# Statistical Guarantees of Generative Models

김지수 (Jisu KIM)

딥러닝의 통계적 이해 (Deep Learning: Statistical Perspective), 2024년 2학기

This lecture note is minor modification from Namjoon Suh, Guang Cheng, A Survey on Statistical Theory of Deep Learning: Approximation, Training Dynamics, and Generative Models, 2024. https://arxiv.org/abs/2401.07187/

## 1 Review

### 1.1 Basic Model for Supervised Learning

- Input(입력) / Covariate(설명 변수) : $x \in \mathbb{R}^d$, so $x = (x_1, \dots, x_d)$.
- Output(출력) / Response(반응 변수): $y \in \mathcal{Y}$. If $y$ is categorical, then supervised learning is "classification", and if $y$ is continuous, then supervised learning is "regression".
- Model(모형) :
$$y \approx f(x).$$
If we include the error $\epsilon$ to the model, then it can be also written as
$$y = \phi(f(x), \epsilon).$$
For many cases, we assume additive noise, so
$$y = f(x) + \epsilon.$$
- Assumption(가정): $f$ belongs to a family of functions $\mathcal{M}$. This is the assumption of a model: a model can be still used when the corresponding assumption is not satisfied in your data.
- Loss function(손실 함수): $\ell(y, a)$. A loss function measures the difference between estimated and true values for an instance of data.
- Training data(학습 자료): $\mathcal{T} = \{(y_i, x_i), i = 1, \dots, n\}$, where $(y_i, x_i)$ is a sample from a probability distribution $P_i$. For many cases we assume i.i.d., or $x_i$'s are fixed and $y_i$'s are i.i.d..
- Goal(목적): we want to find $f$ that minimizes the expected prediction error,
$$f^0 = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{(Y,X)\sim P}\left[\ell(Y, f(X))\right].$$
Here, $\mathcal{F}$ can be different from $\mathcal{M}$; $\mathcal{F}$ can be smaller then $\mathcal{M}$.
- Prediction model(예측 모형): $f^0$ is unknown, so we estimate $f^0$ by $\hat{f}$ using data. For many cases we minimizes on the empirical prediction error, that is taking the expectation on the empirical distribution $P_n = \frac{1}{n}\sum_{i=1}^n \delta_{(Y_i, X_i)}$.
$$\hat{f} = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{P_n}\left[\ell(Y, f(X))\right] = \arg\min_{f \in \mathcal{F}} \frac{1}{n}\sum_{i=1}^n \ell(Y_i, f(X_i)).$$
- Prediction(예측): if $\hat{f}$ is a predicted function, and $x$ is a new input, then we predict unknown $y$ by $\hat{f}(x)$.

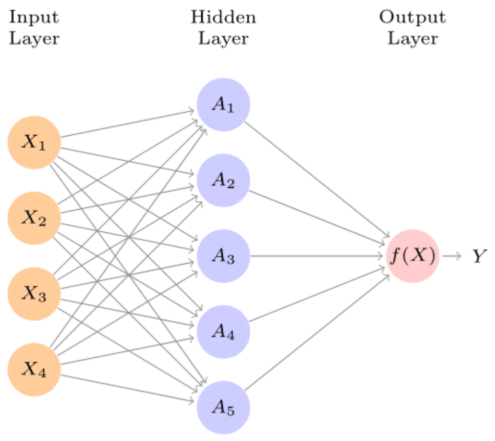

Figure 1: Neural network with a single hidden layer. The hidden layer computes activations $A_j = \sigma_j(x)$ that are nonlinear transformations of linear combinations of the inputs $x_1, \dots, x_d$. Hence these $A_j$ are not directly observed. The functions $\sigma_j$ are not fixed in advance, but are learned during the training of the network. The output layer is a linear model that uses these activations $A_j$ as inputs, resulting in a function $f(x)$. Figure 10.1 from [5].

## 1.2 Two Layer Neural Networks

A two-layer neural network takes an input vector of $d$ variables $x = (x_1, x_2, \dots, x_d)$ and builds a nonlinear function $f(x)$ to predict the response $y \in \mathbb{R}^D$. What distinguishes neural networks from other nonlinear methods is the particular structure of the model:

$$f(x) = f_\theta(x) = g\left(\beta_0 + \sum_{j=1}^{m} \beta_j \sigma(b_j + w_j^\top x)\right),$$

where $x \in \mathbb{R}^d, b_j \in \mathbb{R}, w_j \in \mathbb{R}^d, \beta_0 \in \mathbb{R}^D, \beta_j \in \mathbb{R}^D$. See Figure 1.

- $\theta = \{[\beta, a_j, b_j, w_j] : j = 1, \dots, m\}$ denotes the set of model parameters.
- $x_1, \dots, x_d$ together is called an input layer.
- $A_j := \sigma_j(x) = \sigma(b_j + w_j^\top x)$ is called an activation.
- $A_1, \dots, A_m$ together is called a hidden layer or hidden unit; $m$ is the number of hidden nodes.
- $f(x)$ is called an output layer.
- $g$ is an output function. Examples are:
  - softmax $g_i(x) = \exp(x_i)/\sum_{l=1}^{D} \exp(x_l)$ for classification. The softmax function estimates the conditional probability $g_i(x) = P(y = i|x)$.
  - identity/linear $g(x) = x$ for regression.
  - threshold $g_i(x) = I(x_i > 0)$
- $\sigma$ is called an activation function. Examples are:
  - sigmoid $\sigma(x) = 1/(1 + e^{-x})$ (see Figure 2)
  - rectified linear (ReLU) $\sigma(x) = \max\{0, x\}$ (see Figure 2)
  - identity/linear $\sigma(x) = x$
  - threshold $\sigma(x) = I(x > 0)$, threshold gives a direct multi-layer extension of the perceptron (as considered by Rosenblatt).

Activation functions in hidden layers are typically nonlinear, otherwise the model collapses to a linear model. So the activations are like derived features - nonlinear transformations of linear combinations of the features.

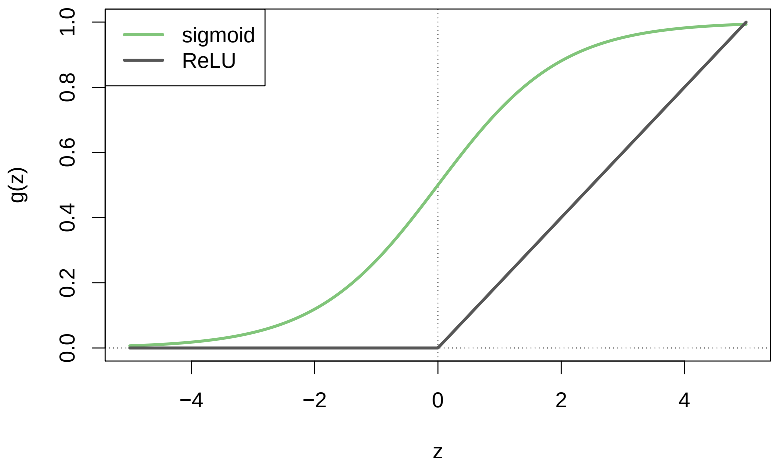

Figure 2: Activation functions. The piecewise-linear ReLU function is popular for its efficiency and computability. We have scaled it down by a factor of five for ease of comparison. Figure 10.2 from [5].

## 2 Notation and Goal

From here, we only consider regression problem, so $g(x) = x$. We assume $\beta_0 = 0$. Hence, for the two-layer neural network with the width of the hidden layer $m$ and activation function $\sigma$, the function space we consider is

$$\mathcal{F}_{m,\sigma} = \left\{ f_\theta : f_\theta(x) = \sum_{j=1}^{m} \beta_j \sigma(b_j + w_j^\top x) \right\},$$

and if we consider all two-layer neural network with arbitrary width, then

$$\mathcal{F}_\sigma = \bigcup_{m=1}^{\infty} \mathcal{F}_{m,\sigma} = \left\{ f_\theta : f_\theta(x) = \sum_{j=1}^{m} \beta_j \sigma(b_j + w_j^\top x),\, m \in \mathbb{N} \right\}.$$

In this note, we sequentially explore the statistical literature on three topics: Generative Adversarial Networks (GANs), diffusion models, and in-context learning in large language models (LLMs).

## 3 Generative Adversarial Networks (GAN)

Over the past decade, Generative Adversarial Networks (GANs) [4] have stood out as a significant unsupervised learning approach, known for their ability to learn the data distribution and efficiently sample the data from it. The main goal of GAN is to learn the target distribution $X \sim \nu$ through the adversarial training between a Discriminator and a Generator. Here, the generator takes the input $\mathcal{Z}$ from prior distributions such as Gaussian or Uniform (i.e., $\mathcal{Z} \sim \pi$) and the input is push-forwarded by transformation map $g : \mathcal{Z} \to g(\mathcal{Z})$. In this seminal paper, the distribution of random variable $g(\mathcal{Z})$ is referred to as an implicit distribution $\mu$. The primary objective of the generator is to produce synthetic data from µ that closely resembles samples from the target distribution.

The adversarial training between the discriminator and generator is enabled through optimizing the following minimax problem w.r.t. functions in the generator class $\mathcal{G}$ and discriminator class $\mathcal{F}$:

$$(g^*, f^*) \in \text{argmin}_{g \in \mathcal{G}} \text{argmax}_{f \in \mathcal{F}} \left\{ \mathbb{E}_{\mathcal{Z}\sim\pi} f(g(\mathcal{Z})) - \mathbb{E}_{\mathcal{X}\sim\nu} f(y) \right\}. \tag{1}$$

In practice, the above expectations can be approximated with $m$ training data from $\nu$ and $n$ drawn samples from $\mu$. The inner maximization problem in (1) is an Integral Probability Metric (IPM), which quantifies the discrepancy between two distributions $\mu$ and $\nu$ w.r.t. a symmetric function class, i.e., $f \in \mathcal{F}$, then $-f \in \mathcal{F}$;

$$d_{\mathcal{F}}(\mu, \nu) = \sup_{f \in \mathcal{F}} \left\{ \mathbb{E}_{x\sim\mu} f(x) - \mathbb{E}_{y\sim\nu} f(y) \right\}.$$

When $\mathcal{F}$ is taken to be all 1-Lipschitz functions, $d_{W_1}(\cdot,\cdot)$ is the Wasserstein-1 distance; when $\mathcal{F}$ is the class of all indicator functions, $d_{TV}(\cdot,\cdot)$ is the total variation distance; when $\mathcal{F}$ is taken as a class of neural networks, $d_{NN}(\cdot,\cdot)$ is the "neural net distance". Under this setting, a generator from $\mathcal{G}$ attempts to minimize the IPM between $\mu$ and $\nu$.

Note that the functions in discriminator and generator classes can be either classical non-parametric regressors such as random forests, local polynomial, etc., or can be both parametrized by neural networks. Here, we focus on the latter case which is more commonly used in practice. Specifically, let us denote $\mathcal{F} := \{f_{w=\omega}(\cdot) : \mathbb{R}^d \to \mathbb{R}\}$ as the Discriminator class and $\mathcal{G} := \left\{g_\theta(z) : \mathbb{R}^m \to \mathbb{R}^d\right\}$ (with $m \leq d$) as the Generator class with $\omega$ and $\theta$ being denoted as network parameters of respective classes, and we are interested in estimating the parameters by minimizing following optimization problem:

$$(\hat{\theta}_{m,n}, \hat{\omega}_{m,n}) \in \text{argmin}_{\theta: g_\theta \in \mathcal{G}} \text{argmax}_{\omega: f_\omega \in \mathcal{F}} \left\{ \hat{\mathbb{E}}_m f_\omega(g_\theta(\mathcal{Z})) - \hat{\mathbb{E}}_n f_\omega(X) \right\}, \tag{2}$$

where $\hat{\mathbb{E}}_m$ (resp. $\hat{\mathbb{E}}_n$) denotes the empirical expectation with $m$ generator samples. (resp. $n$ traning samples.)

Given that the optimal parameters of generator $\hat{\theta}_{m,n}$ in (2) can be obtained, [8] studied how well the implicit distribution estimator $\mu_{\hat{\theta}_{m,n}}$ (i.e., distribution of random variable $g_{\hat{\theta}_{m,n}}(\mathcal{Z})$ is close to the target distribution $\nu$ in a Total Variation distance. Under some regularity assumptions on architectures of $g_\theta$ and $f_\omega$, Theorem 19 in [8] proved the existence of $(g_\theta(z), f_\omega)$ pairs satisfying the bound:

$$\mathbb{E} d_{TV}^2(\nu, \mu_{\hat{\theta}_{m,n}}) \leq \sqrt{d^2 K \log(dK) \left( \frac{\log m}{m} \vee \frac{\log n}{n} \right)}. \tag{3}$$

In the rate (3), $K$ and $d$ are the depth and width of the generator networks, respectively. This result allows the very deep network as $K \leq \sqrt{(n \wedge m)/\log(n \vee m)}$. It is worth noting that the generator requires the width of the network to be the same as the input dimension $d$ so that the invertibility condition on the generator is satisfied. As for the discriminator, it can be constructed by concatenating two networks that have the same architecture as the one from the generator, and with the additional two layers, i.e., network $f_\omega$ has $K+2$ layers. The $g_\theta$ and $f_\omega$ used leaky ReLU and dual leaky ReLU as activation functions respectively for their invertibility. However, this invertibility condition is often violated in practical uses of GANs.

## 4 Score-based diffusion models

Score-based diffusion model consists of two processes. The first step, forward process, transforms data into noise. Specifically, the score-based diffusion model [10] uses the following stochastic differential equations (SDEs) for data perturbation:

$$dx = f(x,t)dt + g(t)dW_t, \tag{4}$$

where $f(x,t)$ and $g(t)$ are drift and diffusion coefficient, respectively, and $W_t$ is a standard Wienerprocess (a.k.a. Brownian Motion) indexed by time $t \in [0,T]$. Here, the $f$ and $g$ functions are user specified.

Allowing diffusion to continue long enough with $T$ being sufficiently large, it can be shown the distribution of $x_t$ converges to some easy-to-sample distributions such as normal or uniform distributions. Specifically, when $f := -x_t$ and $g := \sqrt{2}$, (4) is known as Ornstein-Uhlenbeck (OU) process, it is proven $p_t := \text{Law}(x_t) \to \pi$ with $\pi$ being normal, exponentially fast in 2-Wasserstein distance. However, despite this convergence result, it is analytically difficult to know the exact form of $p_T$, and it is often replaced by the $\pi$ in practice when starting the reverse process.

The second step, reverse process, is a generative process that reverses the effect of the forward process. This process learns to transform the noise back into the data by reversing SDEs in (4). Through the Fokker-Planck equation of marginal density $p_t(x)$ for time $t \in [t_0, T]$, the following reverse SDE [Anderson, 1982] can be easily derived:

$$dx = \left[f(x,t) - g(t)^2 \nabla_x \log p_t(x)\right] dt + g(t) d\bar{W}_t. \tag{5}$$

Here, the gradient of $\log p_t(x)$ w.r.t to the perturbed data $x(t)$ is referred to as score function, $dt$ in (5) is an infinitesimal negative time step, and $d\bar{W}_t$ is a Wiener-Process running backward in time, i.e., $t : T \to t_0$. In practice, $t_0$ is usually chosen to be a small number close to 0, but not too close to 0 to prevent the blow-up of the score function. There are various ways to solve (5); for instance, the discretization scheme such as Euler-Maruyama, or the theory-driven method such as probability-flow.

The score function, $\nabla_x \log p_t(x)$, is approximated by a time-dependent score-based model $S_\theta(x(t), t)$ which is parametrized by neural networks in practice. The network parameter $\theta$ is estimated by minimizing the following score-matching loss:

$$\theta^\star := \text{argmin}_\theta \mathbb{E}_{t\sim\mathcal{U}[t_0,T]}\mathbb{E}_{x(t)|x(0)}\mathbb{E}_{x(0)}\left[\lambda(t)^2 \left\|S_\theta(x(t), t) - \nabla_x \log p_t(x(t)|x(0))\right\|_2^2\right],$$

where $\mathcal{U}[t_0, T]$ is a uniform distribution over $[t_0, T]$, and $\lambda(t)(> 0)$ is a positive weighting function that helps the scales of matching-losses $\|S_\theta(x(t), t) - \nabla_x \log p_t(x(t)|x(0))\|_2^2$ to be in the same order across over the time $t \in [t_0, T]$. The transition density $p_t(x(t)|x(0))$ is a tractable Gaussian distribution, and x(t) can be obtained through ancestral sampling.

Under this setting, consider the following theoretical questions :

Q1 Can the diffusion model estimate the target distribution $\nu$ via the learned score function? If so, under which conditions on $\nu$, do we have the polynomial convergence guarantees in generalization error bound $\epsilon$ measured in TV or Wasserstein distances?

Q2 Do neural networks well approximate and learn the score functions? If so, how one should choose network architectures and what is the sample complexity of learning? Furthermore, if the data distribution has a special geometric structure, is the diffusion model adaptive to the structure, just as GAN models do?

The main statistical object of interest in these two lines of research is the generalization bound measuring the distance between target distribution $\nu$ and estimated distribution $\hat{\mu}_\theta$ from the samples $x_{t_0}$ by solving the reverse SDE in (5). Here, the score function is substituted by the estimated timedependent neural network $S_\theta(x(t), t)$. The first line of work mainly focuses on the sampling perspective of the diffusion model, given that we have good estimates of the score function. The second line of work extends the attention to the score function approximation through neural networks. Furthermore, under some highly stylized settings, they specify the explicit network structures which give good generalization guarantees.

For Q1, [6] first gives the polynomial guarantees in TV distance under $L^2$-accurate score for a reasonable family of distributions:

Under appropriate setup, for small enough $\epsilon_{TV}$ and if $\epsilon = O\left(\epsilon_{TV}^4\right)$, then with appropriate choices of $T$, correct step sizes $h_m$ and predictor step size $h$, produces a sample from a distribution $q_T$ such that

$$TV(q_T, p_{data}) < \epsilon_{TV}.$$

However, their result is based on the assumption that the distribution meets certain smoothness criteria and the log-Sobolev inequality, which essentially confines the applicability of their findings to distributions with a single peak. Recently, two works [7], [2] try to avoid the strong assumptions on the data distributions and to get the polynomial convergence guarantees under general metrics such as TV or Wasserstein. Specifically, [7] give 2-Wasserstein bounds for any distributions with bounded support. The results they provide have polynomial complexity guarantees without relying on the functional inequality on distributions such as log-Sobolev inequality. They further give TV bounds with polynomial complexity guarantees under the Hessian availability assumption. Similarly with [6], under general data distribution assumption, i.e., second moment bound of  and L-Lipschtizness of score function, [2] give the polynomial TV convergence guarantee, where only $\tilde{\Theta}\left(\frac{L^2 d}{\epsilon^2}\right)$ discretization is needed. Here, $\epsilon$ is a TV generalization error, $d$ being a data dimension.

Due to its recent theoretical advancements, the list of research attacking Q2 is short in the literature. One work [1] proved that the diffusion model is adaptive to estimating the data distribution supported in a lower-dimensional subspace. They design a very specific network architecture for $S_\theta(x(t), t)$ with an encoder-decoder structure and a skip-connection. Under a more general setting, another work [9] proves the distribution estimator from the diffusion model can achieve nearly minimax optimal estimation rates. Specifically, they assume the true density is supported on $[-1, 1]^d$, in the Besov space with a smooth boundary. The Besov space unifies many general function spaces such as Hölder, Sobolev, continuous, or even non-continuous function classes. The result in [9] is valid for the non-continuous function class, and this should be contrasted with the aforementioned works [7, 2] where they assume the Lipschitzness of the score function.

## 5 In-Context Learning in Large Language Model

We provide recent theoretical understandings of an interesting phenomenon observed in LLM, i.e., In-Context Learning (ICL). It refers to the ability of LLMs conditioned on prompt sequence consisting of examples from a task

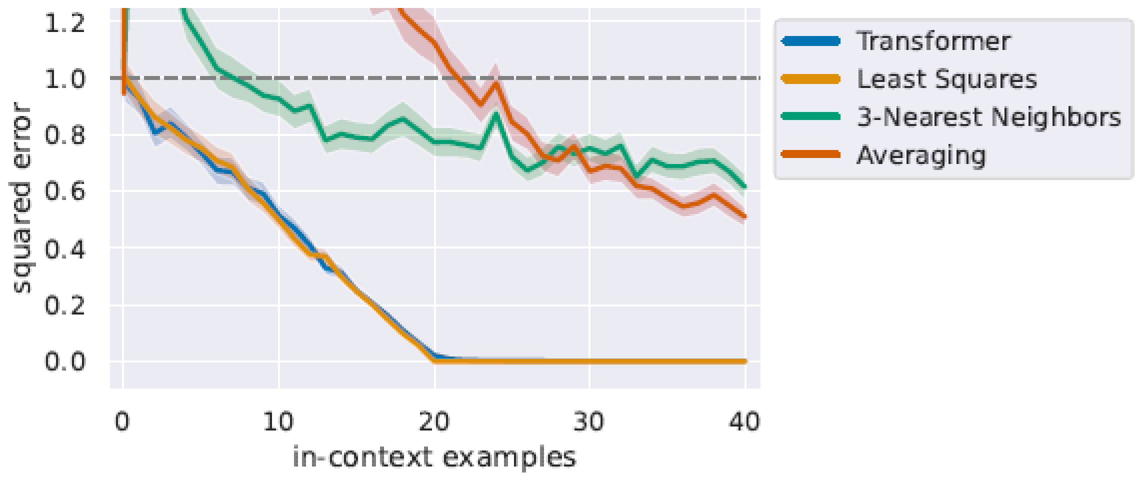

Figure 3: Evaluating the trained Transformer on in-context learning linear functions. This is the plot of the normalized squared error of the Transformer ($(M(P) - w^\top x_{\text{query}})^2/d$), along with the relevant baselines, as a function of the number of in-context examples. Figure 2 from [3].

(input-output pairs) along with the new query input, the LLM can generate the corresponding output accurately. An instance taken from Garg et al. [2022], these models can produce English translations of French words after being prompted on a few such translations,

$$\underbrace{\text{maison} \to \text{house}, \quad \text{chat} \to \text{cat}, \quad \text{chien} \to}_{\text{prompt}} \quad \underbrace{\text{dog}}_{\text{completion}} .$$

This capability is quite intriguing as it allows models to adapt to a wide range of downstream tasks on-the-fly without the need to update the model weights after training. Readers can refer to the backbone architecture of LLM (i.e., Transformer) in the seminal paper .

Toward further understanding ICL, researchers formulated a well-defined problem of learning a function class $\mathcal{F}$ from in-context examples. Formally, let $\mathcal{D}_{\mathcal{X}}$ be a distribution over inputs and $\mathcal{D}_{\mathcal{F}}$ be a distribution over functions in $\mathcal{F}$. A prompt $P$ is a sequence $(x_1, f(x_1), \ldots, x_k, f(x_k), x_{\text{query}})$ where inputs (i.e., $x_i$ and $x_{\text{query}}$) are drawn i.i.d. from $\mathcal{D}_{\mathcal{X}}$ and $f$ is drawn from $\mathcal{D}_{\mathcal{F}}$. In the above example, it can be understood $\{(x_1, f(x_1)), (x_2, f(x_2))\} := \{(\text{maison}, \text{house}), (\text{chat}, \text{cat})\}$, $x_{\text{query}} = \text{chien}$, and $f(x_{\text{query}}) = \text{dog}$. Now, we provide the formal definition of in-context learning.

**Definition 1.** Model $M_\theta$ can in-context learn the function class $\mathcal{F}$ up to $\epsilon$, with respect to $(\mathcal{D}_{\mathcal{F}}, \mathcal{D}_{\mathcal{X}})$, if it can predict $f(x_{\text{query}})$ with an average error

$$\mathbb{E}_{P\sim(x_1, f(x_1), \ldots, x_k, f(x_k), x_{\text{query}})} \left[\ell(M_\theta(P), f(x_{\text{query}}))\right] \le \epsilon,$$

where $\ell(\cdot, \cdot)$ is some appropriate loss function, such as the squared error.

The paper [3] empirically investigated the ICL of Transformer architecture [11] by training the model $M_\theta$ on random instances from linear functions, two-layer ReLU networks, and decision trees. Specifically, they showed the predictions of Transformers on the prompt $P$ behave similarly to those of ordinary least-squares (OLS), when the models are trained on the instances from linear function classes $\mathcal{F}^{\text{Lin}} := \left\{f : f(x) = w^\top x, w \in \mathbb{R}^d\right\}$ for random weights $w \sim \mathcal{N}(0, I_d)$, see Figure 3. A similar phenomenon was observed for the models trained on sparse linear functions as the predictions behave like those of Lasso estimators.

## References

[1] Minshuo Chen, Kaixuan Huang, Tuo Zhao, and Mengdi Wang. Score approximation, estimation and distribution recovery of diffusion models on low-dimensional data. In Andreas Krause, Emma Brunskill, Kyunghyun

Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 4672–4712. PMLR, 2023.

[2] Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru Zhang. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[3] Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What can transformers learn in-context? A case study of simple function classes. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[4] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014.

[5] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning—with applications in R*. Springer Texts in Statistics. Springer, New York, [2021] ©2021. Second edition [of 3100153].

[6] Holden Lee, Jianfeng Lu, and Yixin Tan. Convergence for score-based generative modeling with polynomial complexity. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[7] Holden Lee, Jianfeng Lu, and Yixin Tan. Convergence of score-based generative modeling for general data distributions. In Shipra Agrawal and Francesco Orabona, editors, *International Conference on Algorithmic Learning Theory, February 20-23, 2023, Singapore*, volume 201 of *Proceedings of Machine Learning Research*, pages 946–985. PMLR, 2023.

[8] Tengyuan Liang. How well generative adversarial networks learn distributions. *J. Mach. Learn. Res.*, 22:228:1–228:41, 2021.

[9] Kazusato Oko, Shunta Akiyama, and Taiji Suzuki. Diffusion models are minimax optimal distribution estimators. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 26517–26582. PMLR, 2023.

[10] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.