DOUBLE DESCENT AND STATISTICAL THEORY

Bui Gia Khanh *
Department of Physics Hanoi University of Science, Vietnam National University
Hanoi, Vietnam
{fujimiyaamane}@outlook.com

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 would like to review through progress made in this particular field of research, and the phenomena potential explanations up until now.

1 INTRODUCTION

The theory of 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 re-evaluation 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 some particular insights and analysis, and a fix within interpretation of the phenomena.

Our main focus is to review 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 & Ben-David (2014); Hajek & 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 (2000), which serves

*Undergraduate student

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 & 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.

On itself, the phenomena *double descent* has been investigated somewhat comprehensively, firstly introduced by Belkin et al. (2019), which gives it distinctive saddle escape illustration. Further analysis was made by several literatures, particularly attributed the existence of double descent to the concept of model complexity and inductive bias, and potential explanations of such. 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. Recently, there is also the Neural Tangent Kernel (Jacot et al. (2018)) which can be used to explain double descent, however further researches are still required. Preliminary works are done by Lafon & Thomas (2024), Schaeffer et al. (2023), and Liu & 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 & 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.

2 GRAPH THEORETICAL LEARNING

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.

3 METHODOLOGY

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.

3.0.1 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 $\mathcal{G}$.

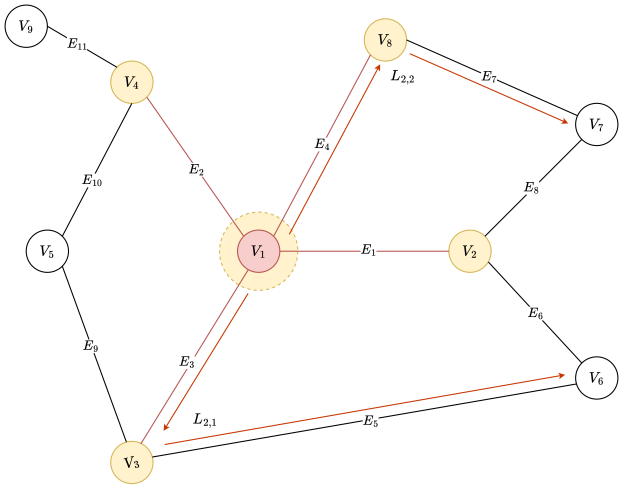

Figure 1: **Illustration of an ordering for a specific graph configuration.** We emphasize the perspective to the higher neighbouring degree node in the graph, namely V_1, and their relative links and connections of the graph itself. A 2-walk (in red) can be seen, which aggregates more to create the 2-neighbourhood of maximally two walks away.

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 2. There are many

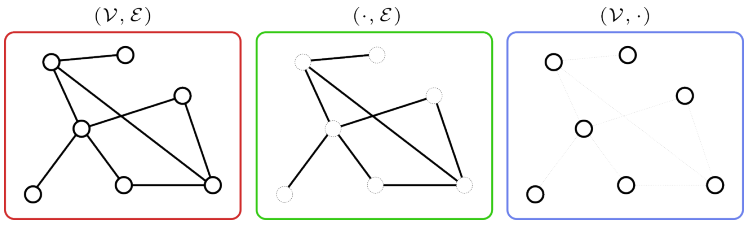

Figure 2: Illustrative decomposition of the graph on its simplexes edges and vertices: Each of the decomposed space $\mathcal{V}$ and $\mathcal{E}$ can be encoded separately to their own ordeal, for example, of the edge connection through incident matrix.

ways 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 & Suzuki (2020); Lopushanskyy & 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 (**?**), 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 3.1 (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.

3.0.2 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 & 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. Given a graph $G(V, E)$, then the neighbourhood $neigh(v_i)$ for $v_i \in V$ is defined by

$$neigh(v_i) = \{v_i, v_j\}_j : (v_i, v_j) \in E$$

Structurally, the description of a GNN is defined by the overall *message-passing neural network* (MPNN), defined by (**??**):

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

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

$$\mathbf{x}_i^{(k)} = \sigma \left(\mathbf{W}_{\text{self}}^{(k)} \mathbf{x}_i^{(k-1)} + \mathbf{W}_{\text{neigh}}^{(k)} \sum_{v \in \mathcal{N}(i)} \mathbf{x}_v^{(k-1)} + \mathbf{b}^{(k)} \right) \tag{2}$$

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 convolutional networks* (GCNs), defined by:

$$\mathbf{x}_i^{(k)} = \sigma \left(\mathbf{W}^{(k)} \sum_{v \in \mathcal{N}(i) \cup \{i\}} \frac{\mathbf{x}_v}{\sqrt{|\mathcal{N}(i)||\mathcal{N}(v)|}} \right) \tag{3}$$

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

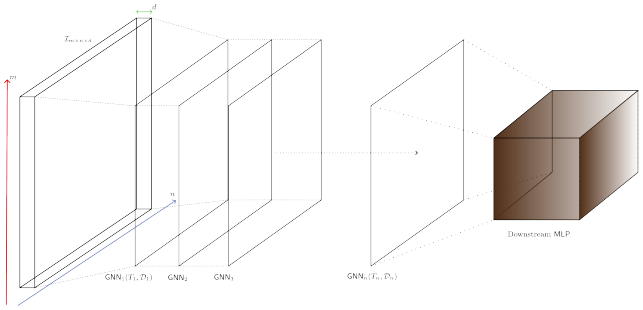

Figure 3: A conceptual illustration on the running flow of an n-layer GNN on particular structure of interest. Note that the data section itself has particular embedding structure on its own.

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.

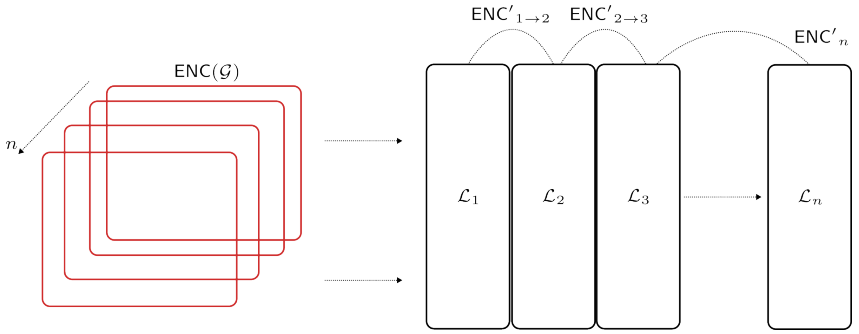

Figure 4: Encoding space and the change of encoding inbetween a static graph (or snapshot-wise) GNN. The encoding space is warped toward arbitrary notions inside a GNN, to the point that after certain layers of processing, the encoding outputs different from expected encoding space (native to the model, not to the designer), though there might be universal notions preserved, like the degree of node represented in a different way.

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.

3.1 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 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.

3.2 Testing scenario

Within the system of graph neural network and graph data, we can form our testing setting and what to consider, beside established notions and goals gained from preliminary experiments. This includes the formulation of the specific **grpah problem**.

Definition 3.2 (Graph-theoretical problem)**.** *Given a graph $G(V, E, \psi_E)$ for ψ_E the edge proprietary classification, there exists an encoding space $\mathsf{ENC}(G)$ that the graph lives in, and a function $\phi : (E, \psi_E) \to (E', \psi'_E)$ such that to change the configuration of the connector space. The **learning problem** is for a learner $L(H, M, C)$ to learn a function ϕ that is appropriate of the intended use case, such that for any given data $D(V^*, \odot)$, either:*

1. *Assign E^* and transform to ψ^*_E.*
2. *For existing E^*, transform to ψ^*_E.*

All within marginal error evaluation of L.

The learner, $L(H, M, C)$, consists of the hypothesis - or rather, pattern memorial H contains the learner's perception of the problem, the **memory** M consists of experiences (in a sense, somewhat similar to recurrent actions), and C is the various configurations available. Theoretically, there are several problems, but the main two problems being the following:

Question 3.1. *Given a learner $L(H, M, C)$ on graph problem,*

1. *For any given learner L, can there exists any given any class of graph that it is not able to learn?*
2. *For a learner L, assume the coherent representability of the graph pattern H. How many counter M there are, or iterative sequence that is needed, so that for $\epsilon > 0$, and $\delta > 0$, then*

$$\Pr_{x,y \in D}[(L(x) \neq y) < 1 - \epsilon] < 1 - \delta$$

assuming the usual inference pair $(x, y) \in D$?

REFERENCES

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.

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].

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, pp. 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*, 2000. URL https://api.semanticscholar.org/CorpusID:2063488.

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.

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.

Arthur Jacot, Fran¸cois Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. Kernel view of wide-network behavior.

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.

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

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].

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.

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.

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.

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.