이학박사학위논문

Information Theoretic Analysis of Machine Learning

기계 학습의 정보 이론적 해석

2022 년 2 월

서울대학교 대학원

물리천문학부

이 성 엽

이학박사학위논문

Information Theoretic Analysis of Machine Learning

기계 학습의 정보 이론적 해석

2022 년 2 월

서울대학교 대학원

물리천문학부

이 성 엽

Information Theoretic Analysis
of Machine Learning

기계 학습의 정보 이론적 해석

지도교수 김 석

이 논문을 이학박사 학위논문으로 제출함

2022 년 2 월

서울대학교 대학원

물리천문학부 물리학전공

이 성 엽

이성엽의 이학박사 학위논문을 인준함

2021 년 12 월

위 원 장	이원종	(인)
부위원장	김 석	(인)
위 원	백용주	(인)
위 원	조정효	(인)
위 원	현창봉	(인)

Abstract

Information Theoretic Analysis of Machine Learning

Sungyeop Lee
Department of Physics and Astronomy
The Graduate School
Seoul National University

I aim to provide a deeper understanding of how machine learning works so great for solving various problems. Using information theory, I study the information flow, internal representations, and parameter optimization of neural networks. First, I visualize the compression and transmission of information flows of various types of autoencoders, and examine how the various models remove irrelevant information to reproduce input data. Second, I observe that the internal representation of neural networks is so special that the frequency of distinct representations follows scale-invariant power laws both in supervised and unsupervised learning. I derive how this universal behavior can naturally arise without explicit regularization during the learning process. Finally, I introduce the mirror descent algorithm in terms of information geometry, and explain how the learning algorithm can effectively update the parameters of learning models in the dual space of the primary parameter space. In conclusion, information theory and geometry are excellent tools to visualize, analyze, and optimize neural networks.

Keywords: Machine learning, Deep learning, Information theory, Information geometry, Statistical physics
Student Number: 2012-23096

Contents

Chapter 1

Introduction

Recently, machine learning shows successful applications in various fields. In particular, as data includes more complex patterns, deep learning is more powerful for the data analysis than traditional statistical models. Deep learning approximates mapping between input and output as nonlinear transformations using numerous parameters. Currently, deep learning can be technically trained using advanced optimization methods, but there is still a lack of understanding on how to interpret the inside of deep learning. In order to develop advanced machines that can be explained through principle-based learning, it is necessary to understand the inside of the machine. In this thesis, I analyze various phenomena that arise in the hidden latent and parameter space of deep learning in terms of information theory and statistical physics.

Information theory is an excellent tool to examine the learning process in deep neural networks as information transmission and compression of data. The information flows can be visualized on the information plane of the mutual

information among input, hidden, and output layers. I examine how the information flows are shaped by the network architectures, such as depth, sparsity, weight constraints, and hidden representations. I adopt autoencoders (AEs) as a model of deep learning, because they have clear guidelines for their information flows, and various species, such as vanilla, sparse, tied, variational, and label AEs. I measured their information flows using Rényi's matrix-based α-order entropy functional. As learning progresses, they show a typical fitting phase where the amounts of input-to-hidden and hidden-to-output mutual information both increase. In the last stage of learning, however, some AEs show a compression phase, where input-to-hidden mutual information diminishes. I derived the physical meaning of the compression phase through the information plane analysis, and visualized the different patterns of information flows according to the characteristics of AE variants.

Information bottleneck theory has proposed that neural networks are effective information compressors. Motivated from this theory, I examined the internal representations of neural networks. When input data is converted into internal representations in the learning process, similar data have close internal codes. It has been observed that data clustering based on the shared code follows the power law. This scale-invariant distribution implies that machine learning largely compresses frequent typical data, and at the same time, differentiates many atypical data as outliers. The scale invariance was explained in the previous research in the case of unsupervised learning. I extend this finding into supervised and self-supervised learning. Furthermore, I theoretically derive how the power laws can naturally arise in all types of machine learning. In terms of information theory, the scale-invariant representation corresponds to a maximally uncertain data grouping among possible representations that guarantee pre-specified learning accuracy. It is experimentally verified that this

phenomenon can emerge robustly regardless of the model architecture and training time. In addition, I investigated how this result is affected by input data distributions by using two-dimensional equilibrium Ising spin data.

The above studies focus on the representation of neural activation, while the following study is an analysis of model parameters in the learning process. Training deep neural networks is a complex optimization problem in the geometry of the loss landscape. Various optimization techniques based on gradient descent (GD) have been developed to efficiently find the global minimum of the loss function. In particular, mirror descent (MD), a generalization of GD, not only has a profound background from the information geometry perspective, but also technically implements the implicit regularization. I experimentally validate the performance of MD when training the overparameterized model where the number of parameters is greater than that of sample numbers. It is confirmed that when optimal MD corresponding to the model architectures is applied, training performance is remarkably enhanced and even better than momentum-based adaptive algorithm. By examining the update process of MD, I identify the potential to treat the gradient vanishing problem with the suitable choice of convex function empirically adopted in MD.

Table 1.1 Abbreviations used in this thesis.

AE	Autoencoder
BCE	Binary Cross Entropy
CE	Cross Entropy
DPI	Data Processing Inequality
ELBO	Evidence Lower Bound
GD	Gradient Descent
IB	Information Bottleneck
IP	Information Plane
KL	Kullback-Leiber
LAE	Label Autoencoder
MD	Mirror Descent
MSE	Mean Squared Error
PCA	Principal Component Analysis
RBMs	Restricted Boltzmann Machines
RKHS	Reproducing Kernel Hilbert Space
SAE	Sparse Autoencoder
TAE	Tied Autoencoder
VAE	Variational Autoencoder

Chapter 2

Information flows of machine learning

2.1 Information plane analysis

Since the development of information theory as a theory of communication by Shannon [1, 2], it has played a crucial role in various domains of engineering and science, including physics [3], biology [4], and machine learning [5]. The information bottleneck (IB) theory interprets the learning process of neural networks as the transmission and compression of information [6]. Neural networks encode input X into internal representation Z; then, they decode Z to predict the desired output Y. The IB theory is a rate-distortion theory that compresses irrelevant information in X for predicting Y to the maximum extent. The objective function of this theory can be mathematically described as minimizing the mutual information $I(X; Z)$ between X and Z, given a required

transmission I_{req} of the mutual information $I(Z;Y)$ between Z and Y:

$$\min_{p(Z|X)} I(X;Z) - \beta\Big[I(Z;Y) - I_{req}\Big], \tag{2.1}$$

where β is a trade-off coefficient that balances the information compression and transmission. The numerical method for solving this optimization problem, so called Blahut–Arimoto algorithm [7, 8], has been extensively studied [9, 10]. Theoretical aspects of IB theory and its applications are well summarized in [11].

Here, it is important to note that machine learning models, including ours in this study, do not take Equation (2.1) as the loss function, although new deep variational IB models directly adopt it as the loss function [12]. The IB theory provides a nice interpretation of learning process, but it does not work for the optimization of neural networks in general. The mutual information provides a potent tool for visualizing the learning processes by displaying a trajectory on the two-dimensional plane of $I(X;Z)$ and $I(Z;Y)$, called the information plane (IP). Through IP analyses, Shwartz-Ziv and Tishby found that the training dynamics of neural networks demonstrate a transition between two distinct phases: fitting and compression [13,14]. Supervised learning experiences a short fitting phase in which the training error is significantly reduced. This first phase is characterized by increases in $I(X;Z)$ and $I(Z;Y)$. Then, in the learning process, a large amount of time is spent on finding the efficient internal representation Z of input X when the fitting phase secures a small training error. During this second phase of compression, $I(X;Z)$ decreases while $I(Z;Y)$ remains constant. To avoid unnecessary confusion with the usual data compression or dimensionality reduction, henceforth, I denote the second phase as "simplifying phase" instead of the original name of "compression phase".

The simplifying phase associated with the generalization ability of machine by compressing irrelevant information of training data to prevent overfit-

ting [14]. The non-trivial simplifying phase and its association with generalization have been further observed in other studies using different network models with different data; however, the universality of the simplifying phase remains debatable [15–17]. The debates can be partly attributed to the sensitivity toward the architecture of neural networks, activation functions, and estimation schemes of information measures.

I investigate how the information flows are shaped by the network designs, such as depth, sparsity, weight constraints, and hidden representations, by using autoencoders (AEs) as specific models of machine learning. AEs are neural networks that encode input X into internal representation Z and reproduce X by decoding Z. This representation learning can be interpreted as self supervised learning where a label is input as itself, such as $Y = X$. To examine the IP analyses of representation learning, I considered AEs because (i) they have a concrete guide ($Y = X$) for checking the validity of $I(X; Z)$ and $I(Z; Y)$ on the IP, (ii) they have various species to fully explore trajectories on the IP, and (iii) they are closely related to unsupervised learning.

The remainder of this chapter is organized as follows. I introduce various types of AEs in Section 2.2 and explain our matrix-based kernel method for estimating mutual information in Section 2.3. Then, I examine the IP trajectories of information transmission and compression of the AEs in Section 2.4. Finally, I summarize and discuss our findings in Section 2.5.

2.2 Representation learning in autoencoders

2.2.1 Information plane of autoencoders

AEs are neural networks specialized for dimensional reduction and representation learning in an unsupervised manner. A deep AE consists of a symmetric structure with encoders and decoders as follows:

$$X - E_1 - \cdots - E_L - Z - D_1 - \cdots - D_L - X'. \tag{2.2}$$

where E_i and D_i denote the i-th encoder and decoder layer, respectively, and Z is the bottleneck representation with the smallest dimension. The deep AE trains an identity function to reproduce input X from output X'. During the training process, the AE extracts relevant features for reproducing X while compressing the high-dimensional input X into an internal representation Z on a low-dimensional bottleneck layer. The encoder and decoder layers form Markov chains that should satisfy the data processing inequality (DPI), analogously to supervised learning [18]:

$$\text{Forward DPI:} \qquad I(X;E_1) \geq \cdots \geq I(X;E_L) \geq I(X;Z), \tag{2.3}$$

$$\text{Backward DPI:} \qquad I(Z;X') \leq I(D_1;X') \leq \cdots \leq I(D_L;X'). \tag{2.4}$$

The forward DPI represents information compression as input X is processed into the bottleneck layer, whereas the backward DPI represents information expansion as the compressed representation Z is transformed into output X'. It is noteworthy that the usual AEs have physical dimensions, narrowing toward the bottleneck and expanding away from the bottleneck, which are consistent with the DPI.

The desired output of this AE is identical to the input ($X' = X$). This identity constrains the input and output mutual information to be located

on a straight line $I(X;T) = I(T;X)$ for arbitrary internal representations, $T \in \{E_1, \cdots, E_L, Z, D_1, \cdots, D_L\}$. Here, if the desired output X in $I(T;X)$ is replaced with the predicted output X' of the AE, the learning dynamics of the AE on the IP can be analyzed [18]. Then, the two sets of mutual information for representing information compression and transmission correspond to

$$I(X;T) \quad = H(T) - H(T|X) \tag{2.5}$$

$$I(T;X') \quad = H(T) - H(T|X'), \tag{2.6}$$

where $H(T)$ represents the Shannon entropy of T, and $H(T|X)$ and $H(T|X')$ are the conditional entropies of T given X and X', respectively. The forward process of the AE realizes the deterministic mapping of $T = T(X)$ and $X' = X'(T)$. Then, one-to-one correspondence of $X \to T$ implies no uncertainty for $H(T(X)|X) = 0$, whereas the possibly many-to-one correspondence of $T \to X'$ implies some uncertainty for $H(T|X'(T)) \neq 0$. Therefore, the inequality of $I(X;T) \geq I(T;X')$ is evident because $H(T) \geq H(T) - H(T|X')$, where the conditional entropy $H(T|X')$ is non-negative. Based on this inequality, the learning trajectory of $I(X;T)$ and $I(T;X')$ on the two-dimensional IP (x, y) can be expected to stay below the diagonal line $y = x$. Once the learning process of the AE is complete with $X' = X$, the two sets of mutual information become equal to $I(X;T) = I(T;X' = X)$, and the learning trajectory ends up on the diagonal line.

2.2.2 Various types of autoencoders

To investigate information flows of machine learning, I adopted AEs because their theoretical bounds of IP trajectories could guide our IP analysis. IP analysis has been used to visualize the information process in AEs [18,19]. Their IP trajectories satisfied the theoretical boundary of $I(X;T) \geq I(T;X')$. Previous

studies examined IP trajectories according to the size of the given bottleneck layers, but they did not investigate the associations between the simplifying phases and generalizations of AEs. Another important advantage of adopting AEs is their diverse variants that enable us to fully explore IP trajectories depending on the network designs, such as depth, sparsity, weight constraints, and hidden representations. In particular, because a certain AE model is directly linked to unsupervised learning, the model can be used to understand the information process of unsupervised learning. Now, I briefly introduce diverse species of AEs used in our experiments.

The simplest structure of AE, called shallow AE, consists of a single bottleneck layer between the input and output layers. In the shallow AE ($X-Z-X'$), the forward propagation of input X is defined as

$$Z = f_E(W_E X + b_E) \tag{2.7}$$

$$X' = f_D(W_D Z + b_D), \tag{2.8}$$

where W and b represent the weights and biases, respectively, and $f(s)$ is a corresponding activation function. Here, the subscripts E and D denote the encoder and decoder, respectively. The shallow AE is trained to minimize the reconstruction errors usually measured by the mean squared error (MSE) between output X' and desired output X. It has been analytically proven that a shallow AE with linear activation ($f(s) = s$) spans the same subspace as that spanned by principal component analysis (PCA) [20, 21]. Deep AEs stack hidden layers in the encoder and decoder symmetrically; moreover, it is well known that deep AEs yield better compression than shallow AEs.

Right up till recently, a myriad of variants and techniques have been proposed to improve the performance of AEs via richer representations, such as sparse AE (SAE) [22], tied AE (TAE) [23], variational AE (VAE) [24], and label

AE (LAE) [25, 26].

- SAE was proposed to avoid overfitting by imposing sparsity in the latent space. The sparsity penalty is considered a regularization term of the Kullback–Leibler (KL) divergence between the activity of bottleneck layer Z and sparsity parameter ρ, a small value close to zero.

- TAE shares the weights for the encoder and decoder part ($W_E = W_D^T$), where superscript T depicts the transpose of a matrix. This model is widely used to reduce the number of model parameters while maintaining the training performance. Owing to its symmetrical structure, it can be interpreted as a deterministic version of restricted Boltzmann machines (RBMs), a representative generative model for unsupervised learning; consequently, the duality between TAE and RBM has been identified [27]. Compared to the vanilla AE, SAE and TAE have regularizations for the degrees of freedom for nodes and weights, respectively. Later, I visually validate how these constraints lead to a difference in the information flow of IP trajectories.

- The ultimate goal of AEs is to obtain richer expressions in the latent space. Therefore, an AE is not a mere replica model, but a generative model that designs a tangible latent representation to faithfully reproduce the input data as output. VAE is one of the most representative generative models with a similar network structure to AE; however, its mathematical formulation is fundamentally different. The detailed derivation of the learning algorithm of VAE is beyond the scope of this study, and thus it will be omitted [24]. In brief, the encoder network of VAE realizes an approximate posterior distribution $q_\phi(Z|X)$ for variational inference, whereas the decoder network realizes a distribution $p_\theta(X|Z)$ for generation. The loss

of VAE, known as the evidence lower bound (ELBO), is decomposed into a reconstruction error given by the binary cross entropy (BCE) between the desired output X and predicted output X', and the regularization of KL divergence between the approximate posterior distribution $q_\phi(Z|X)$ and prior distribution $p(Z)$. As tangible Gaussian distributions are usually adopted as the approximate posterior and prior distributions of $q_\phi(Z|X)$ and $p(Z)$, respectively, VAE has a special manifold of the latent variable Z.

- AEs do not use data labels. Instead, inputs work as self labels for supervised learning. Here, to design the latent space using label guides, I consider another AE, called label AE (LAE). LAE forces the input data to be mapped into the latent space with the corresponding label classifications. Then, the label-based internal representation is decoded to reproduce input data. Although the concept of regularization using labels has been proposed [25, 26], LAE has not been considered as a generative model. Unlike vanilla AEs that use a sigmoid activation function, LAE uses a softmax activation function, $f(Z_i) = \exp(Z_i)/\sum_j \exp(Z_j)$, to impose the regularization of the internal representation Z to follow the true label Y as the cross entropy (CE) between Y and Z. Once LAE is trained, it can generate learned data or images using its decoder, starting from one-hot vector Z of labels with the addition of noise. Additional details of LAE are provided in Appendix B. Later, I compare the IP trajectories of VAE and LAE with those of vanilla AE in a deep structure to examine how the information flow varies depending on the latent space of generative models.

Table 2.1 summarizes the loss function, constraints, and activation function

of the bottleneck layer for each aforementioned AE model.

Table 2.1 Various species of autoencoder. Vanilla AE uses the mean squared error (MSE) loss. I used a sigmoid function as an activation function for the bottleneck layer, which helps in unifying the scales of different layers. Regularization of SAE is the KL-divergence between the hidden activity and sparsity parameter ρ. The only difference in TAE is that it shares the weight of encoder and decoder. The loss function of VAE, known as the evidence lower bound (ELBO), consists of the reconstruction error, binary cross entropy (BCE), and KL-divergence between the approximate posterior $q_\phi(Z|X)$ and prior $p(Z)$; moreover, the stochastic node activities of the bottleneck layer are sampled from Gaussian distributions. In LAE, the classification error, the cross entropy (CE) between the softmax hidden activity Z and true label Y, is used as a regularization term.

Model	**Main Loss**	**Constraint**	**Bottleneck Activation**
AE	$\mathrm{MSE}(X, X')$	None	sigmoid
SAE	$\mathrm{MSE}(X, X')$	$\mathrm{KL}(\rho\|\|Z)$	sigmoid
TAE	$\mathrm{MSE}(X, X')$	$W_E = W_D^T$	sigmoid
VAE	$\mathrm{BCE}(X, X')$	$\mathrm{KL}(q_\phi(Z\|X)\|\|p(Z))$	Gaussian sampling
LAE	$\mathrm{MSE}(X, X')$	$\mathrm{CE}(Y, Z)$	softmax

2.3 Estimation of mutual information

After preparing various species of AE models to explore diverse learning paths on the IP, I need to estimate the mutual information for IP analyses:

$$I(X;Z) = \sum_{x,z} p(x,z) \log \frac{p(x,z)}{p(x)p(z)}. \tag{2.9}$$

In reality, I have samples of data, $\{x(t), z(t)\}_{t=1}^{N}$, instead of their probabilities, $p(x)$, $p(z)$, and $p(x, z)$. Using N samples of data, I may estimate the probabilities. Here, if X and Z are continuous variables, it is inevitable to first discretize them. Then, I can count the discretized samples for each bin and estimate the probabilities. The estimation of mutual information based on this binning method has some limitations. First, its accuracy depends on the resolution of discretization. Second, large samples are required to properly estimate the probability distributions. Suppose that X is an n-dimensional vector. Despite considering the most naive discretization with binarized activities, the total number of configurations for the binarized X is already 2^n. Thus, it becomes impracticable for N finite samples to cover the full range of configurations, e.g., $2^{20} \approx 10^6$ configurations for $n = 20$. Therefore, other schemes, such as kernel density estimation [28], k-nearest neighbors, and matrix-based kernel estimators [29, 30], exist to estimate the entropy and mutual information. The description of each scheme and the corresponding IP results were presented in a pedagogical review [31]. Among these various methods, I adopted a matrix-based kernel estimator, which is mathematically well defined and computationally efficient for large networks. It estimates the Rényi's α-order entropy using the eigenspectrum of covariance matrix of X as follows:

$$S_\alpha(A) = \frac{1}{1-\alpha} \log_2 \left[\mathrm{tr}(A^\alpha)\right] = \frac{1}{1-\alpha} \log_2 \left[\sum_{i=1}^{N} \lambda_i(A)^\alpha\right], \tag{2.10}$$

where A is an $N \times N$ normalized Gram matrix of random variable X with size N and $\lambda_i(A)$ is the i-th eigenvalue of A. Note that tr denotes the trace of a matrix. In the limit of $\alpha \to 1$, Equation (2.10) is reduced to an entropy-like measure that resembles the Shannon entropy of $H(X)$. If I assume that B is a normalized Gram matrix from another random variable Z, the joint entropy

between X and Z is defined as

$$S_\alpha(A, B) = S_\alpha\left(\frac{A \circ B}{\mathrm{tr}(A \circ B)}\right), \tag{2.11}$$

where $A \circ B$ denotes the Hadamard product, i.e., the element-wise product of two matrices. From Equations (2.10) and (2.11), the mutual information can be defined as

$$I_\alpha(X; Z) = S_\alpha(A) + S_\alpha(B) - S_\alpha(A, B), \tag{2.12}$$

which is analogous to the standard mutual information in the new space called reproducing kernel Hilbert space (RKHS). Although $I_\alpha(X; Z)$ is mathematically different from $I(X; Z)$ in Equation (2.9), this quantity satisfies the mathematical requirements as Rényi's entropy [29]. Furthermore, it has a great computational merit in that its computation is not affected much by the dimension n of X, unlike the standard binning method for estimating the mutual information. In a simple setup where an exact computation of $I(X; Z)$ is possible, I confirmed that the matrix-based $I_\alpha(X; Z)$ gives an accurate estimation of $I(X; Z)$ (see Appendix A). Compared to the matrix-based estimator, the binning method gives less accurate results that are violently affected by the resolution of discretization and sample size. Using this estimator, Yu and Principe visualized the IP trajectories of AEs and suggested the optimal design of AEs based on IP patterns [18].

The kernel estimator contains a hyperparameter that defines a kernel function of distances between samples. As the estimator depends on the dimension and scale of variables for samples, the hyperparameter should be carefully determined [19]. Despite careful determination, the matrix-based kernel estimator seems unstable because it is sensitive to the training setup of neural networks. Moreover, once the information process of deep neural networks is quantified

by this estimator, it sometimes violates the DPI, which is a necessary condition for interpreting layer stacks as Markov chains. I found that the raw activities of neural networks can result in inaccurate entropy estimations irrespective of the estimation schemes when they have different dimensions and scales depending on layers. Large activities tend to overestimate their entropies, whereas low activities tend to underestimate their entropies. In particular, the use of a linear activation function or rectified linear unit (ReLU) often results in the violation of DPI (see Figures 6 and 9 in [19]). To address this issue, I unified the activation function of all hidden layers into the sigmoid function ($f(s) = 1/(1+\exp(-s))$), except for VAE and LAE, whose bottleneck layers used Gaussian sampling and a softmax function, respectively; and this setup eliminated the DPI violation.

Saxe et al. argued that using double-sided saturating activation functions such as $f(s) = \tanh(s)$ trivially induces the simplifying phase on the IP, and it is not related to the generalization of machine learning [15]. They showed that the mutual information, estimated by the binning method, first increases and then decreases as the weight parameters of neural networks get larger. The second decreasing phase of mutual information causes the simplifying phase. I performed the same task with various activation functions, including sigmoid and ReLU, but I estimated the mutual information using the aforementioned matrix-based kernel method. Then, I confirmed that the second decreasing phase did not occur by merely increasing the weight parameters, suggesting that the existence of the simplifying phase does not depend on the selection of activation functions in our matrix-based kernel method. Further details on this experiment are provided in section A. For those who are interested in using IP analysis, I have provided the complete source code and documentation on GitHub [32].

2.4 Information plane of autoencoders

In this section, I examine the MI of various AE models using the method introduced in the previous section, and visualize it on IP. Our main concern is whether the phase transition in IP can be observed in representation learning. Furthermore, by comparing the IPs of different AEs, I investigate how the various techniques I adopted for efficient training of neural networks modified the information flow in latent space.

2.4.1 Vanilla autoencoders

In this study, I investigated the information process of representation learning for real image datasets (Figure 2.1 (a)): MNIST [33], Fashion-MNIST [34], and EMNIST [35]. MNIST has 60,000 training and 10,000 testing images of 28×28 pixels of 10 hand-written digits (0–9). Fashion-MNIST has the same data size as MNIST for 10 different fashion products, such as dresses and shirts. Finally, EMNIST is an extension of MNIST; it contains 10 digits and 26 uppercase (A–Z) and lowercase letters (a–z). In this section, I focused on the results of MNIST because the results of Fashion-MNIST and EMNIST are basically the same (refer [32]).

For representation learning of MNIST, I first considered a shallow AE ($X - Z - X'$) that included a single hidden or bottleneck layer (Figure 2.1 (b)). The input, hidden, and output layers had $n_X = 28 \times 28 = 784, n_Z = 50$, and $n_{X'} = 784$ nodes, respectively. I considered a fully-connected network between layers with the loss functions listed in Table 2.1. For the optimization of network weights, I used the stochastic gradient descent method with Adam optimization, given a batch size of 100 for a total of 50 epochs. With each learning iteration, the $\mathrm{MSE}(X, X')$ kept decreasing (Figure 2.1 (c)). This im-

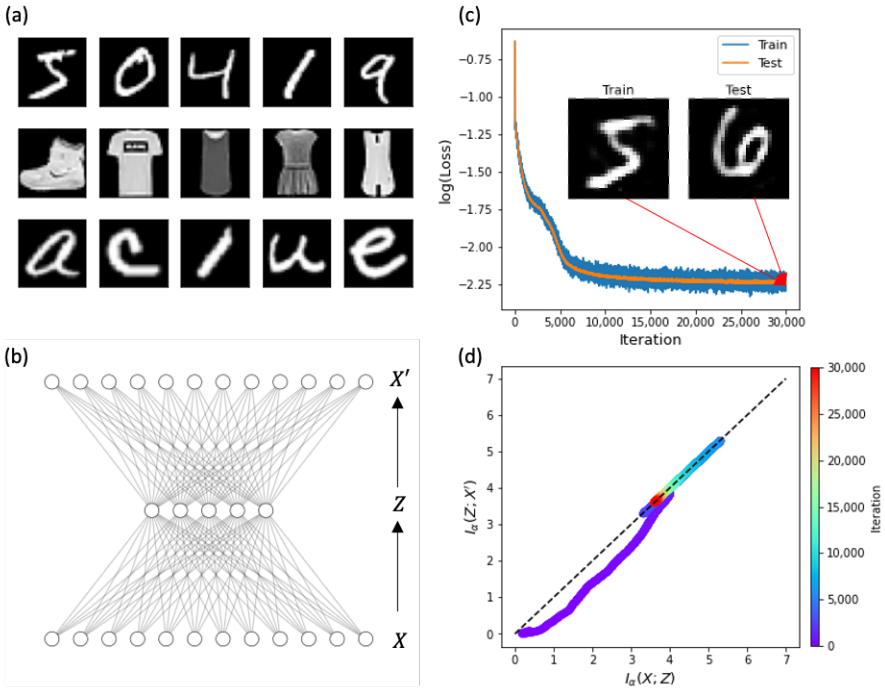

Figure 2.1 Information transmission and compression of autoencoders. (a) Image datasets X of MNIST (top row), Fashion-MNIST (middle), and EMNIST (bottom). (b) Network structure of a shallow autoencoder: input X, hidden Z, and output X'. Note that node numbers are arbitrary for a schematic display. (c) Error (or loss) between desired output X and reconstructed output X' for training (blue) and test (orange) data during learning iterations. Insets are snapshots of reconstructed training and test images of X' at the final iteration. (d) Trajectory of mutual information $(I_\alpha(X;Z), I_\alpha(Z;X'))$ on the information plane. The color bar represents the number of iterations.

plies that the output X' of AE successfully reproduced the input image X of training data. To measure the generalization ability of the AE, I examined the reproduction ability of the AE for test images that were not used in the learning process. I confirmed that the test error was as small as the training error. Given the faithful reproduction of input images, the indifferent error between training and test images defines successful generalization as usual.

The IP trajectory of the AE during the learning process is presented in Figure 2.1 (d). As expected, the trajectory satisfies the inequality of $I_\alpha(X;Z) \geq I_\alpha(Z;X')$, and ended up on their equality line because of $X' \approx X$ at the end of training. As observed by Shwartz and Ziv [14], the IP trajectory showed two distinct phases of fitting and simplifying. In the initial fitting phase, the input mutual information $I_\alpha(X;Z)$ between X and Z increased. Then, during the second simplifying phase, $I_\alpha(X;Z)$ decreased. Note that this representation learning showed a simultaneous decrease in the output mutual information, whereas general supervised learning maintained the output mutual information as constant during the simplifying phase.

Next, I considered a deep AE ($X - E_1 - E_2 - Z - D_1 - D_2 - X'$) with two additional encoder layers before the bottleneck layer and two decoder layers after the bottleneck layer (Figure 2.2 (a)). The corresponding node numbers for the inner layers were $n_{E_1} = 256$, $n_{E_2} = 128, n_Z = 50, n_{D_1} = 128$, and $n_{D_2} = 256$. The deep AE exhibited similar learning accuracy and generalization ability to the shallow AE (Figure 2.2 (b)). During the learning process, I measured the mutual information using the matrix-based kernel estimator and confirmed that the learning process of the deep AE satisfied the DPI (Figure 2.2 (c)). I observed the simplifying phase in the inner layers of E_2, Z, and D_1 (Figure 2.2 (d)). However, the simplifying phase was not evident in the outer layers of E_1

and D_2 that had relatively large dimensions with high information capacity.

Subsequently, I explored whether the simplifying phase appeared even with a small amount of training data. Unless sufficient training data are provided, machine learning can easily overfit a small amount of training data and fail to generalize the test data. I conducted a learning experiment with the deep AE using 10% of the total training data. The training error kept decreasing, similarly to the training error given the full training data. However, the test error was significantly larger than the training error (Figure 2.2 (e)). This demonstrates that the deep AE failed to generalize. After confirming the satisfaction of DPI (Figure 2.2 (f)), I examined the IP trajectories. Unlike the results of full training data, I did not observe the simplifying phase from any layers (Figure 2.2 (g)). Thus, this difference suggests that the simplifying phase seems to be associated with generalization by removing irrelevant details.

2.4.2 Sparse activity and constrained weights

To examine the effect of regularization on the information flows, I considered different species of AEs that can modify the learning phases. SAE and TAE have additional regularization phases for node activities and weight parameters, respectively, in comparison to vanilla AE (Table 2.1). First, I examined SAE, which has the same structure as a shallow AE. SAE showed perfect learning and generalization (Figure 2.3 (a)). It is of particular interest that the simplifying phase is markedly exaggerated in SAE (Figure 2.3 (b)). The sparsity penalty of SAE turns off unnecessary activities of hidden nodes, which can accelerate the simplifying phase.

Second, I examined TAE, which also has the same structure as shallow AE and SAE, but has a weight constraint of $W_E = W_D^T$. Similarly to shallow

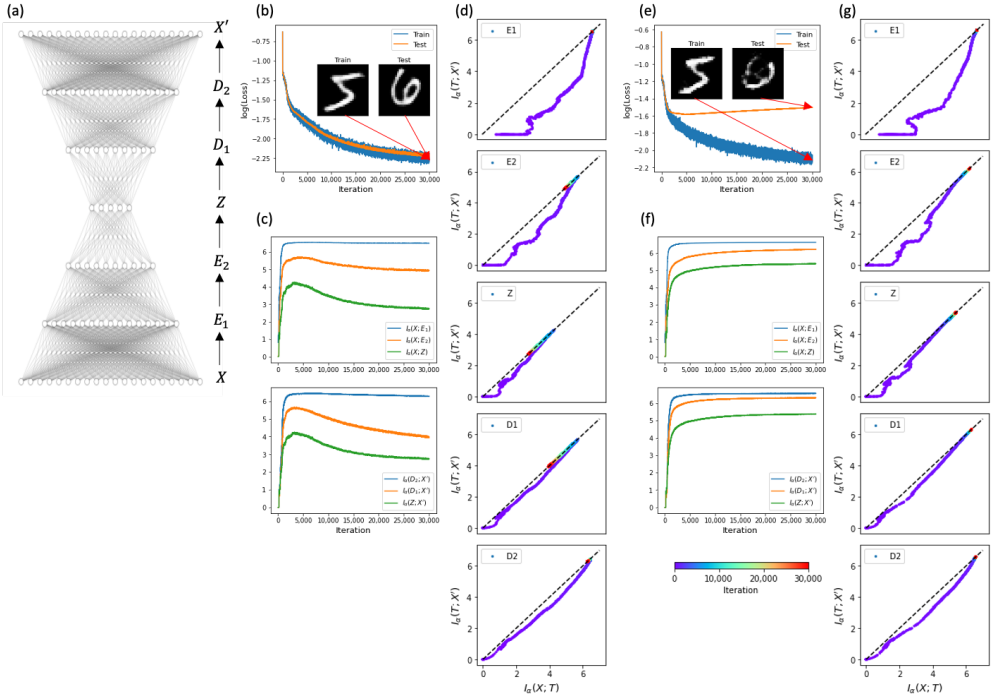

Figure 2.2 The simplifying phase and generalization of representation learning. (a) A deep autoencoder with input X; two encoders, E_1 and E_2; bottleneck Z; two decoders, D_1 and D_2; and output X'. (b) Learning errors for training (blue) and test (orange) data during iterations. Insets are snapshots of reconstructed training and test images of X' at the final iteration. (c) Changes in input mutual information (upper) and output mutual information (lower) during iterations. (d) Learning trajectories on the information plane. The general variable T stands for E_1, E_2, Z, D_1, or D_2. (b–d) Experiments with the full training set of 60,000 MNIST data. (e–g) Experiments with the 10% training set of 6,000 MNIST data.

AE and SAE, TAE showed perfect learning and generalization; however, its learning accuracy was slightly lower under the weight constraint (Figure 2.3 (c)). However, TAE did not exhibit the simplifying phase (Figure 2.3 (d)). This implies that the simplifying phase is not necessary for generalization. Given the weight constraint, TAE seems to have less potential capacity to remove irrelevant information than vanilla AE.

2.4.3 Constrained latent space

Now, I survey another species of AEs that more actively modify the latent space of the bottleneck layer, and further investigate the information flows in the learning process.

VAE is a generative model that maps input data X into a Gaussian distribution $q_\phi(Z|X)$ for the latent variable Z. I considered a deep VAE that had the same structure ($X - E_1 - E_2 - Z - D_1 - D_2 - X'$) as deep AE, and confirmed that the VAE can learn the training data of MNIST and generalize to reproduce the test data (Figure 2.4 (a)). However, because VAE had a special constraint for the bottleneck layer, the information process from the input layer into the bottleneck layer did not satisfy the DPI (Figure 2.4 (b)). The mutual information $I_\alpha(X;Z)$ between X and Z did not change during the training process. Indeed, the fixed value was close to the maximum entropy of X given its batch size of 100 samples, $I_\alpha(X;Z) \approx \log_2(100) \approx 6.6$, which was independent of dimension n_Z of Z (data not shown). It is noteworthy that the mutual information between X and Z did not change for the learning process, although the mapping $X \to Z$ kept reorganizing to distinguish the feature differences of X. This shows a limitation of the information measure $I_\alpha(X;Z)$, which failed to capture the content-dependent representation of Z. Besides the bottleneck

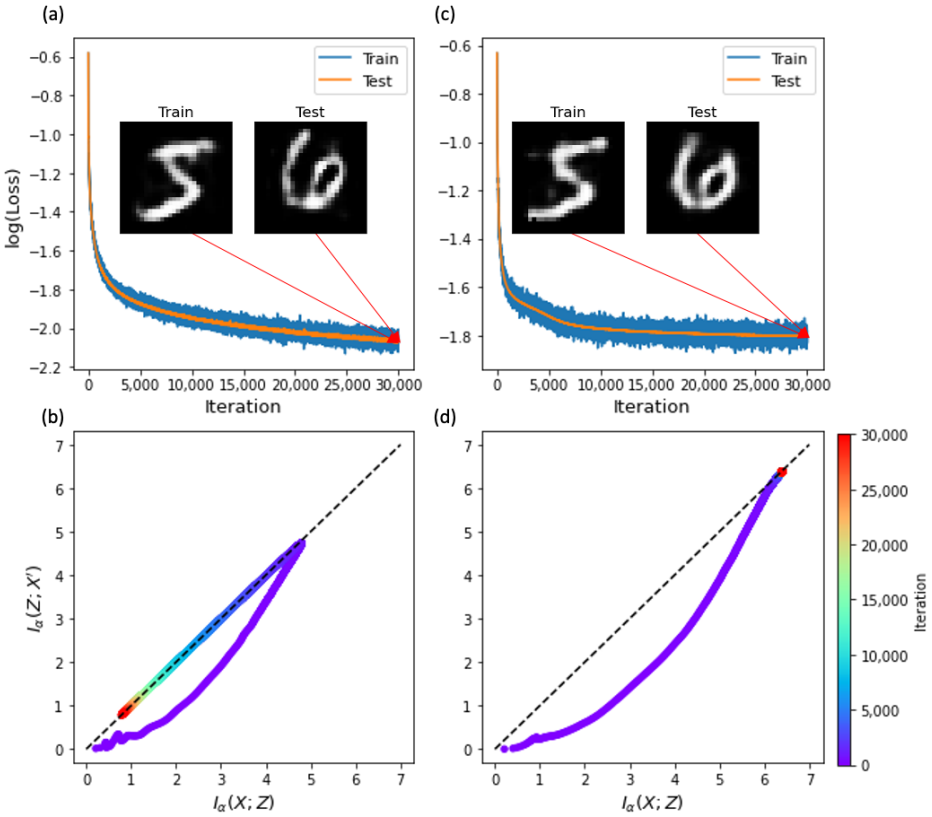

Figure 2.3 Information compression in constrained autoencoders. (a) Learning errors for training (blue) and test (orange) data during iterations. Insets are snapshots of reconstructed training and test images of X' at the final iteration. (b) Learning trajectories on the information plane. (a,b) Results of sparse autoencoders (SAE). (c,d) Results of tied autoencoders (TAE). The network structure of SAE and TAE can be represented by $X - Z - X'$.

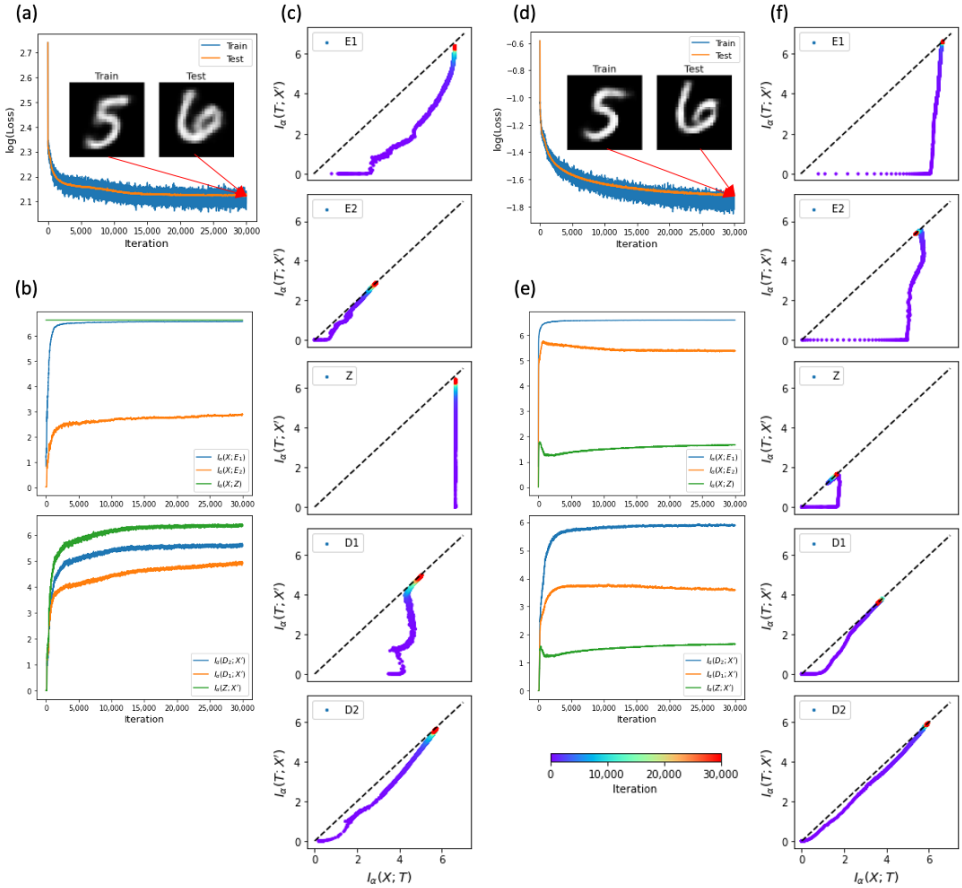

Figure 2.4 Information trajectories of generative models. (a) Learning errors for training (blue) and test (orange) data during iterations. Insets are snapshots of reconstructed training and test images of X' at the final iteration. (b) Changes in the input mutual information (upper) and output mutual information (lower) during iterations. (c) Learning trajectories on the information plane. (a–c) Results of a variational autoencoder (VAE). (d–f) Results of a label autoencoder (LAE). VAE and LAE had a deep network structure with $X - E_1 - E_2 - Z - D_1 - D_2 - X'$. The general variable T denotes E_1, E_2, Z, D_1, or D_2.

layer, other layers still satisfied the DPI. Next, I displayed the IP trajectories of VAE for each layer (Figure 2.4 (c)). I did not observe the simplifying phase in any layer in the VAE. Therefore, VAE can generalize without the simplifying phase, similarly to TAE.

LAE is another generative model that maps X into Z, where Z corresponds to label Y of X. Thus, unlike other AE models, LAE uses label information to shape its latent space. I used the same deep network structure as the deep AE and VAE for LAE. The deep LAE could also learn the training data of MNIST and generalize to reproduce the test data as well (Figure 2.4 (d)). LAE satisfied the DPI (Figure 2.4 (e)), and its IP trajectories also satisfied the inequality of $I_\alpha(X;T) \geq I_\alpha(T;X')$ (Figure 2.4 (f)). It is interesting that LAE has orthogonal learning phases. LAE first increased the input mutual information $I_\alpha(X;T)$ for the encoding part. Once LAE arrived at a certain maximal $I_\alpha(X;T)$, the output mutual information $I_\alpha(T;X')$ started to increase. This shows that LAE first extracts information from the input data relevant for the label classification, and then transfers information to output for reproducing input images. I found that the LAE did not exhibit the simplifying phase, even though it successfully generalized.

2.5 Conclusion

I studied the information flows in the internal representations of AEs using a matrix-based kernel estimator. AEs are perfect models to investigate how the information flows are shaped during the learning process depending on the network designs, since they have diverse species with various depths, sparsities, weight constraints, and hidden representations. When I used sufficient training data, shallow and deep AEs demonstrated the simplifying phase, following

the fitting phase, along with the generalization ability to reproduce test data, thereby confirming the original proposal by Shwartz-Ziv and Tishby [14]. However, when I used a small amount of training data to induce overfitting, the AEs did not exhibit a simplifying phase and generalization, suggesting that the simplifying phase is associated with generalization. When a sparsity constraint was imposed in the hidden activities of SAE, regularization amplified the simplifying phase and provided more efficient representations for generalization. However, the constraining weight parameters ($W_E = W_D^T$) of TAE showed perfect generalization in the absence of the simplifying phase. Furthermore, VAE and LAE, shaping the latent space with a variational distribution and label information, also achieved generalization without the simplifying phase. These counterexamples of TAE, VAE, and LAE clearly demonstrate that the simplifying phase is not necessary for the generalization of models.

It is noteworthy that the absence of the simplifying phase does not mean that compression does not occur in representation learning. When the encoder part has a narrowing architecture, information compression is inevitable, as demonstrated by the DPI. Then, the removal of irrelevant information from data may contribute to the generalization of models. After the completion of representation learning, AEs obtain a certain amount of mutual information $I_\alpha(X;Z) = I_{final}$ between the input data X and its internal representation Z. The paths that obtain I_{final} seem different between AEs. In TAE, VAE, and LAE, $I_\alpha(X;Z)$ monotonically increases to I_{final}. However, in vanilla AE and SAE, $I_\alpha(X;Z)$ first increases beyond I_{final}, and then decreases back to I_{final}. The backward process is called the simplifying phase. As the loss function of representation learning never includes any instruction for the path of $I_\alpha(X;Z)$, it is not surprising that the existence of the simplifying phase is not universal. In summary, in the basic structure of AE, I found that the simplifying phase is

related to the generalization of the model, and confirmed that learning dynamics of neural network can be interpreted with IB theory. However, for several variants of AE, no simplifying phase was observed, suggesting that all types of deep learning do not follow a universal learning dynamics.

Although observations and physical meanings of the phase transition in IP were contradictory in previous studies, it is still manifest that IP analysis is an excellent tool to monitor information transmission and compression inside the "black box" of neural networks. For IP analysis, accurate information estimation is a prerequisite. In general, it is difficult to calculate the entropies of high-dimensional variables, but I could solve this problem by estimating the physical quantities corresponding to the entropies in a kernel space. When I applied the estimator to the representation learning, I found that it is critical to use bounded activation functions. When I used ReLU as an activation function, the DPI was easily violated, although I observed the simplifying phase in this setting. Thus, it can be problematic to estimate the mutual information from unbounded variables with different scales for different layers. In this study, I provided concrete grounds to further explore the theoretical understanding of information processing in deep learning.

Chapter 3

Scale-invariant representation of machine learning

3.1 Internal representation of machine learning

The marvellous performance of machine learning [36–38] is due to the internal representation, denoted by z, of neural networks that extracts features from data. Here, z works as effective representations to discriminate images, speech, time series for pattern recognition [39], classical and quantum phases in matters and active matters [40–43], chemical structures for drug discovery [44, 45], time arrows of non-equilibrium dynamics [46], etc. Therefore, understanding the mechanism behind the effective feature distillation is a fundamental problem in machine learning.

The information bottleneck theory interprets machine learning as information compression and transmission of the communication theory [47]. Neural networks have internal representations z that maximally compress irrelevant

information in input data x for restoring desired outputs y, called labels. Given data without labels, autoencoders use input itself as output $y = x$ [48]. Then, the self-supervised learning can work for the dimensional reduction of data by providing compressed representation z that can faithfully reproduce x. In particular, when the transformation of $x \to z$ is linear mapping, the machine corresponds to the principal component analysis [49, 50]. In addition to the self-supervised learning, unsupervised learning such as restricted Boltzmann machine (RBM) and deep belief networks is also used for dimensional reduction of x [51]. For the unsupervised learning, the internal representation z can be interpreted as emergent labels for each x.

The representation z sometimes reflects feature itself like edge detection in image recognition [52]. On the other hand, z can be considered as a dummy code where its frequency only matters when no prior knowledge on x is provided. A recent interesting observation is that the frequency of z follows power laws in RBMs [53, 54], reminiscent of criticality in statistical mechanics [55]. Using the dummy code z as emergent labels of x, one can interpret the frequency of z as a cluster size of x labeled by z. Then, the power laws imply that the cluster size distribution is scale-invariant. It is of great intellectual interest to address how the power laws arise during the learning process devoid of any specific instruction for the scale-invariance. Song *et al.* have derived that the power-law clustering is an entropy-maximized distribution at a certain compression level of z in RBMs [53]. In this study, I will extend this idea using information theory, and show that the power-law scaling of the cluster size distribution emerges naturally not only in unsupervised learning, but also in supervised learning.

This chapter is organized as follows. In Sec. 3.2, I show that various machine learning models have scale-invariant distributions for their internal representa-

tion. I then explain the emergence of the scale invariance using information theory in Sec. 3.3. Finally, I summarize and discuss our findings in Sec. 3.4.

3.2 Power laws in machine learning

I first reproduce that the frequency of z follows power laws in RBMs (Fig. 3.1 (a)). Next, I observe that the scale-invariant internal representation is also found in supervised learning (Fig. 3.1 (b,c)). Finally, I examine how the emergence of power laws can be dependent on learning process and data preparation. For those who are interested in reproducing our results, I have provided the complete source code and documentation on GitHub [56].

3.2.1 Unsupervised learning

The goal of unsupervised learning is to extract inherent probability distribution $p(x)$ of data x, unlike the supervised learning that extracts the paired information between x and label y. RBM is a representative neural network for unsupervised learning that is composed by input layer for x and hidden layer for z [57, 58]. The RBM has a special graph structure in which input nodes are not connected to other input nodes, and the same is for hidden nodes (Fig. 3.1 (a)). This allows factorization of conditional probability $p(z|x) = \prod_i p(z_i|x)$ for $z = (z_1, z_2, \cdots, z_m)$ and $p(x|z) = \prod_j p(x_j|z)$ for $x = (x_1, x_2, \cdots, x_n)$. The goal of RBM, matching the model distribution $p(x)$ into the data distribution $\hat{p}(x)$, is achieved by the contrastive divergence algorithm, a kind of sampling method using the forward probability $p(z|x)$ and backward probability $p(x|z)$ [59].

For the unsupervised learning, I used the MNIST dataset [33], which consists of 60,000 training and 10,000 testing images of 28×28 pixels of 10 different handwritten digits (0-9). After the RBM successfully generates original digit images,

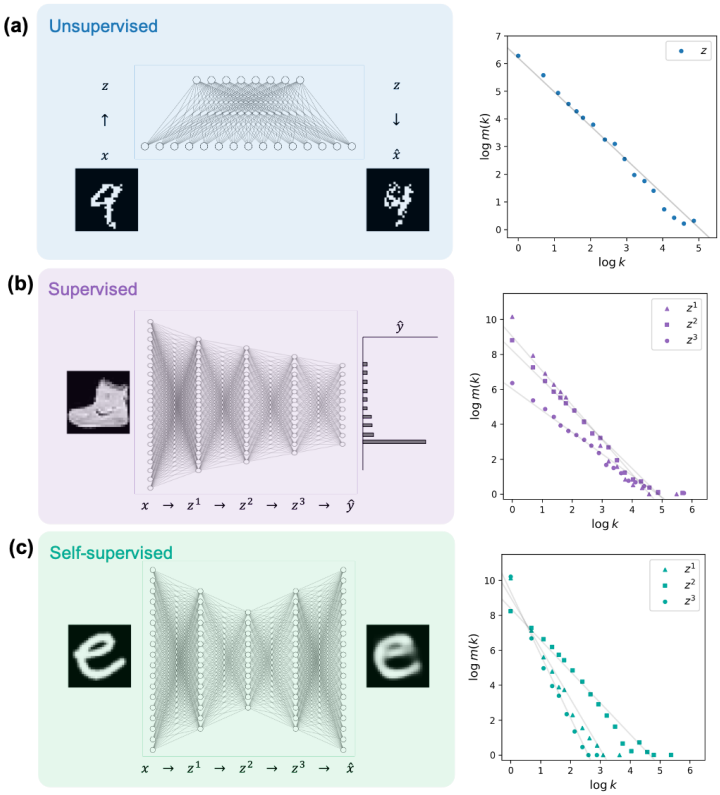

Figure 3.1 (Color online) Scale-invariant internal representations of machine learning. (a) Unsupervised learning of restricted Boltzmann machine (RBM) with MNIST dataset. The network of RBM consists of visible and hidden units of x and z. An image x of MNIST is reconstructed to $\hat{x}$ through the hidden representation z. (b) Supervised learning with image dataset of Fashion-MNIST. Architecture of a deep neural network consists of input x, hidden $z^\mu (\mu = 1, 2, 3)$, and output $\hat{y}$. Note that the number of nodes is arbitrary for a schematic visualization. (c) Self-supervised learning with EMNIST dataset. It has a symmetric structure with the bottleneck layer z^2. Right panels are corresponding log-log plots between degeneracy $m(k)$ and frequency k of internal representations: z (circles) for RBM; and z^1 (triangles), z^2 (squares), and z^3 (circles) for supervised and self-supervised learning.

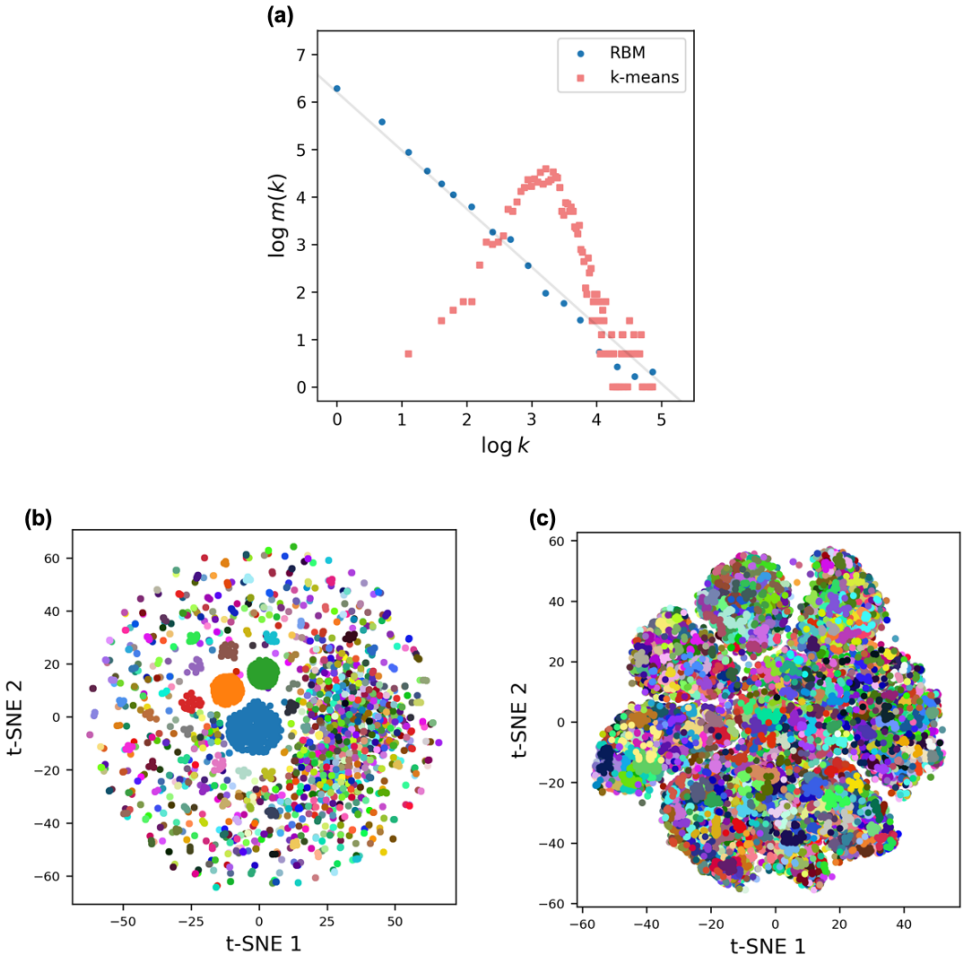

Figure 3.2 (Color online) Power-law data clustering. (a) The size distribution of data clusters via internal representation z of the restricted Boltzmann machine (RBM, blue circles) and k-means clustering (coral squares). (b) Two-dimensional visualization of z using t-distributed Stochastic Neighbor Embedding (t-SNE) with different colors for different z. (c) t-SNE plot of x with different colors for different k-means clusters. I set k= 2048 to be comparable with the total number (≈ 2000) of distinct states in the RBM.

I count frequencies k_z of specific z allowing the mapping of different images of x into the same z. I observed that the neural networks have a few frequent z and many rare z, which is described by the degeneracy of the frequency k, $m(k) = \sum_z \delta\big(k - k_z\big)$. It is of particular interest that the frequency degeneracy follows power laws of $m(k) \sim k^{-\beta-1}$ (Fig. 3.1 (a)), as reported in Song *et al.* [53]. The scale-invariant cluster size distribution is non-trivial given that the representative clustering method of k-means, based on the Euclidean distance between data, produced uni-polar distribution with a characteristic cluster size (Fig. 3.2 (a)). The different cluster size distributions can be further visualized using two dimensional projection through t-distributed Stochastic Neighbor Embedding (t-SNE) method (Fig. 3.2 (b,c)). The scale-invariant distribution of RBMs may have functional advantage to classify the data into a large cluster of frequent typical data, and many small clusters of atypical data as outliers.

3.2.2 Authentic supervised learning

The goal of supervised learning is to predict true labels y from input data x. In the communication theory ($y \to x \to z \to \hat{y}$), a message y is transferred to a noisy code x, which is then mapped into a compressed code z. Finally, I decode z to have $\hat{y}$. The transmission succeeds if the decoded message is consistent with the true message ($\hat{y} = y$).

As a concrete example, I consider Fashion-MNIST dataset [34] in which labels y are ten fashion products such as sneakers and shirts. The labels are assigned to 70,000 (60,000 training and 10,000 test) 28×28 pixel images of x. Using a deep neural network, I transformed $x \to z^1 \to z^2 \to z^3 \to \hat{y}$ while reducing the dimension of the corresponding layers from 784 to $70, 50, 35$, and 10 (Fig. 3.1 (b)). Here, true labels y are expressed as 10-dimensional one-hot

vectors of which components have values between 0 and 1. The network is trained to reduce the discrepancy between y and $\hat{y}$.

Once the classification accuracy for the test data reaches 87%, I examined frequencies of internal representations of z^1, z^2, and z^3. Since they have continuous values in the multi-layer perceptrons, I binarized them to count finite coarse-grained representations. I count frequencies k_z of discretized z allowing the mapping of different images of x into the same z. It is of particular interest that the frequency degeneracy follows power laws of $m(k) \sim k^{-\beta-1}$, where the exponent β depends on the dimension of the μ-th hidden layers z^μ (Fig. 3.1 (b)). I confirmed that the existence of power laws is insensitive to the binarization threshold, and the power-law exponents do not depend on the initialization of learning. It is noteworthy that the scale invariance has never been instructed by the learning algorithm.

3.2.3 Self-supervised learning

In real-world dataset, labels are not always available. For such dataset, if x played the role of label $y = x$, those machines for self-supervised learning are called *autoencoders*, which are useful for dimensional reduction [48], denoising [60], and generation [61, 62]. In particular, the structure of $x \to z \to \hat{x}$ implies that the compressed representation z is used to reconstruct original x (Fig. 3.1 (c)). This study focuses on the compressing conditions where the dimension of z is smaller than the dimension of x. Unlike the narrowing networks, widening networks map different x into different z. This makes the frequency of representation z trivial with $k_z = 1$.

The autoencoders are tightly related with RBMs. The unfolded structure of the forward and backward process of $x \leftrightarrow z$ in RBMs can be interpreted as

information flows, $x \to z \to \hat{x}$, in autoencoders where the weight parameter of the encoder part is transpose of the weight parameter of the decoder part [27]. However, it is noteworthy that z is a stochastic and discrete variable in RBMs, whereas $z(x)$ is a deterministic and continuous function of x in autoencoders. Nevertheless, if one adopts the sigmoidal function as the activation function for autoencoders, $z(x)$ can be interpreted as the expectation value $\mathbb{E}[z]$ of RBMs.

I confirmed that binarized z of the autoencoders also showed the power-law scalings when I carried out all-to-all connnected multilayer perceptron learning on the above Fashion-MNIST data (data not shown). Instead of repeating with the same dataset, here I considered another dataset, EMNIST, consisting of hand-written images of 10 digits (0-9) and 26 uppercase (A-Z) and lowercase letters (a-z) [35]. Among them, I trained 30,000 lowercase letters with deep structure of autoencoders $x \to z^1 \to z^2 \to z^3 \to \hat{x}$. Once the reconstructed image $\hat{x}$ faithfully reproduced the original image x, I counted the frequency of z^μ of each hidden layer. The frequency degeneracy $m(k)$ again follows power laws in every hidden layer with different exponents (Fig. 3.1 (c)).

To examine the robustness of our finding, I change the activiation function from the sigmoid function to Rectified Linear Unit (ReLU). Then, I confirmed that power laws still arise with the ReLU activation (data not shown). Next, I examine a convolutional neural network (CNN) that incorporates the information of proximal sites of x, which is known to show excellent performance for image recognition [63]. I considered CIFAR-10, consisting of 50,000 training and 10,000 testing 32×32 color images in 10 classes such as airplane and automobile [64]. I then adopted convolutional autoencoders. Once the reconstructed image $\hat{x}$ faithfully reproduce the original image x, I counted the frequency of z, and confirmed that the frequency degeneracy $m(k)$ again follows power laws

(Fig. 3.3).

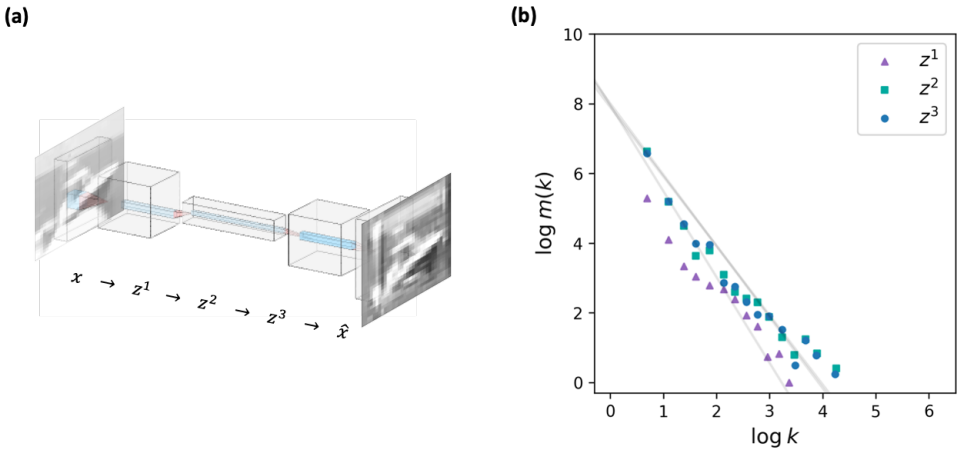

Figure 3.3 (Color online) Internal representations of convolutional neural network (CNN). (a) Architecture of a CNN with three internal layers of z^1, z^2, and z^3. Image data of CIFAR10 is reconstructed through the self-supervised learning of the CNN. (b) The frequency distributions of internal representations: z^1 (purple triangles), z^2 (green squares), and z^3 (blue circles).

3.2.4 Power laws with data distribution

It is questionable whether the power laws originate from learning processes, or merely from data distributions.

Suppose that neural networks start with random initial parameters before learning. Then, input data x is transformed to random internal representations. Depending on parameter initialization schemes, shallow internal representation z^1 sometimes displays a power-law scaling (Fig. 3.4). However, deep internal representation of z^μ for $\mu > 1$ experiences repeated transformation with random weights, which averages out the signal transfers, and converges into a few trivial representation. Therefore, the robust emergence of power laws clearly requires

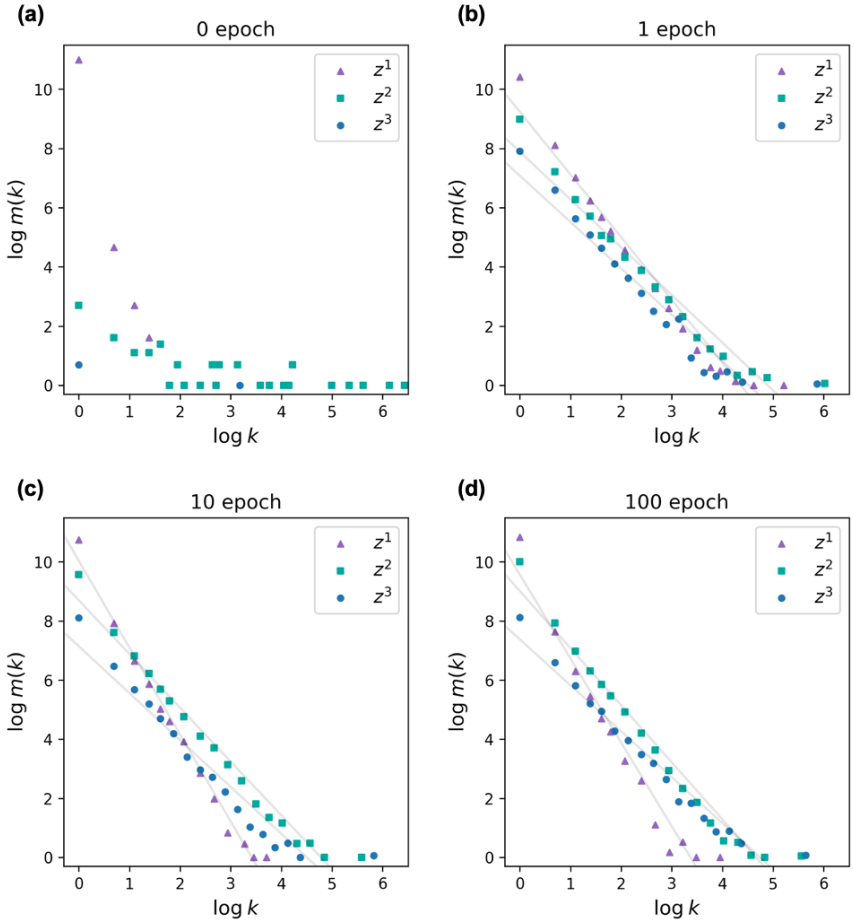

Figure 3.4 (Color online) Data clustering during learning process. Supervised learning of MNIST data through three internal representations of z^1, z^2, and z^3. The frequency distributions of internal representations: z^1 (purple triangles), z^2 (green squares), and z^3 (blue circles) at (a) 0, (b) 1, (c) 10, and (d) 100 epochs.

learning process. This excludes the possibility that the power laws may result from an artifact of random binarization of internal representations. However, it is surprising that one epoch, that experiences every training sample once, is good enough to start to show power-law scaling.

Next, the power laws should depend on original data distributions. For example, when data include identical samples, the distribution of corresponding internal representations is trivially influenced by the frequency of the identical samples. Although the image data in our study do not have identical samples, one may speculate that they have complex structures which generate the power laws without going through learning processes. To check this possibility, I examined structureless patterns x generated from two-dimensional Ising models. The Ising model has equilibrium patterns of x depending on its energy,

$$E(x) = -J \sum_{\langle i,j \rangle} x_i x_j, \tag{3.1}$$

where $\langle i, j \rangle$ represents the nearest-neighboring pairs and I fixed $J = 1$. Then, the realization probability of a pattern x follows the Boltzmann distribution:

$$p(x) = \frac{\exp[-E(x)/T]}{Z}, \qquad Z = \sum_x \exp[-E(x)/T]. \tag{3.2}$$

Depending on temperature T, one can generate diverse patterns. Low temperature produces simple patterns including a few defects, whereas high temperature produces random patterns mixing black ($x_i = 1$) and white ($x_i = -1$) pixels. At the critical temperature ($T \approx 2.26$) in the two-dimensional Ising model, complex patterns arise with long-range correlations between pixels. I prepared 50,000 equilibrium samples at each temperature. Since low temperature produces simple patterns, many identical samples of x are inevitably included in the low-temperature samples. I put the diverse Ising patterns of x as inputs for

a shallow autoencoder ($x \to z \to \hat{x}$) including one hidden layer. The autoencoder was trained to reconstruct pattern $\hat{x}$ which is identical with the input x. Given the input x and transformed z, I obtain their frequencies k_x and k_z, and then their degeneracy distributions of $m(k_x)$ and $m(k_z)$ for x and z (Fig. 3.5). Note that I introduced new notations of $m(k_x)$ and $m(k_z)$ to distinguish the degeneracy $m(k)$ for k_x and k_z. As expected, the peculiar input distribution of k_x due to identical samples at low temperature is trivially reflected in the distribution of k_z. However, if sufficiently diverse patterns, generated above critical temperature, are used for learning, their internal representations follow power laws. This result lends support to our observation of power laws not only for natural images, but also for synthetic structureless images, unless input data distributions have peculiar shapes with many identical samples.

3.3 Theoretical analysis of power laws

An intriguing question is how the power laws arise frequently in the various machine learning for diverse data. None of the learning algorithms have instructed the special shaping of z. Our claim is that the power laws correspond to entropy-maximized distributions among possible ones that satisfy pre-specified learning accuracies (Fig. 3.6). To progressively investigate this idea, I first review the information theoretic concepts of *resolution* and *relevance* [65], and the derivation of the scale-invariant hidden representations of RBMs [53]. Then, I extend this reasoning to explain the scale-invariant hidden representations for self- and authentic supervised learning, which is the major finding of this study.

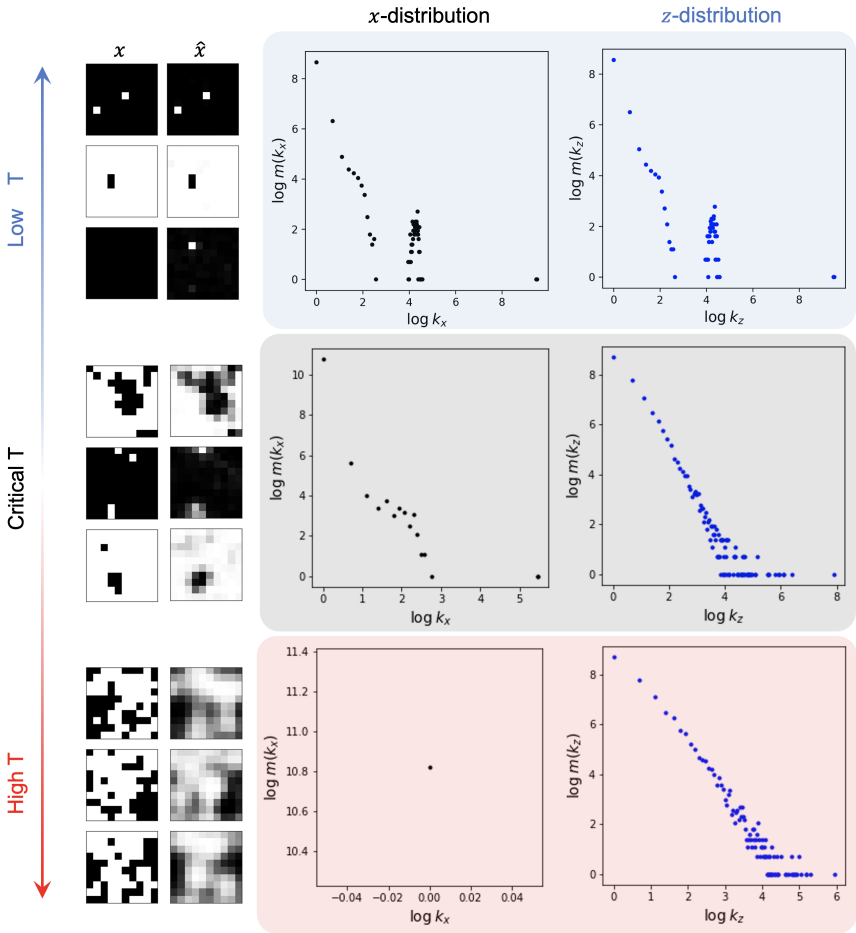

Figure 3.5 Self-supervised learning of Ising patterns. 10×10 lattice Ising patterns are used for input x for a shallow autoencoder ($x \to z \to \hat{x}$). Output $\hat{x}$ corresponds to reconstructed patterns. Two-dimensional Ising model was used to generate 50,000 equilibrium samples at three temperatures (low $T = 1.53$, critical $T = 2.26$, high $T = 3.28$). Input x and internal representation z are characterized with different degeneracy distributions $m(k_x)$ and $m(k_z)$. For the binarization of z, I used a threshold 0.4.

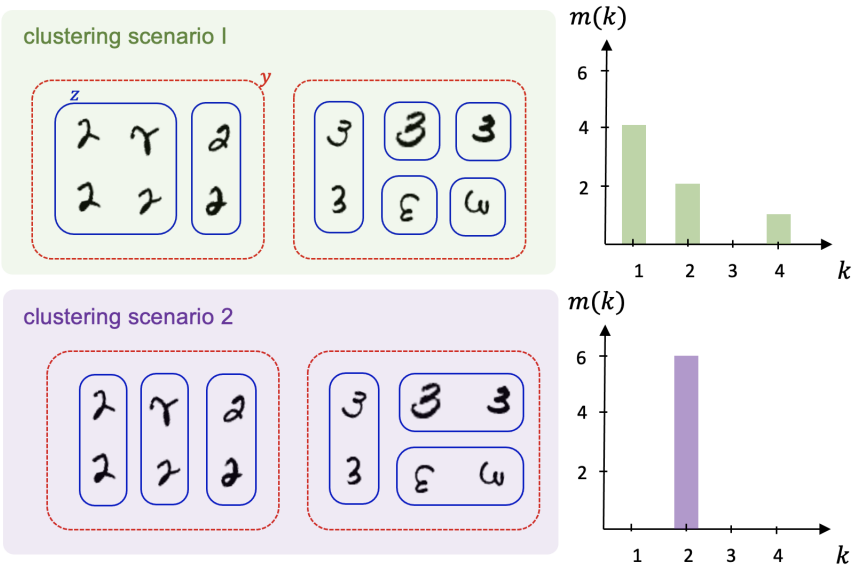

Figure 3.6 (Color online) Cluster size distribution of machine learning. In supervised learning, input image x is represented by its compressed code z (blue solid line), and then finally grouped into output y (red dotted line). Cluster size distribution $m(k)$ of cluster size k for two clustering scenarios. Two clustering scenarios give the same learning accuracy. Then, which one is more likely to happen?

3.3.1 Resolution and relevance

Let us consider a random variable z, of which frequency is k_z with a total number of realizations, $M = \sum_z k_z$. The uncertainty of z can be quantified using the Shannon entropy,

$$H(Z) = -\sum_z \frac{k_z}{M} \log \frac{k_z}{M}. \tag{3.3}$$

Since $H(Z)$ quantifies the effective number of distinct realizations of z, it is named *resolution* [65]. In terms of coding theory, $H(Z)$ corresponds to a minimum description length for z [55]. Different realizations z may take the same frequency $k_z = k$. Unless I have any prior knowledge on z, the frequency k_z may be the only extractable feature for z at the moment. Then, it is natural to consider the degeneracy $m(k)$ of the frequency k. Given the degeneracy $m(k)$, I can reformulate Eq. (3.3) in terms of k-summation instead of z-summation,

$$H(Z) = -\sum_k \frac{km(k)}{M} \log \frac{k}{M} \tag{3.4}$$

with $M = \sum_k km(k)$.

Now let us quantify the uncertainty of k that a certain z has a frequency $k_z = k$,

$$H(K) = -\sum_k \frac{km(k)}{M} \log \frac{km(k)}{M}. \tag{3.5}$$

If one does not distinguish states z that have the same frequency $k_z = k$, the variability of frequency k can measure the amount of relevant information in data. Thus $H(K)$ is named *relevance* [65]. If every z is distinct with the same frequency $k_z = 1$ and $m(1) = M$, I have no relevance $H(K) = 0$, although I have a maximal resolution $H(Z) = \log M$. On the other hand, if every z is identical with $k_z = M$ and $m(M) = 1$, I have also no relevance $H(K) = 0$, but with zero resolution $H(Z) = 0$. Therefore, these two extreme cases correspond to no

frequency variability where I cannot distinguish z in terms of their frequency k_z.

3.3.2 Unsupervised learning

The information measures of resolution and relevance have been adopted to explain the scale-invariant hidden representations of RBMs [53,55]. The graphical model of RBMs consists of visible and hidden units of x and z. RBMs are trained to have a large model probability $p(x, z)$ with a good pairing of data x and hidden representation z [51]. I can interpret z as emergent labels for x. This allows to define a group of x, which have the same label z, as a cluster. In particular, under compressing conditions when the dimension of z is smaller than the dimension of x, similar x is grouped together with label z. Then, k_z denotes the size of the z-labeled cluster, and $m(k)$ corresponds to the distribution of cluster sizes.

What is an expected distribution of the cluster sizes? RBMs do not impose any constraint on the shaping of the size distribution during the learning process. Song *et al.* found that the size distribution $m(k)$ follows power laws, and derived that the power laws correspond to the most likely distribution at a fixed resolution of z [53]. The scale-invariant distribution maximizes the uncertainty of the size k of a cluster to which a specific x belongs. This constrained optimization can be formulated using Lagrange multipliers:

$$\mathcal{L} = H(K) + \beta\Big(H(Z) - R\Big) + \alpha\Big(\sum_k \frac{km(k)}{M} - 1\Big). \tag{3.6}$$

The Lagrange multiplier α controls the normalization for k-distribution, while β controls the resolution for $H(Z) = R$. The maximum frequency variability $H(K)$ subject to the two constraints is obtained at the variation condition of

$\delta\mathcal{L}/\delta m(k) = 0$. The optimal condition leads to the power-law size distribution,

$$m(k) \propto k^{-\beta-1}. \tag{3.7}$$

Moreover, the scale-invariant hidden representation z achieves the goal for unsupervised learning to make the model probability $p(x) = \sum_z p(x, z)$ close to the data probability $\hat{p}(x)$.

3.3.3 Supervised learning

The goal of supervised learning is faithful reproduction of true label y given input x. For multi-layer neural networks, x is transformed to hidden representations z, and then, z is again transformed to output $\hat{y}$. The supervised learning optimizes an appropriate representation z to produce $\hat{y}$, which is ultimately used to predict true y. The learning accuracy can be estimated by the mutual information between internal representation z and true label y,

$$I(Z;Y) = H(Y) - H(Y|Z), \tag{3.8}$$

that quantifies how much uncertainty $H(Y)$ of y is reduced by knowing z. The information gain corresponds to learning accuracy of the internal representation z. Here, I focus on the neural networks of which internal layers have smaller sizes as they get farther from the input layer. The compressing condition provides coarse-grained representation z for x. In particular, if I discretize z, I can again interpret a group of x, that have the same hidden state z, as a cluster. Now, I derive the most likely distribution of hidden representation z that guarantees a certain learning accuracy as $I(Z;Y) = R'$. Like the objective for unsupervised learning in Eq. (3.6), the objective for supervised learning can be formulated as

$$\mathcal{L}' = H(K) + \beta\Big(I(Z;Y) - R'\Big) + \alpha\Big(\sum_k \frac{km(k)}{M} - 1\Big). \tag{3.9}$$

Among possible representations z satisfying pre-specified learning accuracy, I find a representation z that maximizes the uncertainty of the cluster size. It requires caution to understand that this exploration is different from the optimization of learning algorithms. I seek for the most flexible representation z, that gives the largest frequency variability $H(K)$, subject to a fixed learning accuracy. Thus, I explore snapshots of z at any learning status. Figure 3.4 showed power-law distributions of $m(k)$ at any given learning accuracy during learning epochs.

3.3.4 Self-supervised learning

I first consider the self-supervised learning of autoencoders in which labels are input itself as $y = x$. The mutual information of autoencoders is simply $I(Z;X) = H(Z) - H(Z|X) = H(Z)$, where the conditional entropy $H(Z|X) = 0$ vanishes because the hidden representation is a deterministic function $z(x)$ of x in autoencoders. This condition makes Eq. (3.9) identical to Eq. (3.6) with $\mathcal{L}' = \mathcal{L}$ and $R' = R$. I already know that the optimal size distribution follows power laws in Eq. (3.7) that maximizes $\mathcal{L}$. This explains why power laws naturally arise in the self-supervised learning.

As mentioned earlier, the unfolded structure of RBMs can be interpreted as constrained autoencoders [27]. The forward and backward propagation of $x \leftrightarrow z$ corresponds to the information flow of $x \to z \to \hat{x}$. The autoencoders have a constraint that the encoder weight parameter for $x \to z$ is the same with the transpose of the decoder weight parameter of $z \to \hat{x}$. Then, the scale-invariant hidden representation of RBMs can be understood either as (i) maximizing $H(K)$ subject to a certain resolution $H(Z) = R$, or (ii) maximizing $H(K)$ subject to a certain learning accuracy $I(Z;X) = R'$.

3.3.5 Authentic supervised learning

Next, I show that the previous conclusion of the self-supervised learning is also applicable for authentic supervised learning. The mutual information can be decomposed as:

$$I(Z;Y) = H(Y) + H(Z) - H(Y,Z). \tag{3.10}$$

The first term $H(Y)$ is constant, since labels y are given as data in supervised learning. As a result, $H(Y)$ is trivially independent on z and $m(k)$. The second term $H(Z)$ depends on $m(k)$ as shown in Eq. (3.4). The third term is the entropy for the joint frequency $k_{y,z}$,

$$H(Y,Z) = -\sum_{y,z} \frac{k_{y,z}}{M} \log \frac{k_{y,z}}{M}. \tag{3.11}$$

Now, I derive that $H(Y,Z)$ does not explicitly depend on $m(k)$. At first sight, the independence of $H(Y,Z)$ on $m(k)$ looks non-trivial, since both $H(Y,Z)$ and $m(k)$ depend on z. For the self-supervised learning ($Y = X$), I have a constant $H(X,Z) = \log M$, if every x is distinguished ($k_{x,z} = 1$). It is clear that $H(X,Z)$ is always constant independently with $m(k)$.

For the authentic supervised learning, let us imagine a causal graph between variables (Fig. 3.7). The number of realizations of hidden representation z is the frequency k_z, of which degeneracy is $m(k)$. The number of realizations for y and z is $k_{y,z}$, which is used to compute $H(Y,Z)$. Therefore, z is a confounder that affects both $m(k)$ and $H(Y,Z)$ in the causal graph. In terms of causality [66], $H(Y,Z)$ and $m(k)$ are called d-separate for fixed z:

$$H(Y,Z) \perp\!\!\!\perp m(k) \mid z, \tag{3.12}$$

which means that $m(k)$ does not explicitly affects $H(Y,Z)$ and vice versa. The functional independence can be further demonstrated in two situations. The

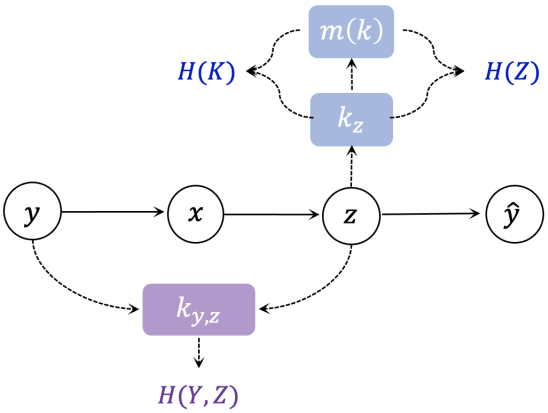

Figure 3.7 Causal graph for supervised learning. Solid arrows represent explicit transformation from parent to child nodes, whereas dotted arrows represent implicit transformation from a set of parent variables to child nodes.

first situation is that $m(k)$ changes, but $H(Y, Z)$ does not change (Fig. 3.8 (a)). The second situation is that $H(Y, Z)$ changes, but $m(k)$ does not change (Fig. 3.8 (b)).

So far, I confirmed that $H(Y)$ and $H(Y, Z)$ in the learning accuracy of Eq. (3.10) do not explicitly depend on $m(k)$. This results in $\delta\mathcal{L}'/\delta m(k) = \delta\mathcal{L}/\delta m(k)$. This explains why the authentic supervised learning has the power-law distributions of $m(k)$ like the self-supervised learning.

It is noteworthy to mention one exceptional situation where every cluster has data x with pure labels y (Fig. 3.8 (c)). This situation is difficult to practically happen in machine learning with large noisy data. The ideal clustering with no impurity leads to $H(Y, Z) = H(Z) + H(Y|Z) = H(Z)$, because labels do not have additional information beyond the hidden representation z. Plugging this result into Eq. (3.10), I have $I(Z; Y) = H(Y)$. This makes $H(K)$ the only $m(k)$-dependent term of $\mathcal{L}'$ in Eq. (3.9). Then, $m(k) \propto k^{-1}$ maximizes $\mathcal{L}'$. This

solution corresponds to the power-law distribution with $\beta = 0$. The zero value of Lagrange multiplier nullifies the constraint for the learning accuracy in $\mathcal{L}'$.

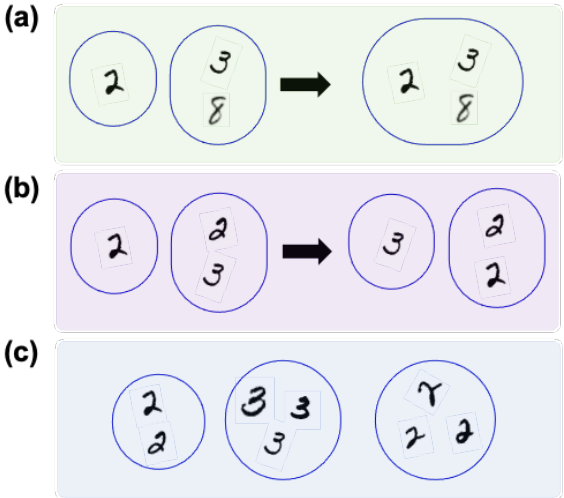

Figure 3.8 Labels and internal representations of data. MNIST digit images x have their true labels (2, 3, and 8), and they are grouped with their internal representations z (blue circles). (a) A scenario that k_z changes from $k_z = \{1, 2\}$ to $k_z = \{3\}$, but $k_{y,z} = 1$ does not change. Therefore, the size distribution $m(k)$ changes, but $H(Y, Z)$ does not change. (b) A scenario that $k_z = \{1, 2\}$ does not change, but $k_{y,z}$ changes from $k_{y,z} = 1$ to $k_{y,z} = \{1, 2\}$. Therefore, the size distribution $m(k)$ does not change, but $H(Y, Z)$ changes. (c) A pure clustering scenario that every x for a given z has the same label y.

In summary, except for the pure clustering, the objective $\mathcal{L}'$ for supervised learning has only two $m(k)$-dependent terms of $H(K)$ and $H(Z)$. This conclusion is the same with $\mathcal{L}$ for unsupervised learning. The objective functional, $\mathcal{L} = H(K) + \beta H(Z)$, has been intensively studied by Marsili and his colleagues [53,55,65,67,68]. Assuming that data follows a Boltzmann distribution, $k/M \propto \exp(-E(k))$, as in equilibrium thermodynamics, the entropy of Z can

be interpreted as an internal energy by ignoring a constant shift:

$$H(Z) = -\sum_k p(k) \log \frac{k}{M} = \sum_k p(k)E(k) = U, \tag{3.13}$$

where $p(k) = km(k)/M$. Then, the Shannon entropy $H(K)$ can be interpreted as thermodynamic entropy:

$$H(K) = -\sum_k p(k) \log p(k) = S. \tag{3.14}$$

Interpreting $\beta = -1/T$ as negative inverse temperature, one can define thermodynamic free energy:

$$\mathcal{F} = -T\mathcal{L} = U - TS. \tag{3.15}$$

Then, the maximization of $\mathcal{L}$ corresponds to the minimization of free energy $\mathcal{F}$. The optimal distribution $m(k)$, satisfying $\delta\mathcal{F}/\delta m(k) = 0$, has been derived to follow the scale-invariant power laws, $m(k) \propto k^{-\beta-1}$ [65]. This explains why power laws naturally arise in both unsupervised and supervised learning. I tempted to call this result as a thermodynamic second law of machine learning.

3.4 Conclusion

I studied internal representations z of data in machine learning. Song *et al.* first observed that restricted Boltzmann machines have special representations with a few frequent z and many rare z, of which frequency distributions follow power laws [53]. In this study, I showed that the scale-invariant representations are observed not only in the unsupervised learning but also in supervised learning. Furthermore, I derived that the scale invariance can naturally arise in machine learning using information theory. The critical representations correspond to entropy-maximized ones given pre-specified learning accuracy. If I define a group of data x that have the same z (compressed codes or emergent labels), the

frequencies of z can be interpreted as cluster sizes of x. Then, the maximum uncertainty of the cluster size distribution implies that the size of a cluster, to which a certain data x belongs, can be most flexibly determined. Therefore, at any given learning accuracy, z can show the criticality.

In this study, I examined compressing structures of neural networks with classical architectures, such as multi-layer perceptrons, vanilla autoencoders, convolutional neural networks, and restricted Boltzmann machines, although recent deep learning considers infinitely-wide networks and overparameterized models as well [69–71]. It remains future research to explore how our conjecture of *thermodynamic second law of machine learning* applies in modern architectures such as transformers [72].

The power laws, also known as the Pareto principle, have been ubiquitously observed in social and biological data including real neural activities [73–77]. Unlike the symbolic property of criticality in statistical mechanics, the emergence of criticality in those data does not require fine tuning [78–80]. Schwab et al. have shown that multivariate systems can generate the criticality without fine tuning when latent variables are involved in the systems [78]. I note that whereas these studies focused on the statistics of observed data x in real neural networks, our study explained the statistics of internal representation z in artificial neural networks. The criticality in the existing data that comprise the natural world and the emergence of criticality during the learning process, discussed in this study, are of great intellectual interest. A further investigation of criticality will stimulate cross-fertilization between the fields [81].

Chapter 4

Mirror descent: learning via dual geometry

4.1 Optimization for machine learning

Deep learning shows remarkable results in various fields [36–38]. Despite the numerous success, the fundamental principles of deep learning are still mysterious. Various studies on the representation of the latent space, distribution of the parameters of the well-trained model are being actively conducted. From a technical viewpoint, training deep neural networks is a complicated optimization problem in extremely high-dimensional parameter space. The geometry of the loss landscape is determined by all settings involved in training and it can be non-convex or even non-smooth. The ultimate goal of training is to find the global minimum of the loss function, which induces the generalization of the model.

As the difficulty of the task using machine learning increases and the model

becomes more complex, overparameterized model is mainly dealt with where the number of model parameters is much greater than the size of data points. In overparameterized model, it is known that there are infinitely many global minima of training data, so model can interpolate any points in training region [82, 83]. Among those tremendous minima, finding the true global minima that can explain the unseen test data is a key problem. Many studies have focused on the characteristics of the overparameterized model with respect to the gradient descent (GD)-based optimizations [70, 84–86].

In the machine learning field, since it is very hard to explore the geometry of the loss function simply using the instantaneous gradients, advanced optimization algorithms have been suggested [87–91]. In fact, there are numerous optimization algorithms that have not been extensively verified, and it would be important task to examine the applicability of those algorithms to machine learning.

In this chapter, I would like to review the traditional optimization algorithm so called mirror descent (MD) [92–94] and show excellent performance when training deep neural network. MD is the generalization of GD from the information geometry viewpoint [95]. MD utilizes the geometry of the dual space, mirror of the primal space, to update the model parameters in non-trivial way. Since GD is the simplest example of MD with the self dual structure where the primal and dual space are identical, it does not take advantage of this hidden structure.

As an alternative to overparameterized model, I train the general fully-connected multilayer perceptron (MLP) networks with small dataset, more realistic settings compared to training extremely heavy model with large dataset. Training performance varies depending on all kinds of settings, but the most

important factors are the activation function, objective function, and learning rate. To compare the performance in different loss landscape, I tested regression and classification models. I used sigmoid activation, mean squared error (MSE) in a regression model and ReLU activation, cross entropy loss are used in a classification model. Each model is trained with large and small learning rate, respectively, and the performance of different MD algorithms are compared with the Adam optimization [90]. It is confirmed in each experiment that the performance is remarkably improved when the optimal MD is applied, and the characteristics of MD contributing to these results are investigated.

This chapter is organized as follows. In section 4.2 and 4.3, I will review the basic concepts of the information geometry and MD, respectively. Experimental results of training overparameterized model are shown in section 4.4 and 4.5. And I discuss the characteristics of MD that contribute to the performance difference in section 4.6. Section 4.7 concludes with comments and future directions.

4.2 Information Geometry

Information geometry is a study on the invariant geometrical structure of a family of probability distributions. I consider a family $S = \{p(x, \theta)\}$ of probability distributions, where x is a random variables and θ is a n-dimensional vector parameter. This forms a manifold of which θ is coordinate system.

Among the various types of probability distributions, the most representative example is the exponential family

$$p(x, \theta) = \exp\{\langle \theta, x \rangle - \psi(\theta)\} \tag{4.1}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product and ψ corresponds to the normalization

factor known as the cummulant generating function or free energy. From the normalization condition $\int p(x,\theta)dx = 1$, ψ can be written as

$$\psi(\theta) = \log \int \exp(\langle \theta, x \rangle) dx \tag{4.2}$$

and it can be shown that $\psi(\theta)$ is a convex function. A dually flat Riemannian manifold is represented by dual coordinate and all the geometrical structure can be derived from the dual convex function. The dual coordinate is given by the Legendre transformation

$$\mu = \nabla \psi(\theta) \tag{4.3}$$

which is the expectation of x,

$$\mu = \mathbb{E}[x] = \int x p(x,\theta) dx. \tag{4.4}$$

The dual convex function φ, the Legendre dual of ψ, is defined by

$$\varphi(\mu) = \langle \mu, \theta \rangle - \psi(\theta) \tag{4.5}$$

From the convex function ψ, I can define the Riemannian metric given as the Hessian of ψ

$$g_{ij}(\theta) = \partial_i \partial_j \psi(\theta). \tag{4.6}$$

The dual Riemannian metric can also be defined in the same way with dual convex function

$$g^{ij}(\mu) = \partial^i \partial^j \varphi(\mu). \tag{4.7}$$

And it can be shown that the Riemannian metric in an exponential family (Eq. (4.1)) is the Fisher information matrix defined by

$$g_{ij}(\theta) = F_{ij} \equiv \mathbb{E}\left[\partial_i \log p(x,\theta) \partial_j \log p(x,\theta)\right] \tag{4.8}$$

I can define divergence function as a statistical distance between two point P and Q, of which coordinates are written as θ_P and θ_Q. Canonical examples of divergence function are f-divergence and Bregman divergences

1) f-divergence

$$D_f[p:q] = \int p(x,\theta) f\left(\frac{q(x,\theta)}{p(x\theta)}\right) dx \tag{4.9}$$

2) Bregman divergence

$$D_\psi[p:q] = \psi(p) - \psi(q) - \langle \nabla\psi(q), (p-q)\rangle . \tag{4.10}$$

Note that Kullback-Leibler (KL) divergence, a well-known function widely used in machine learning, is an example of f-divergence with $f(u) = -\log u$

$$D_{\mathrm{KL}}[p:q] = \int p(x,\theta) \log\left(\frac{p(x,\theta)}{q(x,\theta)}\right) dx. \tag{4.11}$$

One interesting property is that the second order Taylor series expansion of KL-divergence is given by the Fisher information matrix F

$$D_{\mathrm{KL}}[p(\theta) : p(\theta+d)] \approx \frac{1}{2} d^T F d. \tag{4.12}$$

If I want to know the update vector d that minimizes the loss function $\mathcal{L}(\theta)$ in distribution space, I have to minimize the following:

$$\begin{aligned} d^* &= \operatorname*{argmin}_\theta \left\{\mathcal{L}(\theta+d) + \lambda\left(D_{\mathrm{KL}}[p(\theta) : p(\theta+d)] - c\right)\right\} \\ &= \operatorname*{argmin}_\theta \left\{\mathcal{L}(\theta) + \nabla_\theta \mathcal{L}(\theta)^T d + \frac{1}{2}\lambda d^T F d - \lambda c\right\} \end{aligned} \tag{4.13}$$

where the second term in the right hand side of the first line is the Lagrange multiplier that constraints the constant shift. The solution that makes the derivative zero is given by

$$d^* \propto F^{-1} \nabla_\theta \mathcal{L}(\theta). \tag{4.14}$$

In this way, the method of updating the parameters in the steepest directions in the distance distribution is called the natural gradient descent [96]

$$\tilde{\nabla}_\theta \mathcal{L}(\theta) = F^{-1} \nabla_\theta \mathcal{L}(\theta). \tag{4.15}$$

Natural gradient descent assumes the parameter space as the Riemannian manifold and its geometrical structure is used to the update process.

4.3 Mirror descent

Gradient descent (GD) is the most fundamental algorithm for optimization problems. It aims to minimize the loss function iteratively based on the gradient of the loss function with respect to model parameters

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta_t) \tag{4.16}$$

where $\mathcal{L}$ and θ_t are loss function and parameters at time step t, respectively. If the loss function is as simpler as convex function, its gradient monotonically decreases and GD ultimately converges to the global minimum. However, the loss landscape of deep neural networks has a complex geometry, and the gradient frequently vanishes in the global minima and saddle points. Therefore, many advanced optimization techniques that do not simply depend on the instantaneous gradients have been developed.

As an motivation of mirror descent (MD), let us derive the familiar GD in the following way. If I want to minimize the function $\mathcal{L}(\theta)$, I could try to minimize its linear approximation alternatively:

$$\theta_{t+1} = \underset{\theta}{\operatorname{argmin}} \{ \mathcal{L}(\theta_t) + \langle \nabla_\theta \mathcal{L}(\theta_t), \theta - \theta_t \rangle + \frac{1}{2\eta} ||\theta - \theta_t||^2 \} \tag{4.17}$$

where the last term in the right hand side is the L_2 regularization to prevent θ from being too far from the point θ_t. By dropping the term that doesn't depend

on θ, I get

$$\theta_{t+1} = \underset{\theta}{\operatorname{argmin}}\{\langle \nabla_\theta \mathcal{L}(\theta_t), \theta \rangle + \frac{1}{2\eta}||\theta - \theta_t||^2\}, \tag{4.18}$$

and it exactly gives the definition of GD with learning rate η. Here I can replace the regularization term as general distance functions, so called proximity functions Ψ:

$$\theta_{t+1} = \underset{\theta}{\operatorname{argmin}}\{\eta \langle \nabla_\theta \mathcal{L}(\theta_t), \theta \rangle + \Psi(\theta, \theta_t)\}. \tag{4.19}$$

In particular, Bregman divergence is the representative example of the proximity function

$$D_\psi[\theta; \theta'] = \psi(\theta) - \psi(\theta') - \left\langle \nabla_\theta \psi(\theta'), (\theta - \theta') \right\rangle \tag{4.20}$$

where ψ is the arbitrary convex function. The update rule with Bregman divergence gives the definition of MD

$$\nabla_\theta \psi(\theta_{t+1}) = \nabla_\theta \psi(\theta_t) - \eta \nabla_\theta \mathcal{L}(\theta_t) \tag{4.21}$$

$$\theta_{t+1} = (\nabla_\theta \psi)^{-1}(\nabla_\theta \psi(\theta_t) - \eta \nabla_\theta \mathcal{L}(\theta_t)). \tag{4.22}$$

Note that the entire process of MD is purely determined by the choice of convex function ψ in Bregman divergence. The examples of the convex function and corresponding Bregman divergence are summarized in Table 4.1.

The literal interpretation of MD (Eq. (4.22)) is as follows. I first map the primal parameters θ_t to a dual parameter μ_t with the mirror map or link function $\nabla_\theta \psi(\theta)$, i.e., $\mu_t = \nabla_\theta \psi(\theta_t)$. Then, I update the dual parameter with the gradient computed in the primal space $\mu_{t+1} = \mu_t - \eta \nabla_\theta \mathcal{L}(\theta_t)$ and map μ_{t+1} back to a original space via inverse map $\theta_{t+1} = (\nabla_\theta \psi)^{-1}(\mu_{t+1})$. These processes are diagrammatically depicted in Figure 4.1.

Table 4.1 Examples of Bregman divergence.

Convex function $\psi(\theta)$	Bregman divergence $B_\psi(\theta, \theta')$
$\frac{1}{2}\|\|\theta\|\|_2^2$	$\frac{1}{2}\|\|\theta - \theta'\|\|^2$
$\sum_{i=1}^n (\theta_i \ln \theta_i - \theta_i)$	$\sum_{i=1}^n \left(\theta_i \ln \frac{\theta_i'}{\theta_i} - \theta_i + \theta_i'\right)$
$\sum_{i=1}^n e^{\theta_i}$	$\sum_{i=1}^n (\theta_i - \theta_i' + 1)e^{\theta_i'}$
$\frac{1}{2}\|\|\theta\|\|_q^2$	$\frac{1}{2}\|\|\theta\|\|_q^2 - \frac{1}{2}\|\|\theta'\|\|_q^2 - \left\langle \frac{\theta \odot \|\theta\|^{q-2}}{\|\|\theta\|\|_q^{q-2}}, \theta - \theta' \right\rangle$

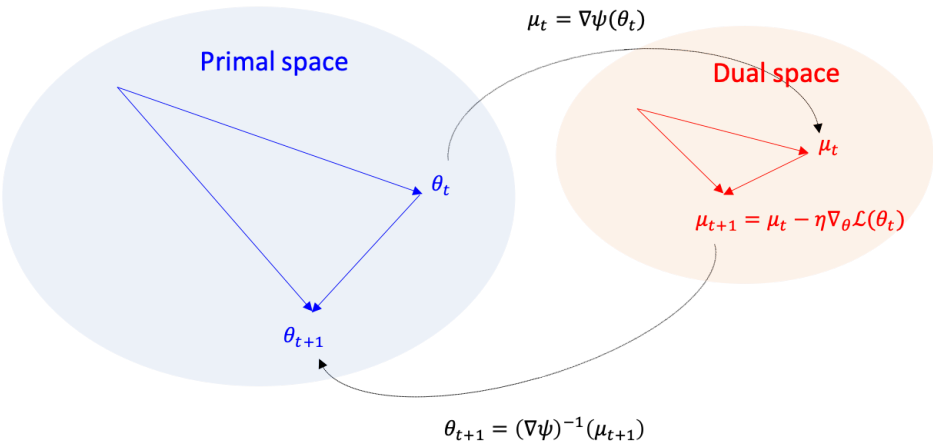

Figure 4.1 Diagram of mirror descent algorithm

It just looks like adding a map and inverse map before and after GD, respectively, but it has several interesting properties. In Equation (4.22), the primal parameters are transformed via mirror map $\nabla\psi$ before updating. Even though inverse map is finally applied, this transformation affects the updating process intrinsically. For this reason, MD is interpreted as implicit regularization to parameters. The effects of this regularization on MD performance will discussed in detail in the next section.

Another property is that MD is associated with natural gradient descent introduced in Equation (4.15) [97]. Using some chain-rules, the update of MD in the dual space can be expressed as follows

$$\mu_{t+1} = \mu_t - \eta \nabla_\theta \mathcal{L}(\theta_t) \tag{4.23}$$

$$= \mu_t - \eta \left[\nabla_\mu^2 \varphi(\mu_t)\right]^{-1} \nabla_\mu \mathcal{L}(\nabla_\mu \varphi(\mu_t)). \tag{4.24}$$

Here, if I assume that dual convex function is the free energy of the dual data distribution given as the exponential family, i.e. $p(x, \mu) = \exp\left(\langle \mu, x \rangle - \varphi(\mu)\right)$, the second derivative of it is the same as the Fisher information matrix ($F \equiv \nabla_\mu^2 \varphi(\mu)$), which gives the definition of natural gradient descent along the dual space.

The essence of MD is determining the type of the convex function defined in Bregman divergence. Although I can define infinitely many classes of Bregman divergence in principle, a practical example applicable to machine learning is the q-norm potential, i.e. $\psi(\theta) = \frac{1}{2}||\theta||_q^2$. And the dual convex function is given as p-norm potential $\varphi(\mu) = \frac{1}{2}||\mu||_p^2$ with the relation $(1/p + 1/q = 1)$. This particular types of MD is called p-norm algorithm [98]. Except for the trivial GD case where $p = q = 2$, other (p, q) combinations induce the different dual geometry and change the way that parameters are updated in the primal space.

Let us explicitly derive the update procedure of MD with q-norm potential

$\psi(\theta) = \frac{1}{2}||\theta||_q^2$. The derivative of the potential is given by

$$\partial_i \psi(\theta) = \frac{\theta_i |\theta_i|^{q-2}}{||\theta||_q^{q-2}}. \tag{4.25}$$

Thus, the mirror map from the primal to dual space is given by

$$\mu_t = \nabla_\theta \psi(\theta_t) = \theta_t \odot \left(\frac{|\theta_t|}{||\theta_t||_q} \right)^{q-2}. \tag{4.26}$$

And the update of the dual parameters is

$$\mu_{t+1} = \theta_t \odot \left(\frac{|\theta_t|}{||\theta_t||_q} \right)^{q-2} - \eta \nabla_\theta \mathcal{L}(\theta_t). \tag{4.27}$$

Finally, the inverse mirror map from dual to the primal space with the dual convex function $\varphi(\mu) = \frac{1}{2}||\mu||_p^2$ is given by

$$\theta_{t+1} = \left(\nabla_\theta \psi(\theta)\right)^{-1} (\mu_{t+1}) = \nabla_\mu \varphi(\mu_{t+1}) \tag{4.28}$$

$$= \mu_{t+1} \odot \left(\frac{|\mu_{t+1}|}{||\mu_{t+1}||_p} \right)^{p-2}. \tag{4.29}$$

Note that the mirror and inverse mirror map act as a scaling operator similar to weight normalization to primal parameters at t-step and dual parameters at $(t+1)$-step, respectively. For GD case with $(q = 2)$, note that the scale transformation becomes identity. When q is larger(smaller) than 2, mirror map acts as a scaling down(up) operator.

In general, most of the optimization techniques for training deep neural networks adjust the magnitude and direction of the gradients to treat the gradient vanishing or exploding problem. On the other hand, MD updates parameters by adjusting the scale of the parameters to the level of the gradient scale. MD internally utilizes the dual space of different scales, but by taking the inverse mirror map ultimately, it prevents the dramatic scale changes of the primal parameters, which induces the stable learning.

4.4 Experimental results of mirror descent

As an alternative example of an overparameterized model, I consider general full-connected multilayer perceptron (MLP) models with small dataset. Among the numerous tasks using MLP, I train regression and classification model, which are canonical examples of supervised learning. I assume that the loss geometry is changed by setting the activation function and objective function differently. MNIST image dataset [99] are trained with a part of the train data (1k) and the generalization is checked with all of the test data (10k). Candidates of MD with different q-norm potentials are summarized in Table 4.2. I also trained each model with Adam optimization as a benchmark.

Table 4.2 Candidates of convex and dual convex functions.

$\psi(\theta) = \frac{1}{2}\|\|\theta\|\|_q^2$, $\varphi(\mu) = \frac{1}{2}\|\|\mu\|\|_p^2$							
q	1.5	2	3	5	6	8	10
p	3	2	1.5	1.25	1.2	1.14	1.11

As a regression model, autoencoder with the 3 hidden layers, sigmoid activation and mean squared error (MSE) is used. Learning rate is set to be 0.01. I trained the model for 20 epochs with single batch size and compared the reconstruction performance of MD for different optimizers. The final loss of the train and test data are summarized in Table 4.3. For graphical simplification, the loss per iteration and reconstructed images of the test data for the some q-values are shown in Figure 4.2 and 4.3, respectively.

In regression problem, the training was slow in GD, while it accelerated at large q-values. However, higher q-norm potential did not always give better results, and among the candidates tried, the generalization performance was the best when $q = 8$.

Table 4.3 Regression loss of MNIST dataset for various MD and Adam optimizers with learning rate 0.01.

Regression	Learning rate=0.01	
Model	Train	Test
MD($q = 1.5$)	0.1136	0.1147
MD($q = 2$)	0.067	0.0694
MD($q = 3$)	0.0283	0.0343
MD($q = 5$)	0.0169	0.0214
MD($q = 6$)	0.015	0.0184
MD($q = 8$)	0.0112	**0.0163**
MD($q = 10$)	0.0111	0.0169
Adam	0.0095	0.0382

Table 4.4 Classification accuracy of MNIST dataset for various MD and Adam optimizers with learning rate 0.01.

Classification	Learning rate=0.01	
Model	Train	Test
MD($q = 1.5$)	8.5	9.6
MD($q = 2$)	9.8	10.2
MD($q = 3$)	90.6	81.9
MD($q = 5$)	100.0	**88.4**
MD($q = 6$)	100.0	87.8
MD($q = 8$)	83.6	71.1
MD($q = 10$)	62.5	55.8
Adam	92.0	79.9

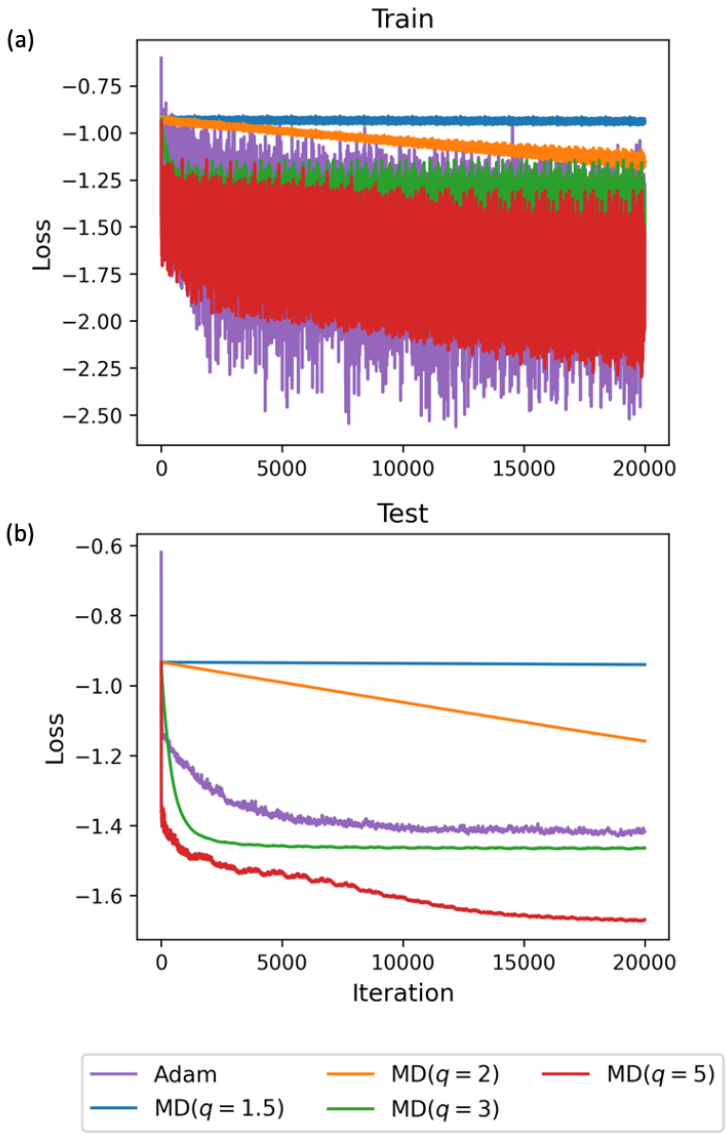

Figure 4.2 Regression results of MNIST dataset. Regression loss of the (a) train and (b) test data for various MD and Adam optimizers with learning rate 0.01.

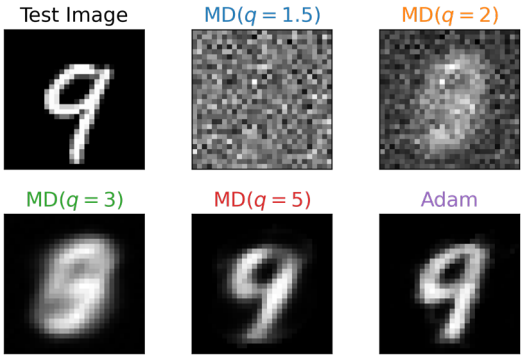

Figure 4.3 Reconstructed images of test data for various MD and Adam optimizers with learning rate 0.01.

Classification model is trained with the 3 hidden layers, ReLU activation and cross entropy as an objective function. Learning rate is set to be 0.01. I trained the model for 20 epochs with single batch size and compared the classification accuracy of MD for different optimizers. The results are shown in Table 4.4 and Figure 4.4, respectively. Like the regression problem, there was a performance improvement of MD with q-value larger than 2. Especially when $q = 5, 6$, the training label were predicted perfectly and showed better generalization than Adam.

4.5 Additional results of mirror descent

This section shows the additional results of the MD with different learning rate and dataset. Table 4.5, 4.6, and Figure 4.5 summarizes the results of MNIST dataset with learning rate 0.05. Compared with the small learning rate case, it can be seen that Adam is sensitive to the learning rate and the performance is highly degraded. However, MD is not significantly affected by the learning rate and maintains similar accuracy. Table 4.7, 4.8, Figure 4.6, and 4.7 shows the

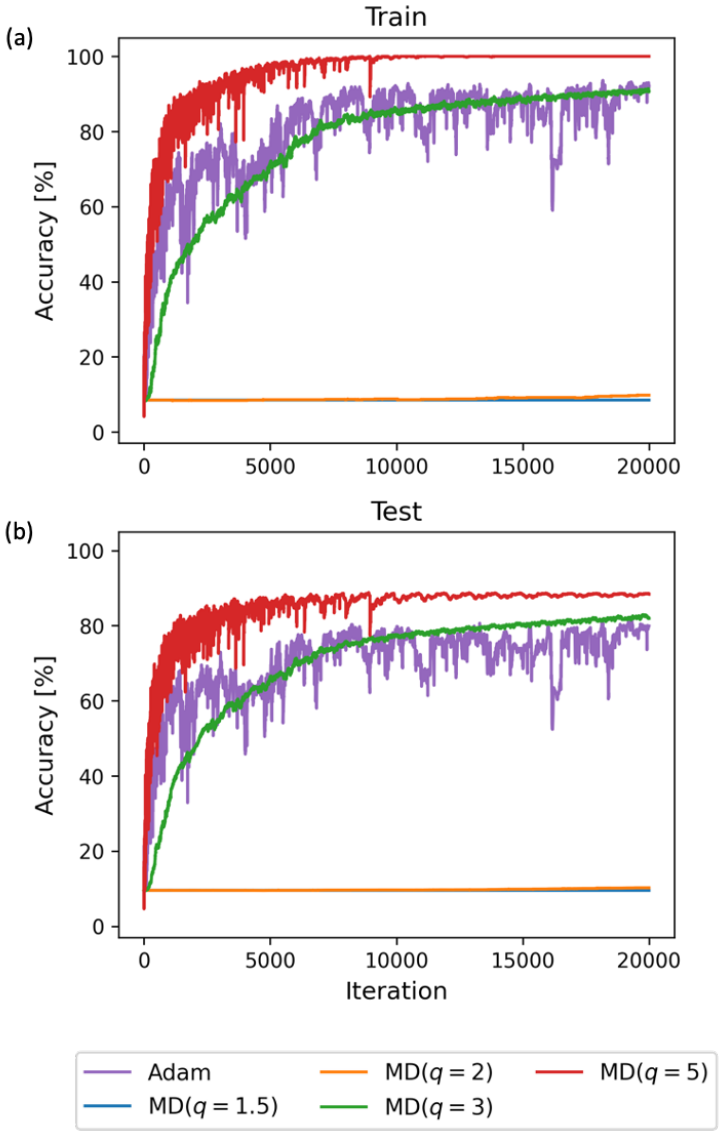

Figure 4.4 Classification results of MNIST dataset. Accuracy of the (a) train and (b) test data for various MD and Adam optimizers with learning rate 0.01.

same experiments with Fashion-MNIST dataset [100]. Although the optimal q-value for the best performance has changed, but it still showed better results than Adam with different learning rates.

Table 4.5 Regression loss of MNIST dataset for various MD and Adam optimizers with learning rate 0.05.

Regression	Learning rate=0.05	
Model	Train	Test
MD($q = 1.5$)	0.107	0.1086
MD($q = 2$)	0.0314	0.0362
MD($q = 3$)	0.0274	0.0335
MD($q = 5$)	0.0111	0.0158
MD($q = 6$)	0.0111	**0.0156**
MD($q = 8$)	0.0147	0.0213
MD($q = 10$)	0.0272	0.0322
Adam	0.0488	0.0703

Table 4.6 Classification accuracy of MNIST dataset for various MD and Adam optimizers with learning rate 0.05.

Classification	Learning rate=0.05	
Model	Train	Test
MD($q = 1.5$)	13.9	15.9
MD($q = 2$)	48.7	47.5
MD($q = 3$)	99.7	86.4
MD($q = 5$)	99.9	**87.7**
MD($q = 6$)	96.1	82.2
MD($q = 8$)	40.4	39.3
MD($q = 10$)	29.8	29.8
Adam	11.6	11.4

Table 4.7 Regression loss of Fashion-MNIST dataset for various MD and Adam optimizers.

Regression	Learning rate=0.01		Learning rate=0.05	
Model	Train	Test	Train	Test
MD($q = 1.5$)	0.0765	0.0852	0.0735	0.0804
MD($q = 2$)	0.0592	0.0621	0.0513	0.0445
MD($q = 3$)	0.0514	0.0428	0.0344	0.0233
MD($q = 5$)	0.0298	**0.0182**	0.0214	**0.0212**
MD($q = 6$)	0.0244	0.0193	0.0386	0.0269
MD($q = 8$)	0.0249	0.0233	0.0573	0.0482
MD($q = 10$)	0.0416	0.032	0.0555	0.05
Adam	0.0358	0.0476	0.0859	0.0887

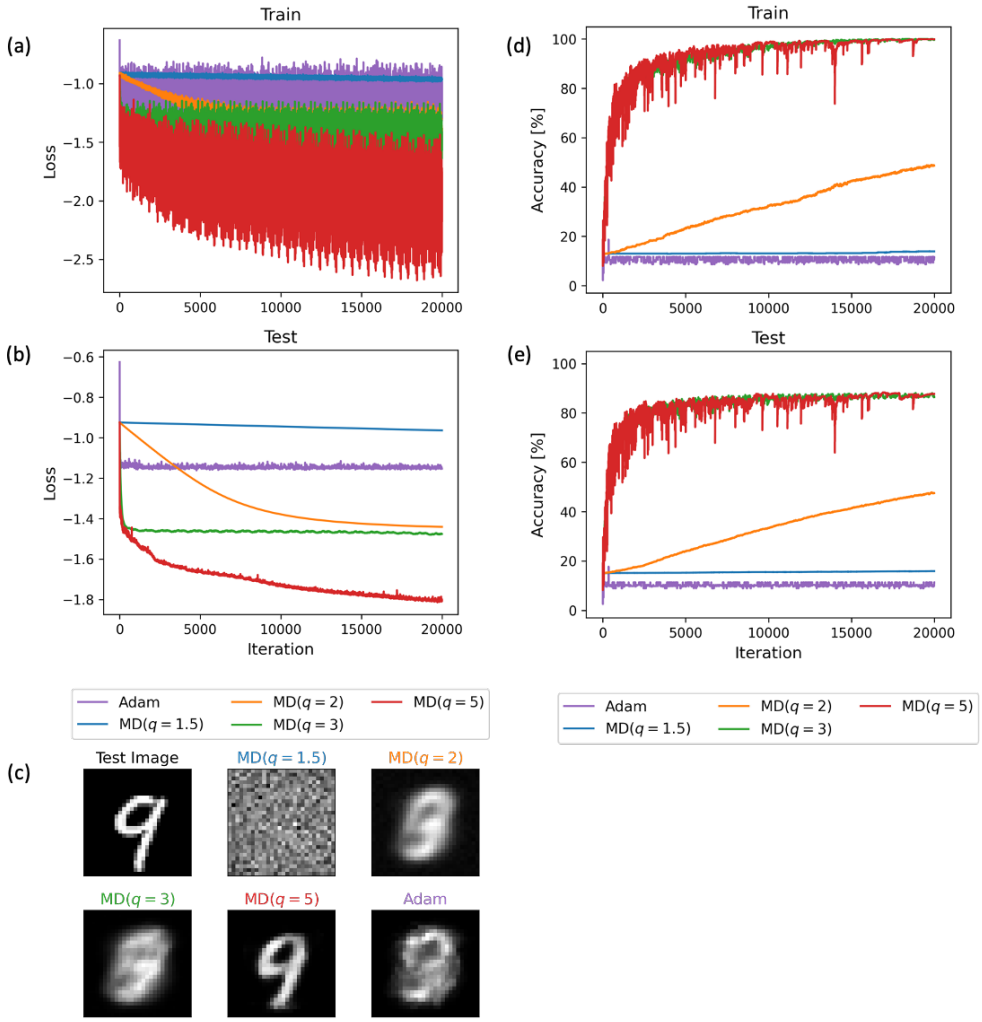

Figure 4.5 Regression(left) and classification(right) results of MNIST dataset. Regression loss and classification accuracy of the train (a,d), test (b,e) data, and reconstructed images of test data (c) for various MD and Adam optimizers with learning rate 0.05.

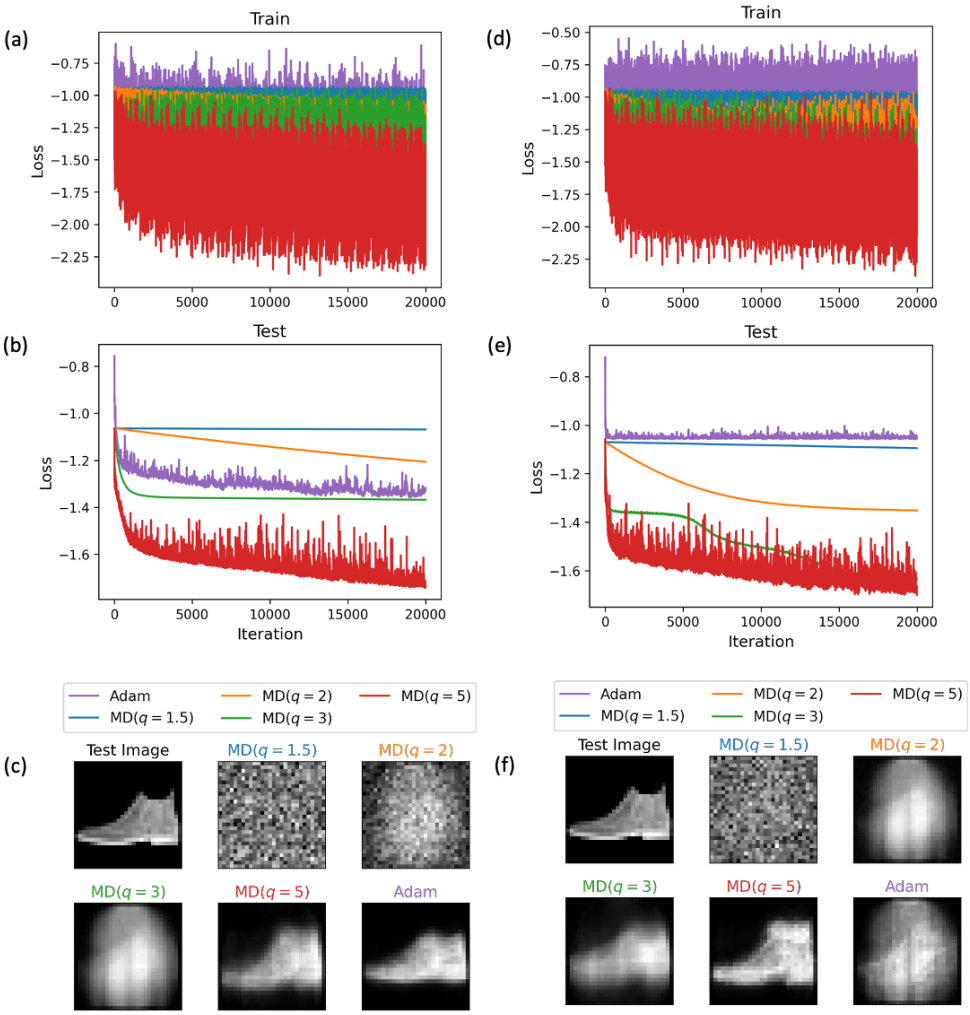

Figure 4.6 Regression results of Fashion-MNIST dataset. Regression loss of the train (a,d), test (b,e) data, and reconstructed images of test data (c,f) for various MD and Adam optimizers with learning rate 0.01 (left) and 0.05 (right).

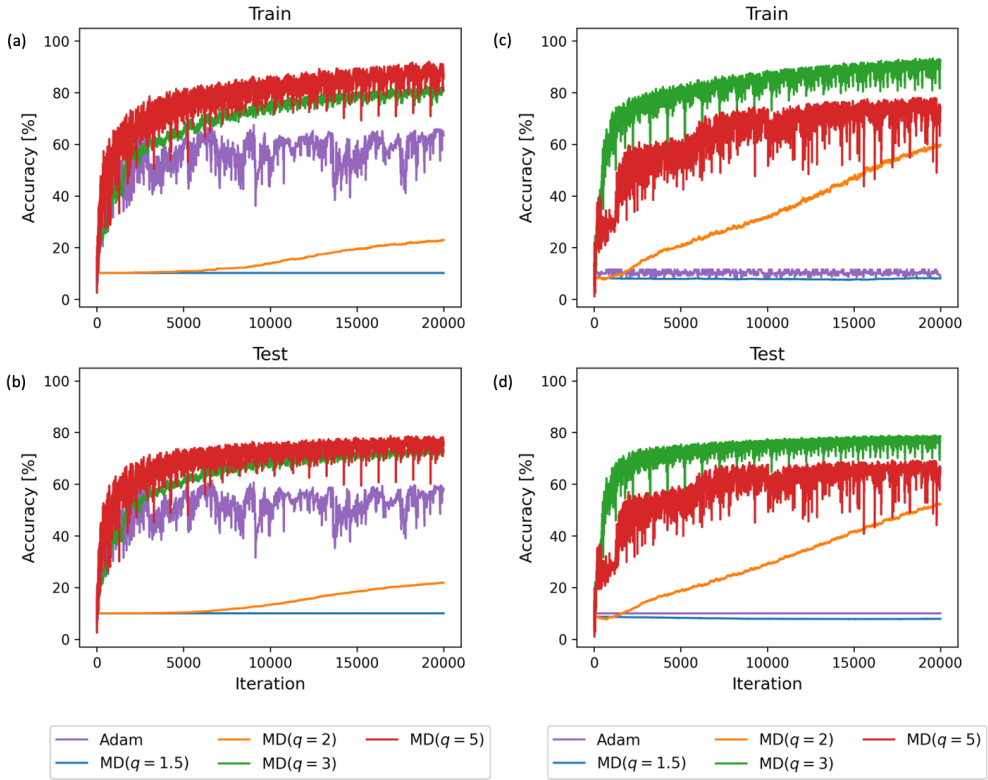

Figure 4.7 Classification results of Fashion-MNIST dataset. Classification accuracy of the train (a,d) and test data (b,e) for various MD and Adam optimizers with learning rate 0.01 (left) and 0.05 (right).

Table 4.8 Classification accuracy of Fashion-MNIST dataset for various MD and Adam optimizers.

Classification	Learning rate=0.01		Learning rate=0.05	
Model	Train	Test	Train	Test
MD($q = 1.5$)	10.2	10.0	8.1	7.9
MD($q = 2$)	22.9	21.9	59.9	52.4
MD($q = 3$)	81.1	73.2	92.3	**78.5**
MD($q = 5$)	87.5	**76.3**	74.5	66.3
MD($q = 6$)	72.8	66.1	20.4	20.5
MD($q = 8$)	30.3	28.9	10.7	10.0
MD($q = 10$)	10.7	10.1	10.7	10.0
Adam	63.4	58.1	8.6	10.0

4.6 Analysis of mirror descent

In the previous section, it was confirmed that the performance was remarkably enhanced when q-value was larger than 2 in common. As shown in Eq. (4.27), the role of q-norm potential in MD is to transform the primal parameters, which balances the scale of gradients. Large q reduces the scale of the primal parameters, which ultimately has the effect of resurrecting small gradients. This means that even if the gradients are small, training can be accelerated through additional process in dual space.

In fact, with conventional initialization schemes such as Xavier and He, the L_2-norm of the initial parameters and their gradients were order of 10^2 and 10^{-2}, respectively, i.e. $||\theta||_2 \sim \mathcal{O}(10^2), ||\nabla_\theta \mathcal{L}||_2 \sim \mathcal{O}(10^{-2})$. The difference in scale was so large that there was no substantial change with naive addition or subtraction like GD. On the other hand, when the large q-norm potential is

used to transform the primal parameters, the scale of the dual parameters is balanced with corresponding gradients, i.e. $||\mu||_2 \sim \mathcal{O}(10^{-1})(q = 5)$.

As shown in section 4.5, the performance of Adam optimization highly depend on the learning rate, while MD show robust result with respect to the learning rate. Learning rate plays a crucial role in optimization in general. However, in MD, q-norm potential has a more dominant effect than the learning rate in terms of scale balance between parameters and update terms. Therefore, the selection of the optimal q-value lowers the sensitivity of the empirically determined learning rate that affects the performance.

4.7 Conclusion

Various optimization algorithms have been developed to explore the complex loss geometry of deep neural networks, which gives a deeper understanding of GD. MD, a generalization of GD, has profound backgrounds both in primal and dual space perspectives. In primal space, it adjust the scale of parameters with implicit regularization. In dual space, MD is equivalent to the natural gradient descent along the dual coordinates. The key step of MD is the selection of an appropriate convex function. In particular, in the p-norm algorithm, it corresponds to finding an appropriate (p, q) combination, which balances the scale of parameters and gradients.

In this chapter, I verified the performance of MD when training general fully-connected MLP model with small dataset, a realistic surrogate of over-parameterized model. I conducted experiments by classifying the training settings according to the activation function, objective function, and learning rate. When proper q-norm potential is used, the performance of training and generalization of MD were remarkably enhanced and it gives the better result than the

state of the art optimization, Adam. More exactly, I concluded that adjusting q-norm value has the potential to solve the gradient vanishing and exploding problems, chronic situations encountered when training deep neural networks. It would be interesting subject how to automatically determine the optimal convex function in accordance with various training options and I leave it as the follow-up study.

Appendix A

Matrix-based kernel estimator of mutual information

The matrix-based kernel method [29] estimates Rényi's α-entropy for a random variable X by

$$H_\alpha(X) = \frac{1}{1-\alpha} \int_{x \in \mathcal{X}} f_X^\alpha(x) dx. \tag{A.1}$$

Let $X = \{x_1, x_2, ..., x_N\}$ denote N data points and $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a real valued positive definite kernel that defines a Gram matrix $K \in \mathbb{R}^{N \times N}$ as $K_{ij} = \kappa(x_i, x_j)$. The normalized Gram matrix is defined as

$$A_{ij} = \frac{1}{N} \frac{K_{ij}}{\sqrt{K_{ii} K_{jj}}}. \tag{A.2}$$

Then, the matrix-based Rényi's α-order entropy is given by

$$S_\alpha(A) = \frac{1}{1-\alpha} \log_2 \left[\mathrm{tr}(A^\alpha) \right] = \frac{1}{1-\alpha} \log_2 \left[\sum_{i=1}^{N} \lambda_i(A)^\alpha \right], \tag{A.3}$$

where $\lambda_i(A)$ denotes the i-th eigenvalue of A. In the limit of $\alpha \to 1$, Equation (A.3) is reduced to the Shannon entropy-like object

$$\lim_{\alpha \to 1} S_\alpha(A) = -\sum_{i=1}^{N} \lambda_i(A) \log_2 \lambda_i(A). \tag{A.4}$$

I used $\alpha = 1.01$ in this study. The joint entropy of two random variables X and Z can be defined as

$$S_\alpha(A, B) = S_\alpha\left(\frac{A \circ B}{\mathrm{tr}(A \circ B)}\right), \tag{A.5}$$

where A and B are Gram matrices of X and Z, respectively, and $A \circ B$ denotes the Hadamard product. From Equations (A.3) and (A.5), the mutual information in the kernel space is defined as

$$I_\alpha(X; Z) = S_\alpha(A) + S_\alpha(B) - S_\alpha(A, B). \tag{A.6}$$

The Gaussian kernel is commonly used:

$$\kappa_\sigma(x_i, x_j) = \exp\left(-\frac{||x_i - x_j||_F^2}{2\sigma^2}\right), \tag{A.7}$$

where $|| \cdot ||_F$ denotes the Frobenius norm. There are crucial factors that affect the estimation performance, such as the Gaussian kernel bandwidth σ and the scale and dimension of kernel input. The asymptotic behavior of entropy by varying σ can be denoted by

$$\lim_{\sigma \to 0} S_\alpha(A) = \log N \tag{A.8}$$

$$\lim_{\sigma \to \infty} S_\alpha(A) = 0. \tag{A.9}$$

Large-scale and high-dimensional features of input have the same effect as a small σ—the overestimation of entropy. In contrast, small-scale and low-dimensional features of input give the same effect as large σ, which results in

the underestimation of entropy. Therefore, proper hyperparameter tuning is required for σ to avoid excessively high or low saturation of entropy during training. Scott's rule [101], a simplified version of Silverman's rule [102], is commonly used for selecting the width of Gaussian kernels:

$$\sigma = \gamma N^{-1/(4+n)}, \tag{A.10}$$

where γ is an empirically determined constant. As Equation (A.10) is a monotonically increasing function with respect to feature dimension n, it compensates for higher feature dimension. I used $\gamma = 2$ for our experiments.

To validate the matrix-based kernel method, I consider a bivariate normal distribution as a simple example. Let us assume that two variables X_1 and X_2 follow a bivariate normal distribution:

$$\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \Sigma \right), \quad \Sigma = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}, \tag{A.11}$$

where μ_i and σ_i are mean and standard deviation of the variable X_i $(i = 1, 2)$, respectively, and ρ denotes their correlation strength. The entropy of each variable and their joint entropy are given as follows:

$$H(X_i) = \frac{1}{2}\log(2\pi e\sigma_i^2), \tag{A.12}$$

$$H(X_1, X_2) = \frac{1}{2}\log((2\pi e)^2|\Sigma|) = \log(2\pi e\sigma_1\sigma_2) + \frac{1}{2}\log(1-\rho^2). \tag{A.13}$$

Then, the mutual information between X_1 and X_2 can be exactly computed as

$$I(X_1; X_2) = H(X_1) + H(X_2) - H(X_1, X_2) = -\frac{1}{2}\log(1-\rho^2). \tag{A.14}$$

Now, I estimated $I(X_1; X_2)$ numerically using a binning method and the matrix-based kernel method with 1,000 samples generated from the bivariate

normal distribution of mean 0 and variance 1. Figure A.1 (a) shows (X_1, X_2) distributions under different correlation strengths. As shown in Figure A.1 (b), the theoretical value of $I(X_1; X_2)$ is consistent with the estimated values of the binning method with a proper quantizer (Bin = 20) and the matrix-based kernel method with a proper hyperparameter ($\gamma = 2$). Note that Bin represents the level of discretization for the continuous activity of X_i. For the binning method, Figure A.1 (c,d) show that its estimate of mutual information largely varies depending on the binning level and sample size. However, the matrix-based kernel method gives a robust estimate relatively less sensitive to the sample size (Figure A.1 (e)).

Saxe *et al.* observed that information estimation depends on the activation functions in a simple setup of a three neuron model ($x-z-y$) [15]. They sampled a scalar input x from a standard normal distribution of $\mathcal{N}(0, 1)$ and multiplied it by a constant weight w; subsequently, they determined the hidden activity $z = f(wx)$ using a nonlinear activation function $f(s)$. Then, they discretized z and estimated the input mutual information $I(x; z)$ using a binning method. When the unbounded activation function of $f(s) = \text{ReLU}(s)$ was used, $I(x; z)$ continued to increase with w. However, when the bounded activation function of $f(s) = \tanh(s)$ was used, $I(x; z)$ first increased with w, and then decreased as w increased. This is a natural result when the binning method is used to estimate the mutual information because large activities are saturated with large w (Figure 2 in [15]). I analyzed the same task with the matrix-based kernel method (Figure A.2 (a)). When the activation function is a sigmoid function of $f(s) = 1/(1 + \exp(-s))$, $I(x; z)$ does not decrease at large w; however, the absolute value looks different from the unbounded activation functions of linear ($f(s) = s$) and ReLU ($f(s) = \text{ReLU}(s)$). I also considered a more complex network with 100-dimensional input X sampled from $\mathcal{N}(0, 1)$. In this case,

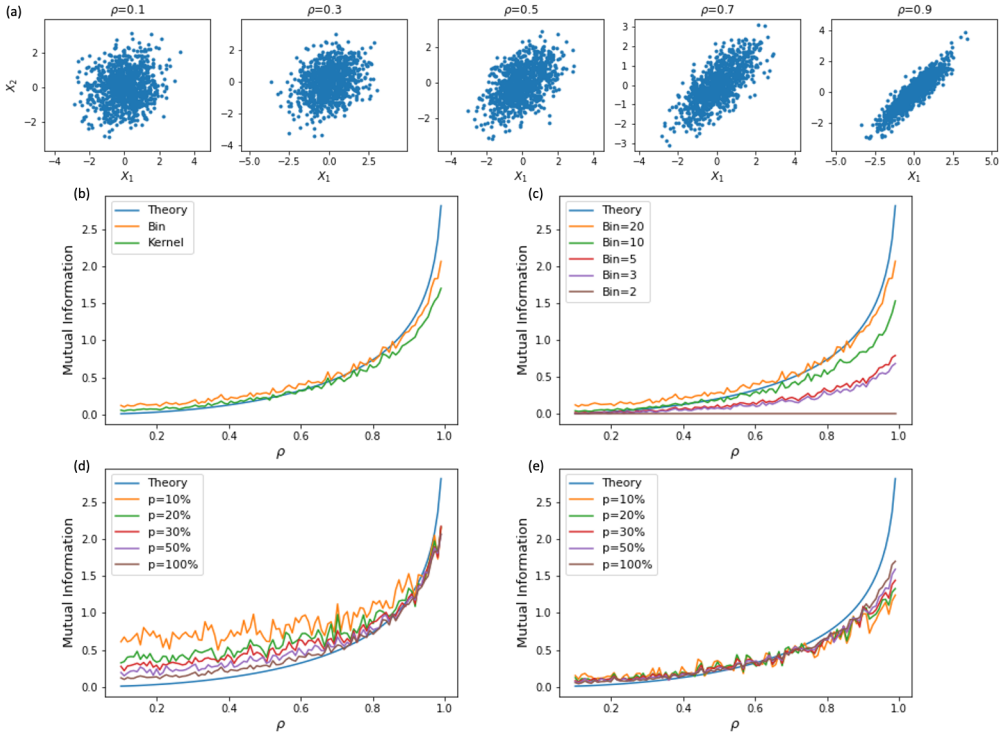

Figure A.1 Estimation of mutual information by binning and matrix-based kernel methods. (a) Distributions of two variables (X_1, X_2) following bivariate normal distributions with various correlation strengths of ρ. (b) Exact mutual information (Theory) and its optimal estimation by the binning method (Bin) and the matrix-based kernel method (Kernel). (c) Mutual information estimated by the binning method with various binning levels (Bin) of discretization for the continuous variables X_1 and X_2. Mutual information estimation with various sample sizes (p, percentage of the sample size to the entire data) of (d) the binning method and (e) the matrix-based kernel method.

weight W was represented by a 50×100 matrix whose elements were sampled from a uniform distribution of $\mathcal{U}(0, 1)$; then, the hidden activity Z becomes a 50-dimensional vector, i.e., $Z = f(WX)$. I then observed $I_\alpha(X; Z)$ while increasing the standard deviation of weight W (Figure A.2 (b)). I confirmed that $I_\alpha(X; Z)$ at large W does not decrease when a sigmoid activation function is used, like the unbounded activation functions of linear and ReLU. Therefore, this experiment demonstrated that the matrix-based kernel method is a robust estimation technique for bounded activation function and the simplifying phase cannot be attributed to the selected activation function.

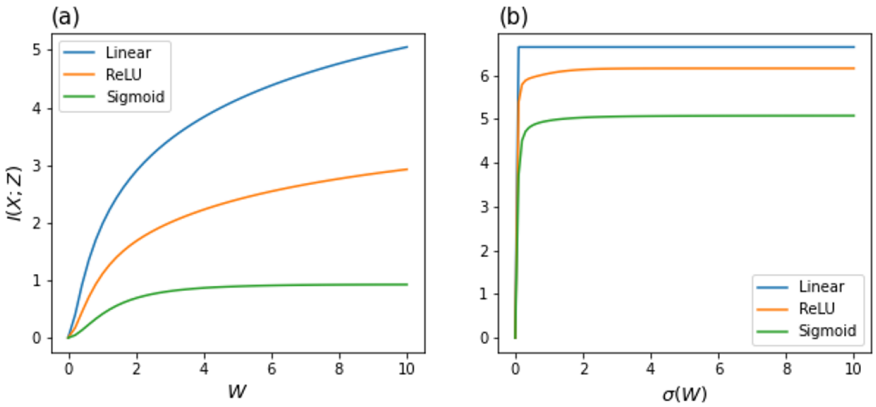

Figure A.2 Mutual information obtained by matrix-based kernel estimation. (a) Input mutual information in a three-neuron network ($x - z - y$). Input x was sampled from a standard normal distribution and the hidden activity was computed by $z = f(wx)$, where w is the weight and $f(s)$ is an activation function. Three different activation functions (linear, ReLU, and sigmoid) were used. (b) Input mutual information for a general setup with 100-dimensional input vector X and 50-dimensional hidden vector $Z = f(WX)$. In this setup, weight W was a 50×100 matrix whose elements were sampled from a uniform distribution.

Appendix B

Manifold learning of label autoencoder

LAE is a generative model that shapes its latent space using label classification. The explicit form of the LAE loss function is given as follows:

$$\mathcal{L}_{\mathrm{LAE}} = \frac{1}{N}\sum_{i=1}^{N} ||X_i - X_i'||^2 - \frac{\lambda}{N}\sum_{i=1}^{N}\sum_{j=1}^{n_Z} Y_{i,j} \log Z_{i,j}, \tag{B.1}$$

where N is the batch size and n_Z is the feature dimensionality of the label. $Z_{i,j}$ is the softmax output of encoder that predicts the j-th class of the i-th sample and $Y_{i,j}$ is the corresponding true label. The first term is a reconstruction error (MSE) and the second term is a regularization given by the classification error of the encoder. The regularization coefficient λ is set to 0.01. Figure B.1 shows the manifold learning of LAE when the one-hot vector corresponding to zero is changed to other digits as an input of the decoder. It is a two-dimensional submanifold embeded in a ten-dimensional label latent space.

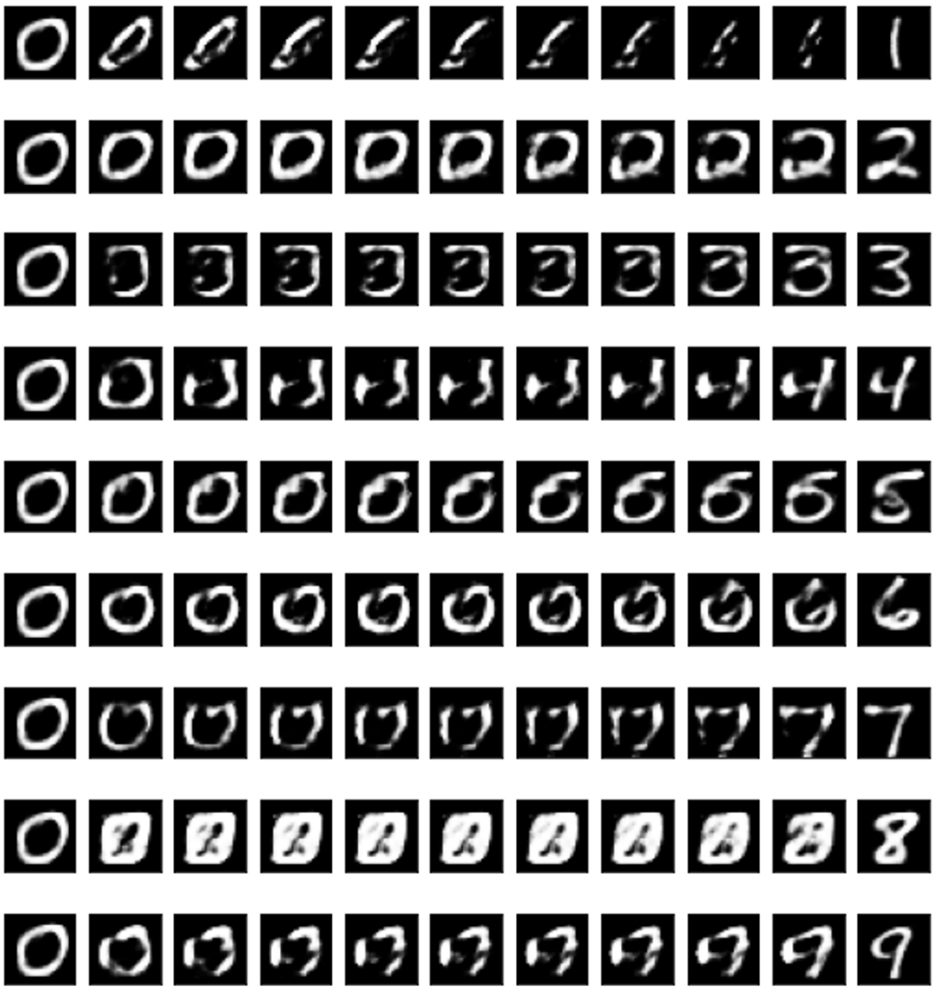

Figure B.1 Manifold learning in LAE. This represents the interpolation image when one-hot vector corresponding to zero; i.e., $Z_0 = [1, 0, 0, \cdots, 0]$, is transformed to other digits as an input of the LAE decoder part. For instance, the first row represents the reproduction X' decoded from $Z = aZ_0 + (1-a)Z_1$ by decreasing a from 1 to 0.

Bibliography

[1] C. E. Shannon, *A mathematical theory of communication*, *The bell system technical journal* **27** (1948) 379.

[2] T. M. Cover, *Elements of information theory.* John Wiley & Sons, 1999.

[3] E. T. Jaynes, *Information theory and statistical mechanics*, *Physical review* **106** (1957) 620.

[4] H. P. Yockey, *Information theory, evolution, and the origin of life.* Cambridge University Press, 2005.

[5] D. J. MacKay and D. J. Mac Kay, *Information theory, inference and learning algorithms.* Cambridge university press, 2003.

[6] N. Tishby, F. C. Pereira and W. Bialek, *The information bottleneck method*, *arXiv preprint physics/0004057* (2000) .

[7] S. Arimoto, *An algorithm for computing the capacity of arbitrary discrete memoryless channels*, *IEEE transactions on information theory* **18.1** (1972) 14.

[8] R. Blahut, *Computation of channel capacity and rate-distortion functions*, *IEEE transactions on information theory* **18.4** (1972) 460.

[9] I. E. Aguerri and A. Zaidi, *Distributed variational representation learning*, *IEEE transactions on pattern analysis and machine intelligence* **43.1** (2019) 120.

[10] Y. Uğur, I. E. Aguerri and A. Zaidi, *Vector Gaussian CEO problem under logarithmic loss and applications*, *IEEE transactions on information theory* **66.7** (2020) 4183.

[11] A. Zaidi, I. Estella-Aguerri and S. S. (Shitz), *On the information bottleneck problems: Models, connections, applications and information theoretic views*, *Entropy* **22.2** (2020) .

[12] A. A. Alemi, I. Fischer, J. V. Dillon and K. Murphy, *Deep variational information bottleneck*, ICLR, 2017.

[13] N. Tishby and N. Zaslavsky, *Deep learning and information bottleneck principle*, *IEEE Information Theory Workshowp (ITX)* (2015) 1.

[14] R. Shwartz-Ziv and N. Tishby, *Opening the black box of deep neural networks via information*, *arXiv:1703.00810* (2017) .

[15] A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey et al., *On the information bottleneck theory of deep learning*, *Journal of Statistical Mechanics* **124020** (2019) .

[16] I. Chelombiev, C. Houghton and C. O'Donnell, *Adaptive estimators show information compression in deep neural networks*, *arXiv:1902.09037* (2019) .

[17] K. Wickstrøm, S. Løkse, M. Kampffmeyer, S. Yu, J. Principe and R. Jenssen, *Information plane analysis of deep neural networks via*

matrix-based Renyi's entropy and tensor kernels, *arXiv:1909.11396* (2019) .

[18] S. Yu and J. C. Principe, *Understanding autoencoders with information theoretic concepts*, *Neural Networks* **117** (2019) 104.

[19] N. I. Tapia and P. A. Estévez, *On the Information Plane of autoencoders*, *International Joint Conference on Neural Networks* (2020) .

[20] H. Bourlard and Y. Kamp, *Auto-association by multilayer perceptrons and singular value decomposition*, *Biological cybernetics* **59.4** (1988) 291.

[21] P. Baldi and K. Hornik, *Neural networks and principal component analysis: learning from examples without local minima*, *Neural networks* **2.1** (1989) 53.

[22] A. Ng, *Sparse autoencoder*, *CS294A Lecture notes* (2011) .

[23] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio and P.-A. Manzagol, *Stacked denoising autoencoders: learning a useful representations in a deep network with a local denoising criterion*, *Journal of machine learning research* **11.12** (2010) .

[24] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, *arXiv:1312.6114* (2013) .

[25] E. Kodirov, T. Xiang and S. Gong, *Semantic autoencoder for zero-shot learning*, *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017) 3174.

[26] L. Le, A. Patterson and M. White, *Supervised autoencoders: Improving generalization performance with unsupervised regularizers*, *Advances in neural information processing systems* **31** (2018) 107.

[27] H. Kamyshanska and R. Memisevic, *The potential energy of an autoencoder*, *IEEE transactions on pattern analysis and machine intelligence* **37.6** (2014) 1261.

[28] A. Kolchinsky and B. D. Tracey, *Estimating mixture entropy with pairwise distances*, *Entropy* **19.7** (2017) 361.

[29] L. G. S. Giraldo, M. Rao and J. C. Principe, *Measures of entropy form data using infinitely divisible kernels*, *IEEE Transactions on Information Theory* **61.1** (2014) 535.

[30] S. Yu, L. G. S. Giraldo, R. Jenssen and J. C. Principe, *Multivariate extension of matrix-based Rényi's α-order entropy functional*, *IEEE Trans Pattern Anal Mach Intell* **42.11** (2019) 2960.

[31] B. C. Geiger, *On information plane analyses of neural network classifiers - a review*, *arXiv:2003.09671* (2020) .

[32] S. Lee and J. Jo. `https://github.com/Sungyeop/IPRL`, 2021.

[33] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, *Gradient-based learning applied to document recognition*, *Proceedings of the IEEE* (1998) 2278.

[34] H. Xiao, K. Rasul and R. Vollgraf, *Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms*, *arXiv:1708.07747* (2017) .

[35] G. Cohen, S. Afshar, J. Tapson and A. van Schaik, *EMNIST: Extending MNIST to handwritten letters*, International Joint Conference on Neural Networks, 2017.

[36] Y. LeCun, Y. Bengio and G. Hinton, *Deep learning*, *nature* **521** (2015) 436.

[37] Z. Ghahramani, *Probabilistic machine learning and artificial intelligence*, *Nature* **521** (2015) 452.

[38] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby et al., *Machine learning and the physical sciences*, *Reviews of Modern Physics* **91** (2019) 045002.

[39] Y. LeCun, Y. Bengio et al., *Convolutional networks for images, speech, and time series*, *The handbook of brain theory and neural networks* **3361** (1995) 1995.

[40] E. P. Van Nieuwenburg, Y.-H. Liu and S. D. Huber, *Learning phase transitions by confusion*, *Nature Physics* **13** (2017) 435.

[41] J. Carrasquilla and R. G. Melko, *Machine learning phases of matter*, *Nature Physics* **13** (2017) 431.

[42] B. S. Rem, N. Käming, M. Tarnowski, L. Asteria, N. Fläschner, C. Becker et al., *Identifying quantum phase transitions using artificial neural networks on experimental data*, *Nature Physics* **15** (2019) 917.

[43] F. Cichos, K. Gustavsson, B. Mehlig and G. Volpe, *Machine learning for active matter*, *Nature Machine Intelligence* **2** (2020) 94.

[44] B. Sanchez-Lengeling and A. Aspuru-Guzik, *Inverse molecular design using machine learning: Generative models for matter engineering*, *Science* **361** (2018) 360.

[45] F. Noé, S. Olsson, J. Köhler and H. Wu, *Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning*, *Science* **365** (2019) .

[46] A. Seif, M. Hafezi and C. Jarzynski, *Machine learning the thermodynamic arrow of time*, *Nature Physics* **17** (2021) 105.

[47] N. Tishby, F. C. Pereira and W. Bialek, *The information bottleneck method*, *arXiv preprint physics/0004057* (2000) .

[48] M. A. Kramer, *Nonlinear principal component analysis using autoassociative neural networks*, *AIChE journal* **37** (1991) 233.

[49] H. Bourlard and Y. Kamp, *Auto-association by multilayer perceptrons and singular value decomposition*, *Biological cybernetics* **59.4** (1988) 291.

[50] P. Baldi and K. Hornik, *Neural networks and principal component analysis: learning from examples without local minima*, *Neural networks* **2.1** (1989) 53.

[51] G. E. Hinton and R. R. Salakhutdinov, *Reducing the dimensionality of data with neural networks*, *science* **313** (2006) 504.

[52] J. Lindsey, S. A. Ocko, S. Ganguli and S. Deny, *A unified theory of early visual representations from retina to cortex through anatomically constrained deep cnns*, *arXiv preprint arXiv:1901.00945* (2019) .

[53] J. Song, M. Marsili and J. Jo, *Resolution and relevance trade-offs in deep learning*, *Journal of Statistical Mechanics: Theory and Experiment* (2018) 123406.

[54] M. E. Rule, M. Sorbaro and M. H. Hennig, *Optimal encoding in stochastic latent-variable models*, *Entropy* **22** (2020) 714.

[55] R. J. Cubero, J. Jo, M. Marsili, Y. Roudi and J. Song, *Statistical criticality arises in most informative representations*, *Journal of Statistical Mechanics: Theory and Experiment* **2019** (2019) 063402.

[56] S. Lee and J. Jo. `https://github.com/Sungyeop/Powerlaw_ML`, 2022.

[57] R. Salakhutdinov, A. Mnih and G. Hinton, *Restricted boltzmann machines for collaborative filtering*, in *Proceedings of the 24th international conference on Machine learning*, pp. 791–798, 2007.

[58] G. E. Hinton, *A practical guide to training restricted boltzmann machines*, in *Neural networks: Tricks of the trade*, pp. 599–619. Springer, 2012.

[59] G. E. Hinton, *Training products of experts by minimizing contrastive divergence*, *Neural computation* **14.8** (2002) 1771.

[60] P. Vincent, H. Larochelle, Y. Bengio and P.-A. Manzagol, *Extracting and composing robust features with denoising autoencoders*, in *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, 2008.

[61] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, *arXiv preprint arXiv:1312.6114* (2013) .

[62] D. P. Kingma and M. Welling, *An introduction to variational autoencoders*, *arXiv preprint arXiv:1906.02691* (2019) .

[63] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, *Gradient-based learning applied to document recognition*, *Proceedings of the IEEE* **86** (1998) 2278.

[64] A. Krizhevsky and G. Hinton, *Learning multiple layers of features from tiny images*, .

[65] M. Marsili, I. Mastromatteo and Y. Roudi, *On sampling and modeling complex systems*, *Journal of Statistical Mechanics: Theory and Experiment* **2013** (2013) P09003.

[66] J. Pearl, *Causality*. Cambridge University Press, 2009.

[67] A. Haimovici and M. Marsili, *Criticality of mostly informative samples: a bayesian model selection approach*, *Journal of Statistical Mechanics: Theory and Experiment* **2015** (2015) P10013.

[68] O. Duranthon, M. Marsili and R. Xie, *Maximal relevance and optimal learning machines*, *Journal of Statistical Mechanics: Theory and Experiment* **2021** (2021) 033409.

[69] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington and J. Sohl-Dickstein, *Deep neural networks as gaussian processes*, *arXiv preprint arXiv:1711.00165* (2017) .

[70] S. Oymak and M. Soltanolkotabi, *Overparameterized nonlinear learning: gradient descent takes the shortest path?*, International conference on machine learning, 2019.

[71] M. Geiger, S. Spigler, A. Jacot and M. Wyart, *Disentangling feature and lazy training in deep neural networks*, *Journal of Statistical Mechanics: Theory and Experiment* **2020** (2020) 113301.

[72] H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, T. Adler et al., *Hopfield networks is all you need*, *arXiv preprint arXiv:2008.02217* (2020) .

[73] M. E. Newman, *Power laws, pareto distributions and zipf's law*, *Contemporary physics* **46** (2005) 323.

[74] J. Hidalgo, J. Grilli, S. Suweis, M. A. Munoz, J. R. Banavar and A. Maritan, *Information-based fitness and the emergence of criticality in living systems*, *Proceedings of the National Academy of Sciences* **111** (2014) 10095.

[75] G. Tkačik, T. Mora, O. Marre, D. Amodei, S. E. Palmer, M. J. Berry et al., *Thermodynamics and signatures of criticality in a network of neurons*, *Proceedings of the National Academy of Sciences* **112** (2015) 11508.

[76] C. James, S. Azaele, A. Maritan and F. Simini, *Zipf's and taylor's laws*, *Physical Review E* **98** (2018) 032408.

[77] N. Tomen, J. M. Herrmann and U. Ernst, *The functional role of critical dynamics in neural systems*, vol. 11. Springer, 2019.

[78] D. J. Schwab, I. Nemenman and P. Mehta, *Zipf's law and criticality in multivariate data without fine-tuning*, *Physical review letters* **113** (2014) 068102.

[79] L. Aitchison, N. Corradi and P. E. Latham, *Zipf's law arises naturally when there are underlying, unobserved variables*, *PLoS computational biology* **12** (2016) e1005110.

[80] J. Touboul and A. Destexhe, *Power-law statistics and universal scaling in the absence of criticality*, *Physical Review E* **95** (2017) 012413.

[81] N. Savage, *How ai and neuroscience drive each other forwards*, *Nature* **571** (2019) S15.

[82] C. Zhang, S. Bengio, M. Hardt, B. Recht and O. Vinyals, *Understanding deep learning requires rethinking generalization*, *Communications of the ACM* **65.3** (2021) 107.

[83] S. Ma, R. Bassily and M. Belkin, *The power or interpolation: Understanding the effectiveness of sgd in modern over-parameterized learning*, in *International conference on machine learning*, 2018.

[84] N. Azizan and B. Hassibi, *Stochastic gradient/mirror descent: minmax optimality and implicit regularization*, *arXiv preprint arXiv:1806.00952* (2018) .

[85] S. Gunasekar, J. Lee, D. Soudry and N. Srebro, *Characterizing implicit bias in terms of optimization geometry*, pp. 1832–1841, International conference on machine learning, 2018.

[86] N. Azizan, S. Lale and B. Hassibi, *Stochastic mirror descent on overparameterized nonlinear models*, *IEEE Transactions on Neural Networks and Learning Systems* (2021) .

[87] N. Qian, *On the momentum term in gradient descent learning algorithms*, *Neural networks* **12.1** (1999) 145.

[88] Y. Nesterov, *A method for unconstrained convex minimization problem with the rate of convergence* $o(1/k^2$, *Doklady an ussr* **269** (1983) .

[89] J. Duchi, E. Hazan and Y. Singer, *Adaptive subgradient for online learning and stochastic optimization*, *Journal of machine learning research* **12.7** (2011) .

[90] D. P. Kingma and J. Ba, *Adam: a method for stochastic optimization*, *arXiv preprint arXiv:1412.6980* (2014) .

[91] S. Ruber, *An overview of gradient descent optimization algorithms*, *arXiv preprint arXiv:1609.04747* (2016) .

[92] A. S. Nemirovskij and D. B. Yudin, *Problem complexity and method efficiency in optimization*. 1983.

[93] A. Beck and M. Teboulle, *Mirror descent and nonlinear projected subgradient methods for convex optimization*, *Operations Research Letters* **31.3** (2003) .

[94] N. Cesa-Bianchi, P. Gaillard, G. Lugosi and G. Stolz, *Mirror descent meets fixed share (and feels no regres)*, *arXiv preprint arXiv:1203.3323* (2012) .

[95] S.-i. Amari and H. Nagaoka, *Methods of information geometry*, vol. 191. American mathematical soc., 2000.

[96] S.-i. Amari, *Natural gradient works efficiently in learning*, *Neural computation* **10.2** (1998) 251.

[97] G. Raskutti and S. Mukherjee, *The information geometry of mirror descent*, *IEEE transactions on information theory* **61.3** (2015) 1451.

[98] C. Gentile, *The robustness of the p-norm algorithms*, *Machine Learning* **53.3** (2003) .

[99] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, *Gradient-based learning applied to document recognition*, *Proceedings of the IEEE* (1998) 2278.

[100] H. Xiao, K. Rasul and R. Vollgraf, *Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms*, *arXiv:1708.07747* (2017) .

[101] D. W. Scott, *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[102] B. W. Silverman, *Density estimation for statistics and data analysis*, 26. CRC press, 1986.

초록

이 논문은 기계 학습이 다양한 문제들을 효과적으로 해결하는 원리에 대해 심층적으로 이해하는 것을 목표로 하며 정보 이론을 활용하여 신경망의 정보 흐름, 내부 표현, 변수 최적화를 연구한다. 먼저, 다양한 종류의 오토인코더에서 압축과 전송으로 이루어지는 정보 흐름을 시각화하고 각 모형들이 입력 데이터 재생성과 무관한 정보를 어떻게 제거하는지 알아본다. 둘째로, 지도 학습과 비지도 학습에서 신경망의 내부 표현이 크기 불변 멱법칙을 따르는 현상을 관찰하고 학습 과정에서 명시적인 규제 없이 어떻게 이 현상이 보편적으로 나타나는지 유도한다. 끝으로, 거울 하강 알고리즘을 정보 기하학 관점에서 소개하고 변수 공간의 쌍대 공간을 통해 학습 변수들이 어떻게 효과적으로 갱신되는지 알아본다. 결과적으로 정보 이론과 정보 기하학이 신경망 내부를 시각화하고, 분석하고, 최적화하는데 유용한 기법임을 확인한다.

주요어: 기계 학습, 심층 학습, 정보 이론, 정보 기하, 통계 물리

학번: 2012-23096

감사의 글

먼저 지도교수님이신 김석 교수님께 깊은 감사를 드립니다. 학부 인턴십부터 박사 졸업에 이르기까지 제가 방황할 때마다 해 주셨던 시기적절한 조언은 언제나 제가 앞으로 나아갈 방향에 대한 큰 가르침이었습니다. 기계 학습 연구지도를 해 주신 조정효 교수님께도 깊은 감사를 드립니다. 교수님의 좋은 지도 덕분에 흥미로운 연구를 하게 되었고 이 자리까지 올 수 있었습니다. 기계 학습의 입자물리 적용 연구를 지도해 주신 조원상 박사님께도 감사드립니다. 물리학과 기계 학습 전반에 걸쳐 다양한 이론과 실험들을 논의하며 많은 것을 배울 수 있었습니다. 학위논문 심사에 참여해 주신 이원종, 백용주, 현창봉 교수님께도 감사의 인사를 전합니다. 연구 방향에 대한 조언과 검토를 통해 저의 부족한 견문을 넓혀주신 이덕선, 송태근 교수님, 황원석 박사님, 황준오 군에게도 감사를 전합니다.

입자물리를 공부하던 시기에 연구실 선배로서 많은 도움을 주었던 김정민, 김준호 박사님과 오랜 시간을 같은 분야에서 함께 한 김동욱, 김정욱, 이기홍, 황윤석, 남궁준, 최선진 박사에게 늦게나마 감사의 인사를 드립니다. 늘 유쾌한 만남으로 즐거운 시간을 가졌던 윤영빈, 양명재 박사에게도 고마움을 전합니다. 비록 많은 시간을 함께 보내진 못 했지만 연구실 후배인 이은우, 최재혁 군에게도 고마움과 격려를 전합니다.

오랜기간 동거동락한 인생 친구 이기원 군에게 특별히 고마움을 전하며 서로의 진로에 조언과 격려를 아끼지 않은 안세일, 이한얼, 박한빈, 손승욱, 문효원, 박찬

주 군에게도 고마움을 표합니다. 힘들 때마다 옆에서 든든하게 있어준 바둑동아리 강정호 선배에게도 감사의 인사를 전합니다. 저의 긴 학위과정 동안 응원해 준 중, 고등학교 동기인 보균, 종현, 성구, 재호, 용준, 문태, 용건, 준석에게도 고맙다는 인사를 하고 싶습니다.

제 모습을 흐뭇하게 지켜보고 계실 아버지께도 마음 속으로 이 기쁨을 전합니다. 아들에 대한 무한한 관심과 교육자로서의 참된 가르침이 지금까지 제가 꾸준히 앞으로 전진할 수 있는 초석이 되었습니다. 저희 가족의 큰 버팀목이신 무상사 석성우 큰스님께 감사의 인사를 드립니다. 한 가족으로서 늘 진심어린 격려를 해 준 매형, 누나에게도 감사드리며 귀염둥이 조카 찬호, 나나를 통해 보답하겠습니다. 멀리서도 항상 응원해주신 여러 친척분들께도 감사드립니다. 끝으로 힘들었던 시간들을 함께 보내며 잘 되리라는 믿음을 언제나 잃지 않으신 어머니께 이 영광을 드립니다.