

9.3 Inductive bias

Inductive bias obviously plays a role, as mentioned by some literature [2], though we don't know exactly how the inductive bias can be formed to play any role in the central dynamic of the model operating process. Certain assumption, for example, the *convexity of the loss function* might actually affect the model as it is, and introduce unwanted patterns in the following process.

9.4 Randomness and probabilistic setting

Random initialization of parameters, as well as the initial conditions of the model initialization process might also be taken into account. Obviously, this is not a new insight, and is rather a very old one, however, this approach still has varied potential for interpreting double descent. Scaling up/down this effect might result in new insight, as well as inspecting the uncertainty and diffusion patterns in large-scale model deployment, in which from an end-to-end perspective requires several space transformation operations in successions.

9.5 Knowledge masking

The very ambiguous idea or treatment of statistical learning, as well as machine learning, is the idea of masked. Or, paraphrased, *hidden information* regarding the underlying process of estimation. Specifically, for example, we tend to assume things to not be able to grasp - either the probability $p_X(c)$ of the concept class, the distribution $\mathcal{D}$ specified to such input distribution, or the concept class $\mathcal{C}$, Markov properties, etc. We would like to formalize this notion as soon and as such in the most direct way.

[2] Need to include some in here.

10 Probing experiment

For understanding and analysing double descent and bias-variance trade-off, and furthermore in later section on identifying double descent, we would like to use several test models specifically for exhibiting bias-variance, as well as testing hypothesis and forming theoretical conjectures. Specifically, we would like to use *polynomial regression model*, the standard architectural description of *support vector machine* (SVM), the vanilla *neural network*, and *neural network* (if able, recurrent neural network). Our goal is pretty conclusive.

10.1 Polynomial regression

For a polynomial model, the complexity measure is harder to grasp. If we treat it the same as the case for linear regression with white Gaussian noise above, we would run into the problem of flattening effect - polynomial explosion of contributing factors between nth and the ith factor, for $i < n$ is very prevalent. Thereby, we might define the complexity the other way.

Most of the examples often seen with bias-variance tradeoff is with the famous example of polynomial regression. Indeed, in the range of interpretable, observable model results, polynomial regression with expressive capacity increased is one of the more famous problem setting, which also lies in the regression learning task. Goodfellow et al. [2016] also used polynomial regression in his book to illustrate the problem of bias-variance tradeoff, and several textbooks, such as ISLR James et al. [2013].

Informally, polynomial regression considers the set of all hypotheses h of the polynomial hypothesis class $\mathcal{H}_p$. In the single, univariate case of the hypothesis representation, the polynomial $p_n(x) \in \mathcal{H}_p$ is expressed by:

$$p_n(x) = \sum_{i=0}^{n} c_i x^i$$

where c_i is the associated constant for each term. The bias term here is controllable, and is part of the polynomial as the mathematical formulation holds. As always, since it is a model, x argument cannot be controlled. The only degree of freedoms provided is the set $\{c_i\}$ of all constants. Hence, we can conceptualize this as always, as a bunch of unit processing embedded each unit, with a scaling factor by i, and control them by the weight. The form $p_n(x)$ in a polynomial regressor will often be $M[p(n, x)] = \mathbf{W} f(\mathbf{x})^\top$ where $f : \mathbb{R}^{n+1} \to \mathbb{R}^{n+1}$ function which scales by applying power per argument. As standard, we will use the typical stochastic gradient descent [3], though it is important to note why we have to use it.

Using the Ruffini-Horner's scheme, we can analyse the polynomial in the form of nested form:

$$P(x) = \sum_{i=0}^{n} c_i x^i = a_0 + x\big(a_1 + x\big(a_2 + \cdots + x\,(a_{n-1} + x\,a_n)\big)\big) \tag{28}$$

or recursively, as the set of recursive relations

$$\begin{cases} b_n & = a_n, \\ b_k & = a_k + x\,b_{k+1}, \quad k = n-1, n-2, \ldots, 0, \\ P(x) & = b_0. \end{cases} \tag{29}$$

[3] As for why it is not pure vanilla gradient descent, we notice that for a gradient descent algorithm in an informal setting, it is wise to notice that for any given configuration dataset space, the path itself is deterministic based on all the description of the dynamical system (hyperparameters like learning rates, the algorithm itself, the hypothesis's parameters and representation, and the data assumptions - for example, without white noise or not). Removing determinism can be done using stochastic gradient descent, even though for now a formal treatment of this 'stripping off deterministic behaviour' is not fully formulated.

Experimentally, we would likely have to present certain interpretation and configuration to the setting, depends on particular subject of interest. So far, this is most presentable using diagram.

Setting 10.1 (Polynomial testing). *We formulate the polynomial testing scenario as followed. Let the hypothesis of the class of all univariate, multi-dimension input* $\mathbf{x} \in \mathbb{R}^d$*, for* $h \in \mathcal{H}_{p(n)}$ *of all univariate polynomial of degree* n*. Let the concept* $c \in \mathcal{C}_{p(m)}$ *be also the class of all univariate polynomial of degree* m*. Assume that* h *does not know anything about the true shape of* c*, except for the fact that it is chosen for this purpose by external factor (testing scenario and designer). Consider the observational space* $\mathbb{O}$ *of* h*. Then, it will be controlled of* $\mathbb{O} = (\mathbb{U}, \mathbb{I})$ *where* $\mathbb{U}$ *is the* ***input space****, and* $\mathbb{I}$ *is the inferred space. For* $\mathbf{x} \in \mathbb{U} \subset \mathbb{R}^d$ *is governed by the sampling process within distribution of* $\mathcal{D}$*, and for the inferred space to be of the form* $c_i + \lambda\mathcal{N}(\cdot)$ *of any given noisy feature of certain formulation of scaling* λ*. If* $\mathcal{N}(\sigma, \mu)$ *then we say the inferred space is fully realized as the white noise inferred space. The goal is then to measure, for supposed* $\mathsf{ERM}(h, c)$ *of the learning process, and* $\mathsf{SRM}(h, c)$ *for learning process with minimal generalization capacity on* $R(h)$*, observe and examine the behaviour of the polynomial setting around the interpolation point and within limits of exhibiting double descent.*

The complexity measure on the surface (operational) would be the degree n, m of the hypothesis and the concept (respectively with the setting), the degree of noise (hence variations) and type of noise, the input space width for $\mathbf{x} \in \mathbb{R}^d$ - for it if being high dimensional means more flexibility - which is inherent of the data, and other hyperparameters typical of controlling standard behaviours for the supervisor's learning procedure. In the experimental session, we would also want to somewhat compare the noise fitting solution, to the best interpolation solutions set, to be able to see the interpolating shape after noise. This will determine $d(c, c')$.

10.2 Support Vector Machine (SVM)

Support vector machine, first formally originated from Vapnik [1999], is a model inherently, specifically defined for pattern recognition task, and binary classification via an *optimal separating hyperplane*. There are two variants for SVM, namely, for linear and nonlinear hyperplane.

The SV machine implements the following two precursor ideas: It maps the input vectors x, supposed of the setting, into a high-dimensional feature space Z through some nonlinear mapping, chosen a priori. In this space, an optimal separating hyperplane is constructed. By statistical learning theory, Vapnik restricted the function class (as for infinite hypothesis it is impossible to learn) to the class of hyperplanes by

$$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0; \quad \mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R} \tag{30}$$

which $\mathbf{w}$ is the controlling weight, $\mathbf{x}$ is the input space to the space of all binary $\{-1, +1\}$ category. Hence, this basically divide the input space into two: one part containing vectors of the class -1 and the others being $+1$. If there exists such a thing, then it is said to be *linearly separable*. To find the class of a particular vector $\mathbf{x}$, we use the following decision function

$$f(\mathbf{x}) = \operatorname{sgn}[\langle \mathbf{w} \cdot \mathbf{x} \rangle + b] \tag{31}$$

This is called the **hyperplane classifier** class. As can be understood simply from such, there exists many hyperplanes that can correctly classifies the classes. It has been then shown that the hyperplane that guarantees the best generalization problem performances is the one with the maximal margin of separation between two classes Cristianini and Shawe-Taylor [2000]. The above form of finding final classification can then be presented in dual form, which then depends only on dot products between vectors, as

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{\ell} y_i \alpha_i \langle \mathbf{x} \cdot \mathbf{x}_i \rangle + b\right) \tag{32}$$

where $\alpha_i \in \mathbb{R}$ is a real-valued variable that can be viewed as a *measure* of how much informational value $\mathbf{x}_i$ has (for x_i the support vectors we get from training)[4]

The second idea is of the *kernel method*. This particular method, useful and well-known of the time SVM was created, gain a more prominent position among other techniques, including neural network. Specifically, this method is characterized by the mapping of the input vectors into a richer (usually high-dimensional) feature space where they satisfy *linear separable* criteria. This prompted the possibility to solve nonlinear problem through such mapping, and indeed yields a nonlinear decision surface in the input space, which is linear in the feature space. A **kernel** (function) is then a function $k(\mathbf{x}, \mathbf{y})$ that given two vectors in input space, return the dot product of their images in feature space such that $k(\mathbf{x}, \mathbf{y}) = \langle \phi(x), \phi(y) \rangle$ for the nonlinear mapping ϕ. The general form of SVM is then

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{\ell} y_i \alpha_i k \langle \mathbf{x} \cdot \mathbf{x}_i \rangle \right) \tag{33}$$

for any particular final categorization. One of the major problem for analysing SVM, is in the properties of it potentially having infinitely many parameters, depends on the size of the dataset. This makes the curve of both bias-variance and double descent not applicable in such regard - infinitely many parameter complexity, yet still finite complexity class.

From such observation, it would seem imperative that the notion of complexity based solely on the parameter counts of certain model construction is ill-equipped for certain flavours of machine learning model, in this case support vector machine (in the same type is the Gaussian Mixture Model). No double descent has been observed by empirical reports record per the author's knowledge.

10.3 Neural network

One more class of elementary and classical experimental setting that would be considered is the class of typical neural network, or in general, Q-width L-depth neural network of general activation type, without specific engineering or modifications. For completeness, we refer to Zhang [2023] for a more mathematical analysis, and our network will be formulated the same. Generally, we define a K-layer fully-connected deep neural network with real-valued output. Let $m^{(0)} = d$ and $m^{(K)} = 1$ for m the width of the network, and d the shape of the input vector. We then recursively define:

$$x_j^{(0)} = x_j \quad (j = 1, \ldots, m^{(0)}), \tag{34}$$

$$x_j^{(k)} = h\left(\sum_{j'=1}^{m^{(k-1)}} \theta_{j,j'}^{(k)} x_{j'}^{(k-1)} + b_j^{(k)} \right) \quad (j = 1, \ldots, m^{(k)}), \quad k = 1, 2, \ldots, K-1 \tag{35}$$

$$f(x) = x_1^{(K)} = \sum_{j=1}^{m^{(K-1)}} u_j x_j^{(K-1)} \tag{36}$$

where the model parameters can be represented by $w = \{[u_j, \theta_{j,j'}^{(k)}, b_j^{(k)}] : j, j', k\}$ with $m^{(k)}$ being the number of hidden units at layer k; $\theta \in \mathbb{R}^m$ the weight of the neuron. The setting that we would typically find ourselves in would be function approximation on neural network class. Complexity wise, the neural network depends on three main choices of interest. Either the complexity based on unit counts, that is, the number of layers, and the number of neurons per layer, and the configuration of each neuronal unit - for example, between σ sigmoid and ReLU functions.

[4] The *support vector* of a particular hyperplane $\mathbf{y}$ is the collections of all points that lies closest to the hyperplane, which is specified as condition for maximization on both side of maximal margin vector machine.

10.4 Experimental setting

We then have the following control group for experimental setting. Taking inspiration from the framework of Vapnik [1999], we would likely want to adopt Vapnik's control method. Similar to how we argue about the notion of statistical learning, here we also indict on the categorization of the testing model in such configuration.

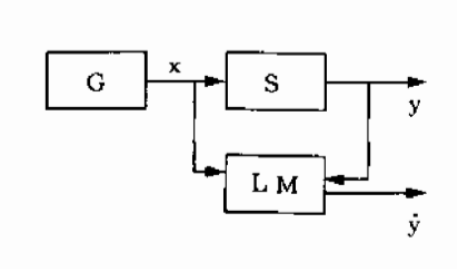


Figure 9: A model of learning from examples. During the learning process, the learning machine observes the pairs (x, y) (the training set). After training, the machine must on any given return a value j. The goal is to return a value y that is close to the supervisor's response y. We will use this interpretation in the experimental setting. Reused from Vapnik [1999]

There by, the class experiments of the above section follows as three types: Next, the SVM model focus on the analysis of binary class for high-dimensional system.

Setting 10.2 (Support vector machine). *We formulate the support vector machine scenario as followed. Let the hypothesis be the class of all $d-1$-dimensional hyperplane opr affine subspace of d-dimensional space equipped with kernel, support vectors, and a binary decision rule* $\operatorname{sign} x \to \{0, 1\}$ *accordingly. Let the concept $c \in \mathcal{C}$ be the class of all binary categorization that either linearly separable or not. Assume a generalized kernel, h holds no knowledge of c aside from binary separation criteria inherent in its design. Consider the set of observations $\mathbb{O}$ of h. Then, it will be controlled of $\mathbb{O} = (\mathbb{U}, \mathbb{I})$ where $\mathbb{U}$ is the* ***input space*** *for k points, and $\mathbb{I} \subseteq \{0, 1\}^k$ is the inferred space of such points. Assume such observations are formed of the distribution $\mathcal{D}^k$ on all points. Consider a random d-dimensional noise (similar to 1-step random walk) $\mathcal{N}(\cdot)$ or normal distribution $\mathcal{N}(\sigma, \mu)$. The learning process then continues as binary classification for either minimization, and accordingly structural minimization.*

The complexity in question that would have to be considered and calculated is the support vectors, or rather, the margin point. Though this effectively means that it does not take all data to form the margin, it still means that there then can exist infinite 'counting complexity' with respect to the parameter counts. Thereby, we would need another meaure of complexity than just as parameters specific to the hypothesis. In fact, the construction of the hypothesis must also consider this aspect itself as a class.

For the neural network structure, we would resort to use a general definition instead; however, the task is function approximation. It is defined as the following setting.

Setting 10.3. *We formulate the neural network scenario as followed. Let the hypothesis be the class of all L-layer fully-connected neural network, for each layer of i standard neuron unit with arbitrary activation functions. Let the operational space be of the field $\mathbb{R}$. The concept $c \in \mathcal{C}[\mathcal{N}_\ell] \cup \mathcal{F}$ is to be all neural network class, fully-connected or not, and all function class $\mathcal{F}$. Consider the observational space $\mathbb{O}$ of h. Then, it will be controlled of $\mathbb{O} = (\mathbb{U}, \mathbb{I})$ where $\mathbb{U}$ is the* ***input space****, and $\mathbb{I}$ is the inferred space. For $\mathbf{x} \in \mathbb{U} \subset \mathbb{R}^d$ is governed by the sampling process*

within distribution of $\mathcal{D}$, and for the inferred space to be of the form $c_i + \lambda\mathcal{N}(\cdot)$ of any given noisy feature of certain formulation of scaling λ. If $\mathcal{N}(\sigma, \mu)$ then we say the inferred space is fully realized as the white noise inferred space. The learning process then continues as binary classification for either minimization, and accordingly structural minimization.

In a sense, the general setting allows us to configure more than necessary. Furthermore, as will be noted in the neural formalism section, virtually every hypothesis can be somewhat generalized to a neural network.

11 Probing experiments' analysis

There are a lot of new observation and problems regarding preliminary models that we would then try to resolve. The double descent curve there is fairly inconclusive, however does exist and indicate that double descent did occur in such specific area.

11.1 Observation 1 - Prescription from Shalev-Shwartz and Ben-David [2014]

The first, and the most simple model that can be taken into account is the model of linear regression with varied setting. The model that is representative of this preliminary observation is from Lafon and Thomas [2024], specifically, linear regression with Gaussian noise. Consider the family class $(\mathcal{H}_p)_{p\in[\![1,d]\!]}$ of linear functions $h : \mathbb{R}^d \mapsto \mathbb{R}$ where exactly p components are non-zero ($1 \leq p \leq d$), we have the following model.

Definition 11.1. *For $p \in [\![1, d]\!]$, $\mathcal{H}_p$ is the set of functions $h : \mathbb{R}^d \mapsto \mathbb{R}$ of the form:*

$$h(u) = u^T w, \quad \text{for } u \in \mathbb{R}^d$$

With $w \in \mathbb{R}^d$ having exactly p non-zero elements. The complexity here itself can be interpreted as the effective random initialization, trimming down data of hidden details instead - this is indicated by the amount of information it observes given d dimension, and p-hypothesis dimension. According to the manifold hypothesis, there are chances where the actual concept can be interpreted and mimicked by seeing that it lies on a lower-dimension manifold inside of its actual expressive dimensional space.

Using such model, they consider the setting of minimizing

$$\min_{w\in\mathbb{R}^d} \frac{1}{2}\|\boldsymbol{X}w - \boldsymbol{Y}\|^2 \to \min_{w\in\mathbb{R}^p} \frac{1}{2}\|\boldsymbol{X}_{\sim \boldsymbol{p}}w - y\|^2 \tag{37}$$

of the sub-problem on p cardinality. This results in the following theorem that 'mimic' double descent on this linear regression model.

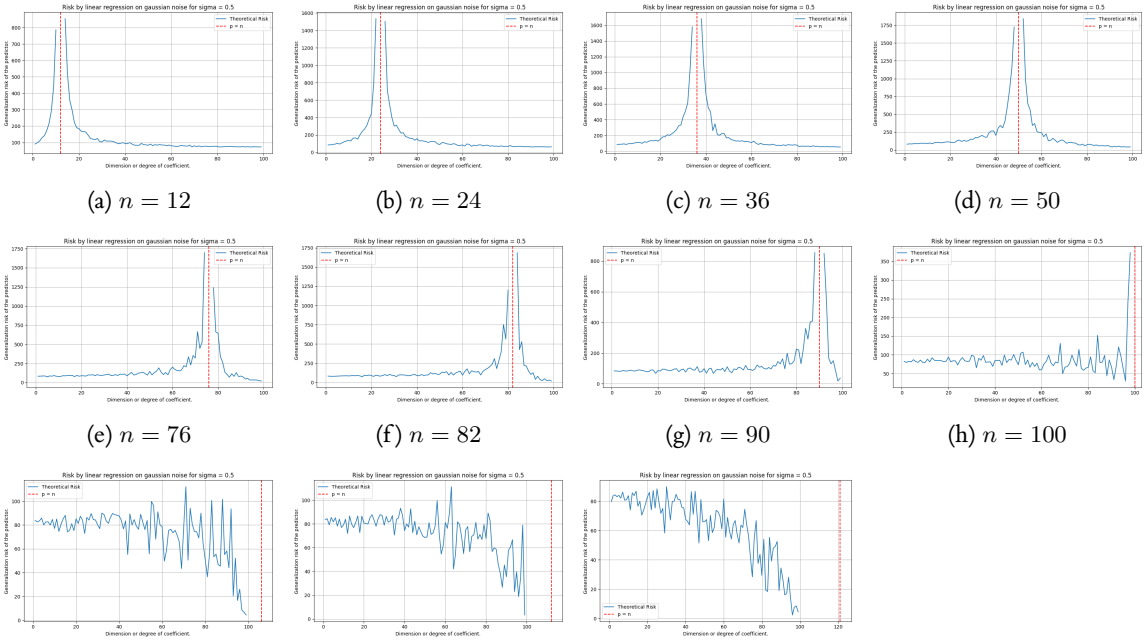
Theorem 11.1 (Double descent on linear regression). *Let $(x, \epsilon) \in \mathbb{R}^d \times \mathbb{R}$ independent random variables with $x \sim \mathcal{N}(0, I)$ and $\epsilon \sim \mathcal{N}(0, \sigma^2)$, and $w \in \mathbb{R}^d$. we assume that the response variable y is defined as $y = x^T w + \sigma\epsilon$. Let $(p, q) \in [\![1, d]\!]^2$ such that $p + q = d$, $\boldsymbol{X}_{\sim \boldsymbol{p}}$ the randomly selected p columns sub-matrix of X. Defining $\hat{w} := \phi_p((\hat{w}), [\hat{w}])$ with $(\hat{w}) = \boldsymbol{X}^+_{\sim \boldsymbol{p}} y$ and $[\hat{w}] = 0$.*
The risk of the predictor associated to $\hat{w}$ is:

$$\mathbb{E}[(y - x^T\hat{w})^2] = \begin{cases} (\|w_{\sim q}\|^2 + \sigma^2)(1 + \frac{p}{n-p-1}) & \text{if } p \leq n-2 \\ +\infty & \text{if } n-1 \leq p \leq n+1 \\ \|w_{\sim p}\|^2(1 - \frac{n}{p}) + (\|w_{\sim q}\|^2 + \sigma^2)(1 + \frac{n}{p-n-1}) & \text{if } p \geq n+2 \end{cases}$$

As said, this theorem is designed to mimic the double descent behaviour on the range of linear regression with Gaussian noise. However, the size of the dataset affects if double descent is emitted or not, at least in terms of regression estimation of wide-field testing set.

The first experimental setting is then conducted with fixed d, varying p for each varying n samples. Since this is a least square result, preliminary run is conducted using one-shot error-risk measure on least square closed-form approximation. Note that $p \leq d$, hence we are regarding the domain of the term underparameterized and equal parameterization setting.

From such data, and setting, we see that for constant noise with expanding dataset, random ordering of such dataset, the analytical solution of best fit turns out to as the risk decomposition, using the



(a) $n = 12$ (b) $n = 24$ (c) $n = 36$ (d) $n = 50$

(e) $n = 76$ (f) $n = 82$ (g) $n = 90$ (h) $n = 100$

(i) $n = 106$ (j) $n = 112$ (k) $n = 121$

Figure 10: Theorem 11.1 behaviours on randomized setting. For this model, we have $d = 100$, $\sigma = 0.5$, and variational n. Full test cases are for $p = [1, 100]$, $n = \{12, 24, 36, 50, 76, 82, 90, 100, 106, 112, 121\}$ accordingly. The test function of the concept itself is the same linear model, but with different input only.

above Theorem 11.1, fails to be realized with large n after certain threshold. What is even more interesting, is that as we find out, the above double-descent replicating formula can actually be realized by the parameter-wised error, or

$$\mathrm{MSE}_{\mathrm{param}} = |\hat{w} - w|_2^2 = \sum_{j=1}^{d} (\hat{w}_j - w_j)^2 \tag{38}$$

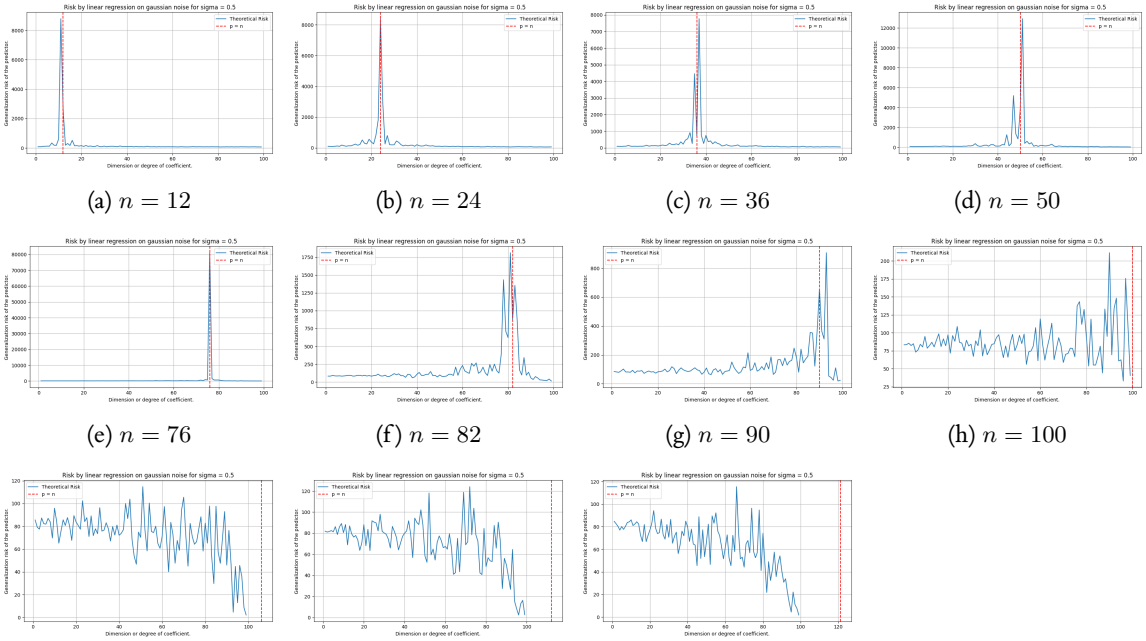
with

$$\hat{w}_j = \begin{cases} \beta_{p,k}, & j = i_k \in S, \\ 0, & j \notin S, \end{cases} \qquad S = \{ i_1, \ldots, i_p \}. \tag{39}$$

and $S = \{i_1, \ldots, i_p\}$ is the set of selected feature indices. The result is represented in Figure 12, as per elementary result. While previously mentioned and analysed in Theorem 11.1 to be equivalent, in such setting, it is entirely nonexistent.

If we are to use the standard measure of MSE on model's response, though, the result is rather fairly disappointing, with no trace of double descent in any given way. This is perhaps the more interesting result, as it represents two masks of measure in one setting that might be overlooked. For the formulation

$$\mathrm{MSE}_{\mathrm{pred}} = \frac{1}{n} \left\| y - X_p \hat{\beta} \right\|_2^2 = \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - x_p^{(i)\,T} \hat{\beta} \right)^2 \tag{40}$$



(a) $n = 12$ (b) $n = 24$ (c) $n = 36$ (d) $n = 50$

(e) $n = 76$ (f) $n = 82$ (g) $n = 90$ (h) $n = 100$

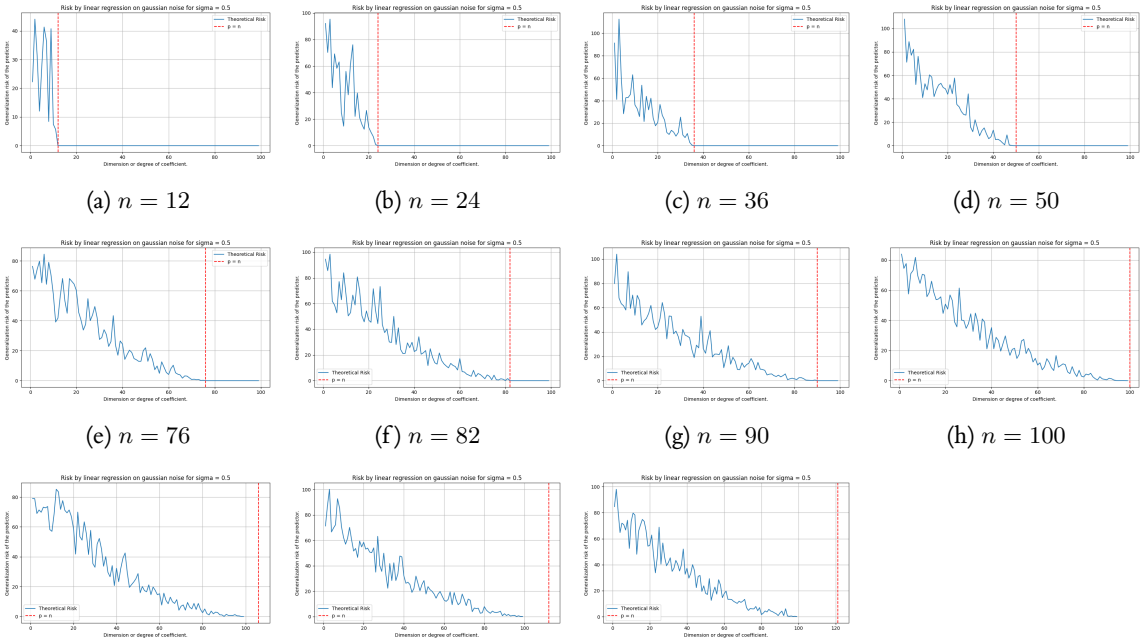
(i) $n = 106$ (j) $n = 112$ (k) $n = 121$

Figure 11: Contrary to Theorem 11.1, this is the behaviours on randomized setting for standard parameter-wised mean square error measure (MSE-on-parameter). For this model, we have $d = 100$, $\sigma = 0.5$, and variational n. Full test cases are for $p = [1, 100]$, $n = \{12, 24, 36, 50, 76, 82, 90, 100, 106, 112, 121\}$ accordingly. The test function of the concept itself is the same linear model, but with different input only.

of the standard model, we gain the following result. This also suggest perhaps double descent indeed is coming from some internal fluctuation and aspect that cannot be dissected from another angle, but apparent in some of the model's behaviours, at least intrinsic in the closed-form approximation solution case.

Thence, the theorem gives us two things - the interpolation point, in this setting, refers to the parameter error for their double descent behaviour. However, on the prediction results' error measure, the interpolation point correlates to the point where the defined generalization risk in such case equals zero. These patterns hold up to certain point. However, the test case is only up to $d = p$. An unexplored test case for $p > d$ indicate the surprising fact according still, to the theorem: the generalization risk takes double descent as a region with disturbance in the middle, but bias-variance appears afterward. This is illustrated in Figure 13, where the same theorem is applied but in the domain which is usually callled as *overparameterized region*.

However, the interpretation of the result is not perfect. Under such setting, then for the concept of the same class, so $d = 150$, any concept of $p > d$ would result in the metric taking zero value of the 'truncated region' where d is literally flat on the $d+1$ onward space. Hence, the error being added on top of such error measure is the component distance of p to the flat sub $p - d$ space itself - which explains the curve going upward, as more and more risks are configured. Nevertheless, the amount of data provided, at least in terms of this model, does indeed have non-trivial effect on the model itself, and the least square



(a) $n = 12$ (b) $n = 24$ (c) $n = 36$ (d) $n = 50$

(e) $n = 76$ (f) $n = 82$ (g) $n = 90$ (h) $n = 100$

(i) $n = 106$ (j) $n = 112$ (k) $n = 121$

Figure 12: Contrary to Theorem 11.1 and both figure 12, this is the behaviours on randomized setting for standard parameter-wised mean square error measure (MSE-on-parameter). Especially, we do not see double descent pattern in this setting, as it is. For this model, we have $d = 100$, $\sigma = 0.5$, and variational n. Full test cases are for $p = [1, 100]$, $n = \{12, 24, 36, 50, 76, 82, 90, 100, 106, 112, 121\}$ accordingly. The test function of the concept itself is the same linear model, but with different input only.

solution of best fit which gives the potential minimal, hence the empirical best. And, considering the results, we can see regions where the theorem 'breaks' of its bias-variance pattern. Further experiment to verify such claim and reconfiguration is needed.

11.2 Experiment 2 - Generalized setting

While the result of previous linear regression is interest, it is without a doubt dubious, especially since the theorem fails in later section for varying n, or at least breaks down in such region. Whether this mean that the theorem predict double descent would not occur in such region, or the theorem fails in said region is unknown. Therefore, we will have to redesign and come up with a separate experiment on our own.

We then define a specific two cases algorithms utilizing either analytical method (least square) or finite optimization (gradient descent) with all controllable factors, as followed.

11.3 Experiment 3 - Polynomial

Figure 13: Similar experiment according to setting and result of Theorem 37, where p is in the range [1,5523]. Here, we can see small double descent modelled exactly at the interpolation threshold, while the later region exhibit similar phenomenon as bias-variance tradeoff.

12 A novel consideration

13 Experiments on Graph Neural Network (GNN)

To confirm and observe particular insights on the problem, experimentally, we focus more on the results given by the analysis on a particular type of neural network, the graph neural network (GNN). However, preliminary experiments are also needed, for example, with various complexity and function class, though most of them are algorithm based on ERM, which typically can be called derivation of gradient descent.

According to major literature, for example, Shi et al. [2024], double descent is absent usually in GNN, for example, GCN networks. However, in fact, it is relatively unstable, as for certain papers and researches point out its existence and variation of it being ubiquitous to the network, some reports none of such occurrence in their experiments. Thereby, our first strategy would be to encounter this absence, and the way to force it to exhibit itself, if the phenomenon is perceived non-existent. In experiments by Shi et al. [2024], Buschjäger et al. [2020], we know that graph network are inherently sensitive to data configuration. The graph MPNN in Hamilton, for example, depends heavily on the configuration of data to present its neighbourhood aggregation for each layer. However, we would like to first claim the dichotomy of bias-variance holds for it to be presented as the first interpolation point.

We would use, and utilizes a hybrid setting between MPNN, GCN and GAN (attention network), to configure out network in itself. However, heterogeneous network - either consists of only MPNN or GCN layers, is somewhat preferred for ease of analysis, and potency of experimental learning control. Because double descent lies between the concept of test error and training error, either supervised or semi-supervised would be used. According to Shi et al. [2024], semi-supervised setting gives better flexibility and overall 'range' of operation. Other than GNN, certain more abstract, 'vanilla' neural network formalism as specified in the above section treatment will also be conducted, in a supplementary manner.

13.1 Methodolody

Because of the irregularity in the statement that GNN does not exhibit any phenomena that is related to double descent Shi et al. [2024], we would be investigating this phenomenon the most. This particularly means that a lot of our focus will be to analyse the GNN network by itself.

13.1.1 Quick introduction to graph theory

A **graph** G is a 2-tuple (V, E) where V is the set of **vertices**(or nodes) and E is the set of **edges**. The set $ne[n]$ stands for the neighbour of vertex n, while $co[n]$ is the set of edges that have n as vertex. Edge is often denoted by (u, v) for vertices u and v. We said that (u, v) **joins** u and v, and it can be directed or undirected. A graph is called a *directed graph* if all edges are direct or *undirected graph* if all edges are *undirected*. The **degree** of vertices v, denoted by $d(v)$, is t number of edges connected with v. The graph data can be loosely (not specifically in cases) as followed. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ be a graph, where $\mathcal{V} = \{1, \ldots, n\}$ is the set of nodes, $\mathcal{E} = \{1, \ldots, M\} \subseteq \mathcal{N} \times \mathcal{N}$ is the set of edges, and $\mathcal{W} : \mathcal{E} \to \mathbb{R}$ is the edge's weight function. There is also the optional weight $\mathcal{W}' : \mathcal{V} \to \mathbb{R}$ for each vertex. In this case, it is for certain problem like TVP, where all cities have certain properties for the path. We say that a data sample $\mathbf{x}$ is a graph data, if its entries are related through the graph $\mathcal{G}$.

Our 2-tuple (V, E) and the collection of all such tuple forms a group of *simple graph* (undirected) and *directed graph*, where the only change is that $(u, v) \neq (v, u)$ for any given edge on a graph. From such, a typical setting, if we are to apply a machine learning setting on graph-theoretical problems, can be described as the following

Much information can be packed on a graph. Typically, the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{M})$ will contains node information and edge information. Edge information can be the path-only, or weight-specific, that is, there exists for an edge (u, v) a property $\phi_{uv} \in [0, 1]$ acting like a weight to gauge a walk on the graph. This can also be used for the *distance measure* of a graph, for such graph being represented with values

per position, that is, each node/vertice has position, and similarly there exists arbitrary distance $|(u, v)|$ between u and v. Any space that encode a graph's information in such way is called an **embedding space**. For node information, connections to other nodes are one type of information; however, it also retains its own properties, typically inferred through the *node embedding space* which contains features of a node and the graph itself. There are a few restrictions of such embedding space, such as to define the node embedding space to be of the same shape n, to avoid 'edge cases' in the process.

The reason for choosing an encoding can be justified as followed. Typically, graphs can be classified and categorized into a specific type of data representation, called *non-Euclidean data*. More specifically, there exists a topological space encapsulating the graph, but exists no fundamental measure on such topological space housing it. For example, there exists no notion of a *discrete distance* on a graph, but only the induced notion of *graph distance* by counting the shortest path from one node to another in a specific graph. Because machine learning works on the assumption that the space of the system gives *measurable space*, a natural response is to encode the system into an encoding space of arbitrary meaning. One, for example, simple encoding is the degree map, where nodes are mapped into an $n \times n$ matrix of nodes and valued by the number of neighbour they have.

This type of encoding then creates, for each graph, a versatiel number of arbitrary properties space they can have, stacked on each other. Typically, for a hypothesis h to work on a graph $\mathcal{G}$, then it will have two main targets that act directly on the graph, as illustrated in Illustration 14. There are many ways

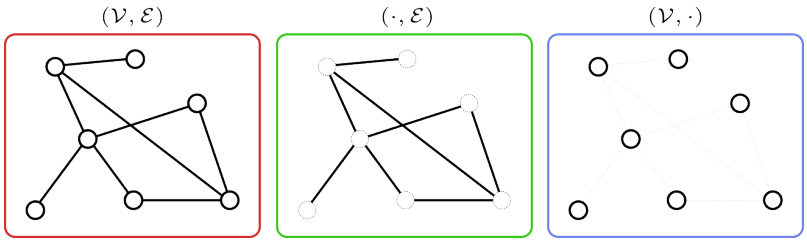


Figure 14: Illustrative decomposition of the graph on its simplexes edges and vertices: Each of the decomposed space $\mathcal{V}$ and $\mathcal{E}$ can be encoded separately to their own ordeal, for example, of the edge connection through incident matrix.

or aspect of a graph to be 'extracted' from the graph features itself. Classically, this is done using feature engineering (Hamilton) and hence the learner is also configured to utilize such features. Nowadays, it is mostly abstracted into an arbitrary representation space of the graph encoding, used in neural network architectures (Oono and Suzuki [2020], Lopushanskyy and Shi [2024], Scarselli et al. [2009], Hamilton). The only different between the classical encoding and the deep learning encoding is on the arbitrary properties of deep learning encoder, of which the transformed feature might not be well-defined feature engineering as the classical case.

This prompts two types of main learning targets: either to have the hypothesis h to learn the edge connections and properties, or to learn on the vertices/node. They are, however, not separated, as to learn about which connections $e \in \mathcal{E}$ to make between two nodes requires information about the general graph's node itself, to predict which node would be connected under obscured information (removing edges). So, there are:

- **Edge reconstruction**: predict if there exists an edge at an arbitrary segment of the graph, for example, between $u, v \in \mathcal{V}$.
- **Edge classification**: An edge $e \in \mathcal{E}$ is endowed a label, and we will then try to classify it based on the actions, or properties it has on the graph.

- **Node classification**: Classify a node $v \in \mathcal{V}$ of certain class of labels.
- **Node regression**: Estimate values of certain nodes. In turns, it might also be used for ranking certain nodes based of specific criteria.

It is helpful to see that a graph can also be classified into two more main archetypes, based off their actions: either **static**, where the node and edges remains the same, or *dynamic*, where deletion of nodes and edges and vice versa are taken into account. The problem revolving a dynamically changing graph (as more prominent in real-life cases) is resolved using temporal methods for graph learning (Rossi et al. [2020]), however, it is much more difficult to analyse, and hence we would restrict ourselves in the case of static graph analysis.

We then state our problems in regard.

Definition 13.1 (Graph learning problem). *Given a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, for any type of graph-theoretical subtype (multigraph, digraph, etc.) by m contraints. There then exist two topologies: the* ***graph-theoretical topology*** *of connections and nodes appearances, and the* ***encoded topology*** *of various measured properties for all $v \in \mathcal{V}, e \in \mathcal{E}$, denoted by (ϕ_G, ϕ_E). A hypothesis $h \in \mathcal{H}$ is then tasked to learn its own encoding, or loosely speaking* ***interpretation*** *of the hypothesis about $\mathcal{G}$ based on the topological space given by $\mathcal{G}$, and use it to target or solve problems related to both ϕ_G and ϕ_E.*

From this, we might derive various problems setting as it is, and considering the set $\{\phi\}$ of encoding intrinsic to the graph.

13.1.2 Graph Neural Network

We target the graph neural network structure (GNN), specifically a neural network implementation for solving graph data problems. A more detailed description can follow from Hamilton, Scarselli et al. [2009]. Typically, graph neural network (Scarselli et al. [2009], Veličković [2023], Tanis et al. [2024], Lopushanskyy and Shi [2024]) follows the instruction flow of the *encoder-decoder* architecture. A GNN is formulated and structured by analysing a graph data system by conceptually apply a *neighbour-dependent* neural input arrangement on top of the data. That is, in principle, it reserves on its own a particular extractor and modifiers on the graph interpretation itself, through its layers.

Structurally, the description of a GNN is defined by the overall *message-passing neural network* (MPNN), defined by (PyTorch Geometric Team [2025], Fey and Lenssen [2019]):

$$\mathbf{x}_i^{(k)} = \gamma^{(k)} \left(\mathbf{x}_i^{(k-1)}, \bigoplus_{j \in \mathcal{N}(i)} \phi^{(k)} \left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{j,i} \right) \right), \tag{41}$$

for $\bigoplus$ the differentiable, permutation invariant function, usually called the aggregator, $\mathbf{x}_i^{(k-1)} \in \mathbb{R}^F$ the node features of node i in passing layer $k-1$, $\mathbf{e}_{j,i} \in \mathbb{R}^D$ the optional edge features from node j to node i. Additionally, γ denotes the nonlinearity differentiable function (usually ReLU, or sigmoid) ϕ denote the MLP layer accompanied They are often called the **updater** and the **messenger**, respectively. A simplified example of this structure in Scarselli et al. [2009], Hamilton as

$$\mathbf{x}_i^{(k)} = \sigma \left(\mathbf{W}_{\text{self}}^{(k)} \mathbf{x}_i^{(k-1)} + \mathbf{W}_{\text{neigh}}^{(k)} \sum_{v \in \mathcal{N}(i)} \mathbf{x}_v^{(k-1)} + \mathbf{b}^{(k)} \right) \tag{42}$$

where $\mathbf{W}_{\text{self}}^{(k)}, \mathbf{W}_{\text{neigh}}^{(k)}$ are trainable parameter matrices, and σ denotes an elementwise nonlinearity, and an optional bias term $\mathbf{b}^{(k)}$. Different flavours of the differentiable aggregator create the *graph convo-*

lutional networks (GCNs), defined by:

$$\mathbf{x}_i^{(k)} = \sigma\left(\mathbf{W}^{(k)} \sum_{v \in \mathcal{N}(i) \cup \{i\}} \frac{\mathbf{x}_v}{\sqrt{|\mathcal{N}(i)||\mathcal{N}(v)|}}\right) \tag{43}$$

A somewhat popular approach is to apply attentional layer and weights to facilitate neighbourhood attention, which is called graph attention network.

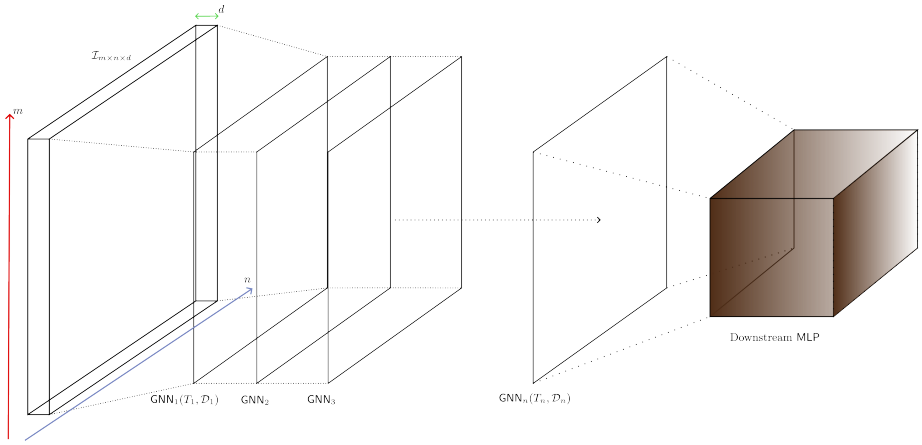


Figure 15: A conceptual illustration on the running flow of an n-layer GNN on particular structure of interest. Note that the data section itself has particular embedding structure on its own.

As noted by Hamilton, node features are often required to be inputted into the GNN. Usually, this is of the form $\mathbf{x}_u$ for all $u \in \mathcal{V}$. The same can be said for edge in specific tasks required of them. However, if there are no sufficient node representation possible in the first glance, then we can use node/edge statistics as classical techniques.

A GNN is, when fitting into the framework of learning system and modeling, is a feature mask generation and adaptation mechanism of the modelling pipeline. What this means is that GNN *assumes* every data point has its own feature-embedded space, and according relationships. By this, we further mean that it creates an embedding space of the aggregator, or the GNN-embedding space within such. While input embedding space captures and represents the best possible interpretation of the input space, GNN-embedding space captures what is required of the aggregation properties that the GNN layers specify. Hence, we can assume relative fallacy in such embedded space. Those n-embedded space of data aggregated at the nth final layer, would then be fed into another straightforward - task-based and structured network, such as a pass to a single-layer sigmoid network, Hamming network, or generally an FCN.

13.2 Double descent and GNN

As of date, there has been no reports on double descent on GNN. This can be taken accounted of the complex and often difficult analysis for graph and the neural network adopted for such task. Because of this, we will have to attempt to force double descent appears instead of simply observing it. To do this, we need to specify both the task for evaluation for the error measure, and model complexity for the other axis. For such, we would often reserve to node-level or edge-level task, and not graph-level tasks.

Aside from layer-wised complexity, GNN also has two types of complexity: the size of the graph itself, and the size of the encoding in specification. Thence, the complexity would then at least be the

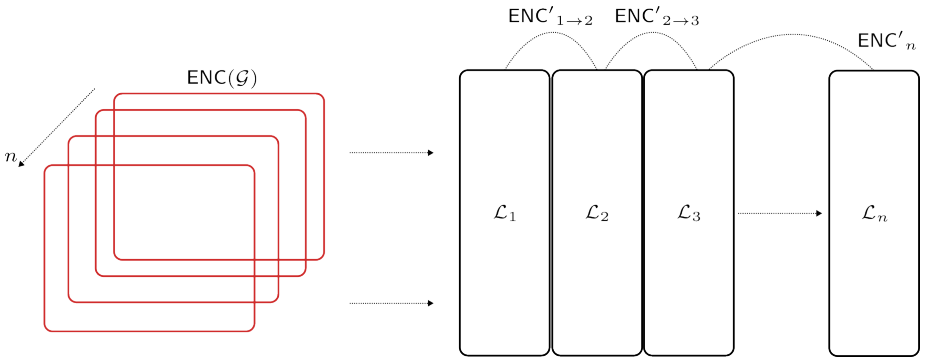


Figure 16: Encoding space and the change of encoding inbetween a static graph (or snapshot-wise) GNN. The encoding space is warped toward arbitrary notions inside a GNN, to the point that after certain layers of processing, the encoding outputs different from expected encoding space (native to the model, not to the designer), though there might be universal notions preserved, like the degree of node represented in a different way.

tuple (m, n, l) for m-size of the original graph, n-depth of the GNN layers, and l layers of GNN. If n varies between layers, then we will have the index set instead. However, for simplicity, we would likely want to have constant n throughout all layer.

14 Physics-inspired perspective

To analyse systems with quenched disorder, such as a neural network trained on a fixed dataset, one must compute the average of the logarithm of the partition function, $\langle \log Z \rangle$, over all possible datasets. This is a mathematically challenging task. The replica method, originating from the statistical mechanics of spin glasses, provides a powerful, albeit non-rigorous, analytical technique using the **replica trick**:

$$\mathbb{E}_{\mathcal{D}}[\log Z] = \lim_{n \to 0} \frac{\log \mathbb{E}_{\mathcal{D}}[Z^n]}{n} \tag{44}$$

This identity converts the difficult average of a logarithm into an average of an integer power Z^n, which is more tractable. The procedure involves introducing n identical, non-interacting copies (replicas) of the system, calculating $\langle Z^n \rangle$, and then analytically continuing the result to the $n \to 0$ limit.

To illustrate, consider a 2-layer committee machine with K hidden units, N inputs, and weights w_k constrained to a hypersphere $||w_k||^2 = N$. The partition function Z, also known as the Gardner Volume, measures the volume of weight space satisfying P input-output constraints (x^μ, y^μ):

$$Z = \int \prod_{k=1}^{K} dw_k \delta(N - ||w_k||^2) \prod_{\mu=1}^{P} \Theta(y^\mu s(x^\mu) - \kappa) \tag{45}$$

where $s(x^\mu)$ is the network output for input x^μ and Θ is the Heaviside step function ensuring a classification margin κ.

After averaging Z^n over the data distribution, the calculation is managed by introducing *order parameters* that capture the geometric relationships between replicas. By taking the $n \to 0$ limit, we derive a set of coupled saddle-point equations whose solution reveals the typical properties of the solution space, such as the critical storage capacity $\alpha_c = P/N$. This method provides a first-principles approach to understanding how network architecture and data properties jointly determine the geometry of the solution space and the onset of phase transitions in learning. While this canonical example differs from the GNNs that are our primary focus, it serves to illustrate the core methodology: how the replica trick, through the introduction of order parameters, can transform a problem of counting solutions into a tractable analysis of the solution space geometry. We will apply the principles derived here to qualitatively understand the more complex GNN landscape.

14.1 A Formal Link via PAC-Bayesian Bounds

We now formalize the connection between the entropy of the solution space and generalization. The Probably Approximately Correct (PAC)-Bayesian framework provides a bound on the true risk $R(Q)$ of a posterior distribution Q over hypotheses (weights):

$$R(Q) \leq \hat{R}(Q) + \sqrt{\frac{KL(Q||P) + \log(m/\delta)}{2m}} \tag{46}$$

Here, P is a data-independent prior, Q is the data-dependent posterior, $\hat{R}(Q)$ is the empirical risk on m samples, and δ is the confidence parameter. The key is the Kullback-Leibler (KL) divergence, $KL(Q||P)$, which measures the information-theoretic "cost" of updating the prior P to the posterior Q after observing the data.

To make this connection rigorous, we introduce the Gibbs posterior, Q^*_λ, which is the optimal choice for Q that minimizes the trade-off $\lambda \hat{R}(Q) + KL(Q||P)$:

$$Q^*_\lambda(h) = \frac{P(h)e^{-\lambda \hat{R}(h)}}{Z(\lambda)}, \quad \text{where} \quad Z(\lambda) = \int P(h')e^{-\lambda \hat{R}(h')}dh' \tag{47}$$

This formulation reveals a direct analogy with statistical mechanics:

- The hypothesis h corresponds to a *state*.
- The empirical risk $\hat{R}(h)$ corresponds to the *energy* E.
- The hyperparameter λ corresponds to the *inverse temperature* $\beta = 1/T$.
- The normalization constant $Z(\lambda)$ is the *partition function*.
- The quantity $F(\lambda) = -(1/\lambda) \log Z(\lambda)$ is the *Helmholtz free energy*.

This establishes a rigorous chain of reasoning: a large volume of low-loss solutions (a "flat minimum") implies high entropy $\rightarrow$ which in turn lowers the Helmholtz free energy $\rightarrow$ leading to a smaller KL divergence term in the PAC-Bayesian bound $\rightarrow$ and ultimately certifying better generalization. The partition function, by integrating over all possible solutions weighted by their empirical risk, effectively measures the volume of "good" solutions. Therefore, finding a region in the parameter space with a large volume of low-loss solutions directly translates to a tighter, more robust generalization bound as certified by the PAC-Bayesian framework.

15 Topological Criticality in Graph Neural Networks

15.1 The Interpolation Peak as a Critical Phenomenon

We hypothesize that the interpolation peak observed in GNNs is a critical point where generalization error ϵ diverges according to a scaling law, $\epsilon(k) \sim |k - k_c|^{-\nu}$, where k is model complexity. While analogies to physical systems like the Ising model are instructive, GNNs on heterogeneous graphs exhibit a unique form of criticality.

Unlike a standard second-order phase transition in a mean-field model, where criticality is marked by the divergence of a correlation length, GNNs on large, sparse random graphs (e.g., Barabási-Albert model) can suffer from an expressivity collapse. This is driven by *local weak convergence*: as graph size $N \rightarrow \infty$, the local r-hop neighbourhood of almost every node converges to a universal random tree structure. Since a k-layer GNN's output is solely a function of its k-hop neighbourhood, the model's output becomes constant across all nodes, losing all discriminative power.

Table 2: Comparison of Criticality Mechanisms

Property	**Ising Model (Mean-Field Analogy)**	**GNN on a Large Random Graph**
Transition Type	Second-order phase transition.	Expressivity collapse.
Governing Concept	Divergence of correlation length ξ.	Local weak convergence of neighbourhoods.
Order Parameter	Magnetization M.	Discriminative power of the GNN output.
Result	System develops long-range correlations, described by exponent $\nu = 1/2$.	GNN output becomes constant, losing all instance-specific info.

The criticality at the GNN interpolation peak is not merely a quantitative divergence but a qualitative shift where the model loses its ability to distinguish between nodes due to fundamental graph topology at scale. We posit that this catastrophic expressivity collapse is the microscopic mechanism responsible for the macroscopic peak in generalization error observed in the double descent curve for GNNs on large, sparse graphs. This reframes the challenge as understanding the architectural and topological conditions that trigger this collapse.

Details for supervised learning setting

In a typical machine learning setting, we obtain assume the following figure 17. This includes the ground, input space (the setting of which the observations are observed), the concept $c \in \mathcal{C}$, the hypothesis $c \in \mathcal{H}$, and a supervisor ∇ that 'correct' the behaviours of the hypothesis to match with c. This is called a supervised learning setting, in which the learning part is governed by the supervisor and the observations from the concept.

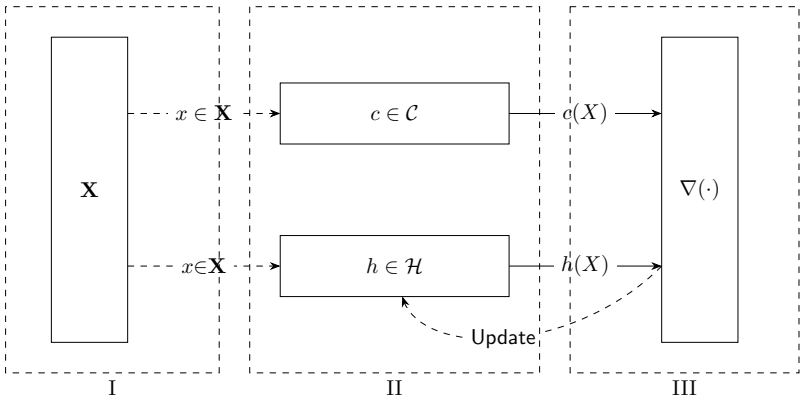


Figure 17: An illustration of the (supervised) statistical process. Phase III contains two parts: First is the evaluation $\nabla(h, c)$ according to the data $\mathcal{D}$, and second is the Update process to re-align c to the actual target.

Phase 1.

Begin with the construction and initialization of the feature space $\mathcal{X}^{\infty}$. By the assumption of a probabilistic process, $\mathcal{X}_m$ or simply $\mathcal{X}$ representing the dataset is assumed to be sampled, or appeared from the distribution $\mathcal{D}(p, \cdot)$ for p an arbitrary probabilistic system.

The *ground space*, or in literature generally called **feature space**, is created. This results in the first component of the tuple D, being $\mathcal{X} \subseteq \mathbb{R}^{m \times k}$, where $|\mathcal{X}| = m$, but $size(x) = k$, for $x \in \mathcal{X}$. We assume there exists the random variable x of a given distribution $\mathcal{D}$, such that the set $\mathcal{X} \sim \mathcal{D}$ of all observation is given by an underlying distribution. A well-known assumption in this case is that $x \in \mathcal{X}$ has no dependent components. That is, for example, there does not exist any function $\psi_x : \{x_i\} \to \{x_j\}$, for $i \neq j$. (Thought, this only helps in Bayesian priori analysis). Furthermore, by specifying that $\mathcal{X}$ belongs to a specific distribution, we argue of the assumption that $x \in \mathcal{X}$ is *independently and identically distributed* (i.i.d.), that is, for a given random sampling interpretation $S \subset \mathcal{X} \sim \mathcal{D}$, all data is sampled with the events space governed by the distribution $\mathcal{D}$ for every instance, and the existence of x_i to x_j for $i \neq j$ is none.

The **role** of the distribution configuration is important. If $\mathcal{D}$ is uniform, that is, $D \sim \mathcal{U}(a, b)$, then we can disregard the aspect of $\mathcal{X}$ with random variables. For $[a, b]$ to be $(-\infty, +\infty)$, the problem changed to a *regression problem*. If $\mathcal{D}$ is some other distribution function, then the problem of *representative data* and *population size* becomes apparent, which requires statistical analysis to be taken into consideration.

Phase 2.

For the constructed feature space $\mathcal{X}$, let $EX(c, \mathcal{D})$ be a procedure (or *oracle*) acting on $\mathcal{C}$ and $\mathcal{X}$ to output $\langle x, c(x) \rangle$. The hypothesis then contains a similar procedure $EX_{\mathcal{H}}(h, \mathcal{D})$, such that to output $\langle x, h(x) \rangle$. We call this the *inference phase*.

The feature space $\mathcal{X}$ is provided to h. We assume c is processed of the same process, resulted in $\mathcal{Y}_c$. Then, $h(\mathcal{X})$ outputs the set of hypothesis' target $\mathcal{Y}_h$. Thereby, we are approximating the *process* c itself. We can approach this in two main ways[5]: either *deterministic* or *probabilistic*[6]. We define such as followed. For the constructed feature space $\mathcal{X}$, let $EX(c, \mathcal{D})$ be a procedure (or *orcale*) acting on $\mathcal{C}$ and $\mathcal{X}$ to output $\langle x, c(x)\rangle$. The hypothesis then contains a similar procedure $EX_{\mathcal{H}}(h, \mathcal{D})$, such that to output $\langle x, h(x)\rangle$. We call this the *inference phase*.

Definition 0.1 (Deterministic - *discriminative modelling*). *A* ***deterministic model*** h_d *assumes its internal space as followed. For any $h \in \mathcal{H}_d$ of deterministic model, h is characterized by the mapping $h_d : \mathcal{X} \to \mathcal{Y}_h$, such that $h \in C^n$ of n-differential space.*

Similarly, if the interpretation of the model is *probabilistic*, we have the following definition of probabilistic-based model.

Definition 0.2 (Probabilistic - *generative modelling*). *A* ***probabilistic model*** h_p *assumes its internal space as followed. For any $h \in \mathcal{H}_p$ of all probabilistic hypothesis, h is characterized by the probability distribution $\Gamma(p_X(X))$, where Γ is the discrete decision process outside the probabilistic interpretation.* [7]

Notice that the process and the learning setting is probabilistic, however, the interpretation of the hypothesis h can be either deterministic or probabilistic, or anything else. This puts the flexibility into the setting of the model, and permits us to consider various representation of the model structure. Furthermore, the hypothesis for the learning process is evaluated relative to the same probabilistic setting, in which the training takes place, and we allow hypotheses that are only approximation of the target concept Sugiyama [2015], Kearns and Vazirani [1994].

The similarity between both of the two general types is that the model is characterized by their **parameters**, both **model-specific parameters** and **hyperparameters**.

Phase 3.

There exists an evaluator (or *supervisor*) $\nabla(h, c)$ that evaluates specific *loss framework* $\ell\{h(x), c(x)\}$ based on available information, and a *update modifier* $U(\ell, \mathcal{A})$ of certain objective to algorithm $\mathcal{A}$.

In this step of the procedural setting, the *supervisor* includes a loss function, ℓ, which takes discrete arguments, and 'binarily' evaluate the discrete result between h and c. This loss function is supposed to have a global minimum, or at least a certain region of minimal approximation. We have the following definition.

Definition 0.3 (Loss function). *A loss function is a non-negative function $\ell : \mathcal{Y} \times \mathcal{Y} \to [0, +\infty)$.*

In general, the following is supposed of the loss function:

Conjecture 0.1 (Loss function convexity). *For any supervisor ∇ of a learner framework $\mathcal{L}(\mathcal{H}, \mathcal{A})$, the loss function ℓ is assumed to be a p-converging function, or as a convex function. That is, for $\ell : \mathbb{R}^n \to \mathbb{R}$, then ℓ is*

[5] Note that, in some aspect, this is similar to the usage of mathematical simulation of certain dynamical system, or *any* system that has certain patterns or relations observable.

[6] The model itself can be entirely deterministic, while the learning process is not. in fact, one of the very first assumption and axiomatic view of the learning problem is that learning occurs in a *probabilistic setting*.

[7] The discrete decision process Γ is very simple to argue. For example, given a Naive Bayes model with $p(C_k \mid D[i])$ of the dataset, the probability distribution is regarded as the argument

$$Y^* = \arg\max_{c_k \in C} P_\theta(c_k \mid x) = \arg\max_{c_k \in C} P_\theta(x \mid c_k) \cdot P(c_k)$$

convex on its domain, or is locally convex on $[a, b] \subset \mathbb{R}^n$*, if, for* $x, y \in \mathbb{R}^n$ *of such interval, and for* $\lambda \in [0, 1]$*, it satisfies*

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y) \tag{48}$$

Under a single structure, that is, for example, a class of mapping $\mathbb{R}^n \to \mathbb{R}$, there can exist many loss functions to be considered. Take for example the setting of regression analysis. Then, ℓ can be either the absolute error, $|h(x) - c(x)|$, or the L^2 norm, $||h(x), c(x)||_2$. Different loss function provides different path and landscape, though they still share somewhat similar interpolation patterns, or either some might be subjected to, under the consideration of a loss landscape, local minima than others. Furthermore, the form and representation of loss functions is not discrete - but rather based of its interpretation and the supposition of the underlying notion required for the loss function to operate in. For example, if the setting is *generative*, the loss function would take the form of a *divergence measure* between two probability distribution. Notice however, that the action potential provided by the model in their operation dissapears, or rather, is not considered, when using the loss function.

Using such loss function, for the case of no noises or interference uncertainty, if the goal is to minimize the empirical risk on such dataset, then we call this the method of *empirical risk minimization*, or ERM. This is born out of the consideration that in practice, the oracle $EX(c, \mathcal{D})$, and the actual generalization error $R(h)$ is not obtainable.

Bibliography

[PDF] A Unifeid Bias-Variance Decomposition and its Applications | Semantic Scholar. URL https://www.semanticscholar.org/paper/A-Unifeid-Bias-Variance-Decomposition-and-its-Domingos/e1ed9d24db5e8f7ab326aeb797e965a94f5ad6d3.

Panos Achlioptas. Stochastic Gradient Descent in Theory and Practice. *Lecture note, Stanford's AI*.

Ben Adlam and Jeffrey Pennington. Understanding double descent requires a fine-grained bias-variance decomposition, 2020. URL https://arxiv.org/abs/2011.03321.

Madhu S. Advani and Andrew M. Saxe. High-dimensional dynamics of generalization error in neural networks, 2017. URL https://arxiv.org/abs/1710.03667.

Pablo Barceló, Mikaël Monet, Jorge Pérez, and Bernardo Subercaseaux. Model interpretability through the lens of computational complexity, 2020. URL https://arxiv.org/abs/2010.12265.

Mikhail Belkin, Siyuan Ma, and Soumik Mandal. To understand deep learning we need to understand kernel learning, 2018. URL https://arxiv.org/abs/1802.01396.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine learning practice and the bias-variance trade-off. *Proc. Natl. Acad. Sci. U.S.A.*, 116(32):15849–15854, August 2019. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1903070116. URL http://arxiv.org/abs/1812.11118. arXiv:1812.11118 [cs, stat].

Olivier Bousquet, Steve Hanneke, Shay Moran, Ramon van Handel, and Amir Yehudayoff. A theory of universal learning, 2020. URL https://arxiv.org/abs/2011.04483.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021. URL https://arxiv.org/abs/2104.13478.

Gavin Brown and Riccardo Ali. Bias/variance is not the same as approximation/estimation. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=4TnFbv16hK.

Sebastian Buschjäger, Lukas Pfahler, and Katharina Morik. Generalized Negative Correlation Learning for Deep Ensembling, December 2020. URL http://arxiv.org/abs/2011.02952. arXiv:2011.02952 [cs, stat].

Nello Cristianini and John Shawe-Taylor. An introduction to support vector machines and other kernel-based learning methods. 2000. URL https://api.semanticscholar.org/CorpusID:60486887.

Stéphane d' Ascoli, Levent Sagun, and Giulio Biroli. Triple descent and the two kinds of overfitting: where & why do they appear? In *Advances in Neural Information Processing Systems*, volume 33, pages 3058–3069. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1fd09c5f59a8ff35d499c0ee25a1d47e-Abstract.html.

Xander Davies, Lauro Langosco, and David Krueger. Unifying Grokking and Double Descent, March 2023. URL http://arxiv.org/abs/2303.06173. arXiv:2303.06173 [cs].

Pedro M. Domingos. A unified bias-variance decomposition for zero-one and squared loss. In *AAAI/IAAI*, 2000a. URL https://api.semanticscholar.org/CorpusID:2063488.

Pedro M. Domingos. A Unifeid Bias-Variance Decomposition and its Applications. In *Semantic Scholar*, June 2000b. URL https://www.semanticscholar.org/paper/A-Unifeid-Bias-Variance-Decomposition-and-its-Domingos/e1ed9d24db5e8f7ab326aeb797e965a94f5ad6d3.

George Casella E. L. Lehmann. *Theory of Point Estimation*. Springer Texts in Statistics. Springer-Verlag, New York, 1998. ISBN 978-0-387-98502-2. doi: 10.1007/b98854. URL http://link.springer.com/10.1007/b98854.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. URL https://arxiv.org/abs/1903.02428.

Scott Fortmann. Understanding the Bias-Variance Tradeoff, 2012. URL https://scott.fortmann-roe.com/docs/BiasVariance.html.

Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992. doi: 10.1162/neco.1992.4.1.1.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Bruce Hajek and Maxim Raginsky. *Statistical Learning Theory*, volume 1. 2021. URL https://maxim.ece.illinois.edu/teaching/SLT/.

William L. Hamilton. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159.

Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *arXiv preprint arXiv:1903.08560*, 2019.

Xia Hu, Lingyang Chu, Jian Pei, Weiqing Liu, and Jiang Bian. Model complexity of deep learning: A survey, 2021. URL https://arxiv.org/abs/2103.05127.

Gareth James, Trevor Hastie, Robert Tibshirani, and Daniela Witten. *An introduction to statistical learning : with applications in R*. New York : Springer, [2013] ©2013, 2013. URL https://search.library.wisc.edu/catalog/9910207152902121.

Romuald A. Janik and Przemek Witaszczyk. Complexity for deep neural networks and other characteristics of deep feature representations, 2021. URL https://arxiv.org/abs/2006.04791.

Steven M. Kay. Fundamentals of statistical signal processing: estimation theory | Guide books | ACM Digital Library, 1993. URL https://dl.acm.org/doi/10.5555/151045.

Michael J. Kearns and Umesh V. Vazirani. *An introduction to computational learning theory*. MIT Press, Cambridge, MA, USA, 1994. ISBN 0262111934.

Marc Lafon and Alexandre Thomas. Understanding the Double Descent Phenomenon in Deep Learning, March 2024. URL http://arxiv.org/abs/2403.10459. arXiv:2403.10459 [cs, stat].

Chris Yuhao Liu and Jeffrey Flanigan. Understanding the role of optimization in double descent, 2023. URL https://arxiv.org/abs/2312.03951.

Dmytro Lopushanskyy and Borun Shi. Graph neural networks on graph databases, 2024. URL https://arxiv.org/abs/2411.11375.

Jing Luo, Huiyuan Wang, and Weiran Huang. Investigating the impact of model complexity in large language models, 2024. URL https://arxiv.org/abs/2410.00699.

Song Mei and Andrea Montanari. The generalization error of random features regression: Precise asymptotics and double descent curve. *arXiv preprint arXiv:1908.05355*, 2019.

Song Mei and Andrea Montanari. The generalization error of random features regression: Precise asymptotics and double descent curve, 2020. URL https://arxiv.org/abs/1908.05355.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. ISBN 026201825X.

Christoph Molnar, Giuseppe Casalicchio, and Bernd Bischl. *Quantifying Model Complexity via Functional Decomposition for Better Post-hoc Interpretability*, page 193–204. Springer International Publishing, 2020. ISBN 9783030438234. doi: 10.1007/978-3-030-43823-4_17. URL http://dx.doi.org/10.1007/978-3-030-43823-4_17.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep Double Descent: Where Bigger Models and More Data Hurt, December 2019. URL http://arxiv.org/abs/1912.02292. arXiv:1912.02292 [cs, stat].

Brady Neal. On the bias-variance tradeoff: Textbooks need an update, 2019. URL https://arxiv.org/abs/1912.08286.

Brady Neal, Sarthak Mittal, Aristide Baratin, Vinayak Tantia, Matthew Scicluna, Simon Lacoste-Julien, and Ioannis Mitliagkas. A modern take on the bias-variance tradeoff in neural networks. *arXiv preprint arXiv:1810.08591*, 2018.

Amanda Olmin and Fredrik Lindsten. Towards understanding epoch-wise double descent in two-layer linear neural networks, 2024. URL https://arxiv.org/abs/2407.09845.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1ldO2EFPr.

Liam Paninski. Statistics 4107: Intro to Math Stat (fall 2005), 2005. URL https://sites.stat.columbia.edu/liam/teaching/4107-fall05/.

David Pfau. A generalized bias-variance decomposition for bregman divergences. Technical report, 2013.

Villares Piera and Nemesio Javier. *Sample Covariance Based Parameter Estimation For Digital Communications*. Doctoral thesis, Universitat Politècnica de Catalunya, October 2005. URL https://upcommons.upc.edu/handle/2117/94206. Accepted: 2011-04-12T15:27:01Z ISBN: 9788468995571 Publication Title: TDX (Tesis Doctorals en Xarxa).

PyTorch Geometric Team. Creating message passing networks, 2025. URL https://pytorch-geometric.readthedocs.io/en/2.6.1/notes/create_gnn.html.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs, 2020. URL https://arxiv.org/abs/2006.10637.

Sebastian Ruder. An overview of gradient descent optimization algorithms, June 2017. URL http://arxiv.org/abs/1609.04747. arXiv:1609.04747 [cs].

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. doi: 10.1109/TNN.2008.2005605.

Rylan Schaeffer, Mikail Khona, Zachary Robertson, Akhilan Boopathy, Kateryna Pistunova, Jason W. Rocks, Ila Rani Fiete, and Oluwasanmi Koyejo. Double Descent Demystified: Identifying, Interpreting & Ablating the Sources of a Deep Learning Puzzle, March 2023. URL http://arxiv.org/abs/2303.14151. arXiv:2303.14151 [cs, stat].

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, USA, 2014. ISBN 1107057132.

Rahul Sharma and Alex Aiken. Bias-variance tradeoffs in program analysis. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '14, pages 127–137, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 978-1-4503-2544-8. doi: 10.1145/2535838.2535853. URL https://doi.org/10.1145/2535838.2535853.

Cheng Shi, Liming Pan, Hong Hu, and Ivan Dokmanić. Homophily modulates double descent generalization in graph convolution networks, 2024. URL https://arxiv.org/abs/2212.13069.

Tom F. Sterkenburg. Statistical learning theory and occam's razor: The core argument. *Minds and Machines*, 35(1), November 2024. ISSN 1572-8641. doi: 10.1007/s11023-024-09703-y. URL http://dx.doi.org/10.1007/s11023-024-09703-y.

Masashi Sugiyama. *Introduction to Statistical Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2015. ISBN 9780128023501.

James H. Tanis, Chris Giannella, and Adrian V. Mariano. Introduction to graph neural networks: A starting point for machine learning engineers, 2024. URL https://arxiv.org/abs/2412.19419.

L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, November 1984. ISSN 0001-0782. doi: 10.1145/1968.1972. URL https://doi.org/10.1145/1968.1972.

Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer: New York, 1999.

Petar Veličković. Everything is connected: Graph neural networks. *Current Opinion in Structural Biology*, 79:102538, April 2023. ISSN 0959-440X. doi: 10.1016/j.sbi.2023.102538. URL http://dx.doi.org/10.1016/j.sbi.2023.102538.

Zitong Yang, Yaodong Yu, Chong You, Jacob Steinhardt, and Yi Ma. Rethinking Bias-Variance Trade-off for Generalization of Neural Networks, December 2020. URL http://arxiv.org/abs/2002.11328. arXiv:2002.11328 [cs, stat].

Jiawei Zhang. Gradient Descent based Optimization Algorithms for Deep Learning Models Training, March 2019. URL http://arxiv.org/abs/1903.03614. arXiv:1903.03614 [cs].

Tong Zhang. *Mathematical Analysis of Machine Learning Algorithms*. Cambridge University Press, 2023. doi: 10.1017/9781009093057. URL https://www.cambridge.org/core/books/mathematical-analysis-of-machine-learning-algorithms/EB9BABB05A5C312F19C38E5A01A5ECFC.