Chapter 4

Representation Learning via Lossy Compression

"*We compress to learn, and we learn to compress.*"

– *High-Dimensional Data Analysis*, Wright and Ma, 2022

Let us recap what we have covered so far. We have shown that it is possible to identify and access a distribution with arbitrary low-dimensional support, such as the one illustrated in Figure 4.1 (left), by gradually transforming the distribution of pure noise (high entropy) into the distribution to be learned via *iterative denoising*. We have shown that the denoiser at each iteration is directly connected to the gradient of the distribution's log-density via Tweedie's formula (3.2.23). We showed that in practice such a denoiser $\bar{\boldsymbol{x}}_\theta$ can be estimated using finite samples. Thus we developed a general way of learning or pursuing a data distribution.

Nevertheless, in this methodology the encoding of the distribution is implicit in the denoiser's functional form and parameters, if any. In fact, acute readers might have noticed that for a general distribution we have never explicitly specified what the functional form for the denoiser should be. In practice, people typically model it with some deep neural network that has an empirically designed architecture. In addition, although we know the above denoising process reduces the entropy, we do not know by how much, nor do we know the entropy of the intermediate and resulting distributions. Hence it would be very useful to know the (optimal) form of the denoising operation and how to measure the change in entropy. Finally, we showed at the end of the last Chapter that if the denoising process reduces the entropy *too much* it leads to learning only the training data, with no possible interpolation or extrapolation. The methodological innovation which resolves all of these issues will be developed in the Chapter to come.

Measure of information. Recall that our general goal is to model data from a continuous distribution with low-dimensional support. If we want to identify the simplest model that generates the data, we might consider three typical measures of parsimony: dimension, volume, or differential entropy. If we use dimension, then obviously the best model for a given dataset is the empirical distribution itself, which is zero-dimensional.[1] If we use the entropy of the data instead, then for all distributions with low-dimensional support, the differential entropy is always negative infinity. Or if we use the volume of the space spanned by the data, the volume of the support of any low-dimensional model is always zero. So, among all distributions with low-dimensional support that could have generated the same data samples, how can we decide which one is better based on these measures of parsimony that cannot distinguish among themselves at all?

In this Chapter, we discuss a framework that alleviates the above technical difficulty by associating the learned distribution with an explicit computable encoding and decoding scheme, following and making more explicit the second approach suggested at the end of Section 3.1.3. As we will see, such an approach essentially allows us to accurately approximate the entropy of the learned distributions in terms of a *lossy coding rate* associated with the coding scheme, as described and motivated (but not formalized) at the end of Section 3.3. Roughly speaking, such a measure can be viewed as a generalized notion of volume of the empirical data distribution. With such a measure, we can accurately quantify how much the entropy is reduced through a transformation of the distribution, say by a denoiser. In addition, with such a measure, we can derive, at least in principle, the optimal denoising operator that reduces the entropy the fastest.

Representation learning. Even though the diffusion and denoising process introduced in the previous chapter allows us, in principle, to learn an arbitrary low-dimensional distribution, it does not give an explicit representation of the distribution. As we will see, the learned distribution is to be repeatedly accessed and used for subsequent tasks (such as used as the prior for Bayesian inference in Chapter 7). Hence it is desirable that we can transform the data distribution, say by a mapping $f(\boldsymbol{x}, \theta)$, to some standard structured form that facilitates efficient access, say a mixture of (incoherent) linear subspaces or low-rank Gaussians, as illustrated in Figure 4.1 right. We will formalize this process as "representation learning" in this Chapter.

As we will also see, for an arbitrary distribution, the above measure of information (coding rate) is not (easily) computable. But for a mixture of subspaces or low-rank Gaussians, we can derive an accurate closed form for the measure. By optimizing this measure, it will allow us to find the optimal representation. As we will see in the next Chapter 5, optimizing the measure also allows us to derive explicit forms of the operator that performs the transformation $f(\boldsymbol{x}, \theta)$ in

[1] In the Introduction Chapter 1, we argued that one important goal of intelligence is to pursue low-dimensional structures in observed data. On the technical level though, one shall not take the word "low-dimensional" literally. It is the goal of this chapter to clarify precisely what low-dimensional structures we should be pursuing and how.

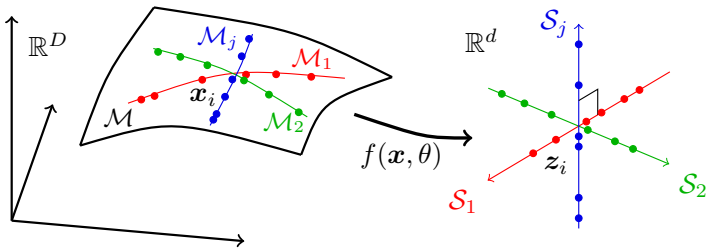


Figure 4.1: Learn a structured representation (right) for the data distribution (left). Notice that in addition to learning (and sampling) the distribution of the original data as we did in the previous Chapter 3, we here want to learn a transform $f(\boldsymbol{x}, \theta)$ for the data distribution so as to result in a representation that has desired better geometric or statistical properties, say linear and incoherent.

the most efficient way. This will lead to a principled explanation for the architecture of deep networks, as well as to more efficient deep-architecture designs.

4.1 Compression via Lossy Coding

In this section, we develop the *lossy coding rate* as a computable measure of complexity for distributions with low-dimensional support—a setting where classical measures such as entropy, dimension, and volume all fail to distinguish among candidate models. We first motivate the necessity of lossy coding with respect to these pathologies (Section 4.1.1), then show that the rate distortion function connects this coding-theoretic measure to the geometry of the data support via sphere covering (Theorem 4.1). This connection is the bridge between information-theoretic and geometric approaches to learning low-dimensional distributions. Finally, we develop a tight closed-form approximation to the lossy coding rate for Gaussians (Section 4.1.3) and an effective algorithm for clustering mixtures of low-dimensional Gaussians (Section 4.1.4), which will be generalized to nonlinear distributions in the following section.

4.1.1 Necessity of Lossy Coding

We have previously, multiple times, discussed a difficulty: if we learn the distribution from finite samples in the end, and our function class of denoisers contains enough functions, how do we ensure that we sample from the *true* distribution (with low-dimensional supports), instead of any other distribution that may produce those finite samples with high probability? Let us reveal some of the conceptual and technical difficulties with some concrete examples.

Example 4.1 (Volume, Dimension, and Entropy). For the example shown on the top of Figure 4.2, suppose we have taken some samples from a uniform distribution on a line (say in a 2D plane). The volume of the line or the sample

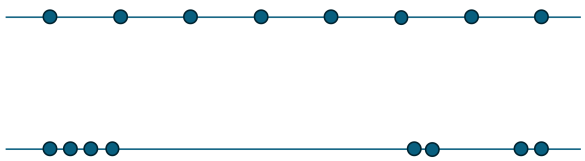


Figure 4.2: Eight points observed on a line with different geometric configurations; evenly distributed (top) or closely clustered (bottom).

sets is zero. Geometrically, the empirical distribution on the produced finite sample set is *the minimum-dimension* one which can produce the finite sample set.[2] But this is in seemingly contrast with yet another measure of complexity: entropy. The (differential) entropy of the line is negative infinity but the (discrete) entropy of this sample set is finite and positive. So we seem to have a paradoxical situation according to these common measures of parsimony or complexity: they cannot properly differentiate among (models for) distributions of low-dimensional supports at all, and some seem to differentiate them even in exactly opposite manners.[3] ■

Example 4.2 (Density). Consider the two sets of sampled data points shown in Figure 4.2. Geometrically, they are essentially the same: each set consists of eight points and each point has occurred with equal frequency 1/8th. The only difference is that for the second data set, some points are "close" enough to be viewed as having a higher density around their respective "cluster." Which one is more relevant to the true distribution that may have generated the samples? How can we reconcile such ambiguity in interpreting this kind of (empirical) distributions? ■

There is yet another technical difficulty associated with constructing an explicit encoding and decoding scheme for a data set. Given a sampled data set in $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$, how to design a coding scheme that is implementable on machines with finite memory and computing resources? Note that even representing a general real number requires an infinite number of digits or bits. Therefore, one may wonder whether the entropy of a distribution is a direct measure for the complexity of its (optimal) coding scheme. We examine this matter with another simple example.

Example 4.3 (Precision). Consider a discrete distribution $\boldsymbol{X} = [e, \pi]$ with equal probability 1/2 taking the values of the Euler number $e \approx 2.71828$ or the number $\pi \approx 3.14159$. The entropy of this distribution is $H = 1$, which suggests that one may encode the two numbers by a one-bit digit 0 or 1, respectively. But can you realize a decoding scheme for this code on a finite-state machine? The

[2] A set of discrete samples are all of zero dimension whereas the supporting line is one dimension.

[3] Of course, strictly speaking, differential entropy and discrete entropy are not directly comparable.

answer is actually no, as it takes infinitely many bits to describe either number precisely. ■

Hence, it is generally impossible to have an encoding and decoding scheme that can precisely reproduce samples from an arbitrary real-valued distribution.[4] But there would be little practical value to encode a distribution without being able to decode for samples drawn from the same distribution.

So to ensure that any encoding/decoding scheme is computable and implementable with finite memory and computational resources, we need to quantify the sample $\boldsymbol{x}$ and encode it only up to a certain precision, say $\epsilon > 0$. *By doing so, in essence, we treat any two data points as equivalent if their distance is less than* ϵ. More precisely, we would like to consider coding schemes

$$\boldsymbol{x} \mapsto \hat{\boldsymbol{x}} \tag{4.1.1}$$

such that the expected error caused by the quantization is bounded by ϵ. It is mathematically more convenient, and conceptually almost identical, to bound the expected *squared* error by ϵ^2, i.e.,

$$\mathbb{E}[d(\boldsymbol{x}, \hat{\boldsymbol{x}})^2] \leq \epsilon^2. \tag{4.1.2}$$

Typically, the distance d is chosen to be the Euclidean distance, or the 2-norm.[5] We will adopt this choice in the sequel.

4.1.2 Rate Distortion and Data Geometry

Of course, among all encoding schemes that satisfy the above constraint, we would like to choose the one that minimizes the resulting coding rate. For a given random variable $\boldsymbol{x}$ and a precision ϵ, this rate is known as the *rate distortion*, denoted as $\mathcal{R}_\epsilon(\boldsymbol{x})$.

To understand this rate distortion better, we turn to a deep theorem of information theory, originally proved in Shannon [Sha59], which relates this rate distortion to other information-theoretic quantities. To understand this theorem, we first need to define a measure of *difference* between two distributions known as the *Kullback-Leibler* (KL) divergence, or *relative entropy*, which is denoted by $\mathsf{KL}(\cdot \parallel \cdot)$ and defined in relation to the entropy and cross-entropy (see Section 3.1.3) as

$$\begin{aligned} \mathsf{KL}(p \parallel q) &= \mathrm{CE}(p \parallel q) - h(p) = \mathbb{E}_{\boldsymbol{x}\sim p}[-\log q(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim p}[-\log p(\boldsymbol{x})] && (4.1.3)\\ &= \mathbb{E}_{\boldsymbol{x}\sim p}[\log(p(\boldsymbol{x})/q(\boldsymbol{x}))]. && (4.1.4) \end{aligned}$$

The KL divergence is always non-negative, and equals zero if and only if $p = q$;[6] in the case where p and q have the same support and both are probability

[4] That is, if one wants to encode such samples precisely, the only way is to memorize every single sample.

[5] More generally, we can replace d^2 with any so-called *divergence*.

[6] Technically, this equality should be taken to mean "almost everywhere", i.e., except possibly on a set of zero measure (volume), since this set would not impact the value of any integral.

densities, this can be proved by the following calculation based on Jensen's inequality and the fact that the function $\log(\cdot)$ is strictly concave:

$$-\mathsf{KL}(p \parallel q) = -\mathbb{E}_{\boldsymbol{x}\sim p}[\log(p(\boldsymbol{x})/q(\boldsymbol{x}))] = \mathbb{E}_{\boldsymbol{x}\sim p}[\log(q(\boldsymbol{x})/p(\boldsymbol{x}))] \tag{4.1.5}$$

$$\leq \log \mathbb{E}_{\boldsymbol{x}\sim p}[q(\boldsymbol{x})/p(\boldsymbol{x})] = \log \int_{\mathbb{R}^D} \frac{q(\boldsymbol{\xi})}{p(\boldsymbol{\xi})} p(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi} \tag{4.1.6}$$

$$= \log \int_{\mathbb{R}^D} q(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi} = \log \mathbb{E}_{\boldsymbol{x}\sim q}[1] = \log 1 = 0. \tag{4.1.7}$$

Then, we define the *mutual information* $I(\cdot;\cdot)$, which is a measure of *dependence* between two random variables, as

$$I(\boldsymbol{x};\hat{\boldsymbol{x}}) = \mathsf{KL}(p(\boldsymbol{x},\hat{\boldsymbol{x}}) \parallel p(\boldsymbol{x})p(\hat{\boldsymbol{x}})), \tag{4.1.8}$$

With the previous definitions in place, we may write the rate distortion $\mathcal{R}_\epsilon$ in purely probabilistic terms as

$$\mathcal{R}_\epsilon(\boldsymbol{x}) = \min_{p(\hat{\boldsymbol{x}}|\boldsymbol{x}):\mathbb{E}[\|\boldsymbol{x}-\hat{\boldsymbol{x}}\|_2^2]\leq\epsilon^2} I(\boldsymbol{x};\hat{\boldsymbol{x}}). \tag{4.1.9}$$

Note that the minimization in (4.1.9) is over all conditional distributions $p(\hat{\boldsymbol{x}} \mid \boldsymbol{x})$ that satisfy the distortion constraint $\mathbb{E}[\|\boldsymbol{x}-\hat{\boldsymbol{x}}\|_2^2] \leq \epsilon^2$. Each such conditional distribution induces a joint distribution $p(\boldsymbol{x},\hat{\boldsymbol{x}}) = p(\hat{\boldsymbol{x}} \mid \boldsymbol{x})p(\boldsymbol{x})$, which determines the mutual information (4.1.8). Many convenient properties of the mutual information (and hence the rate distortion) are implied by corresponding properties of the KL divergence. From the definition, we know that $\mathcal{R}_\epsilon(\boldsymbol{x})$ is a *non-increasing* function in ϵ.

Example 4.4 (Rate distortion of Gaussian source (Theorem 13.3.3, [CT91])). Consider a Gaussian random variable $\boldsymbol{x} \sim \mathcal{N}(0,\sigma^2)$. For this case, it is well-known that the rate distortion function can be computed in closed form as:

$$\mathcal{R}_\epsilon(x) = \begin{cases} \frac{1}{2}\log\left(\sigma^2/\epsilon^2\right), & 0 \leq \epsilon \leq \sigma, \\ 0, & \epsilon > \sigma \end{cases}. \tag{4.1.10}$$

This extends in a particularly interesting way to the case of D dimensions. Suppose that $\boldsymbol{x} \sim \mathcal{N}(\mathbf{0},\boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma} \in \mathsf{PSD}(D)$ has eigenvalues $\lambda_1 \geq \cdots \geq \lambda_D \geq 0$. Then the rate distortion is obtainable via "water filling" as

$$\mathcal{R}_\epsilon(\boldsymbol{x}) = \frac{1}{2}\sum_{i=1}^{D} \log\left(\frac{\lambda_i}{\min\{\kappa,\lambda_i\}}\right), \tag{4.1.11}$$

where the "water level" κ satisfies $\sum_{i=1}^{D}\min\{\kappa,\lambda_i\} = \epsilon^2$. Intuitively, directions with variance λ_i larger than κ are effectively encoded, while directions with smaller variance are discarded, ensuring that the distortion constraint is satisfied with minimal mutual information. Please refer to Exercise 4.4 for a detailed derivation of this closed form. ■

Remark 4.1. As it turns out, the rate distortion is an implementable approximation to the entropy of $\boldsymbol{x}$ in the following sense. Assume that $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ are continuous random vectors. Then the mutual information can be written as

$$I(\boldsymbol{x}; \hat{\boldsymbol{x}}) = h(\boldsymbol{x}) - h(\boldsymbol{x} \mid \hat{\boldsymbol{x}}), \tag{4.1.12}$$

where $h(\boldsymbol{x} \mid \hat{\boldsymbol{x}}) = -\mathbb{E}[\log_2 p(\boldsymbol{x} \mid \hat{\boldsymbol{x}})]$ is the *conditional entropy* of $\boldsymbol{x}$ given $\hat{\boldsymbol{x}}$. Hence, the minimal coding rate is achieved when the difference between the entropy of $\boldsymbol{x}$ and the conditional entropy of $\boldsymbol{x}$ given $\hat{\boldsymbol{x}}$ is minimized among all distributions that satisfy the constraint: $\mathbb{E}[\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2] \leq \epsilon^2$.

In fact, it is not necessary to assume that $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ are continuous to obtain the above type of conclusion. For example, if both random vectors are instead discrete, we have after a suitable interpretation of the KL divergence for discrete-valued random vectors that

$$I(\boldsymbol{x}; \hat{\boldsymbol{x}}) = H(\boldsymbol{x}) - H(\boldsymbol{x} \mid \hat{\boldsymbol{x}}), \tag{4.1.13}$$

where $H(\boldsymbol{x})$ is defined in (3.1.1) of Section 3.1.3. More generally, advanced mathematical notions from measure theory can be used to define the mutual information (and hence the rate distortion) for arbitrary random variables $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$, including those with rather exotic low-dimensional distributions; see Cover and Thomas [CT91, §8.5].

Remark 4.2. Given a set of data points in $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$, one can always interpret them as samples from a uniform discrete distribution with equal probability $1/N$ on these N vectors. The entropy for such a distribution is $H(\boldsymbol{X}) = \frac{1}{N} \log_2 N$.[7] However, even if $\boldsymbol{X}$ is a uniform distribution on its samples, the coding rate $\mathcal{R}_\epsilon(\boldsymbol{X})$ achievable with a lossy coding scheme could be significantly lower than $H(\boldsymbol{X})$ if these samples are not so evenly distributed and many are clustered closely to each other. Therefore, for the second distribution shown in Figure 4.2, for a properly chosen quantization error ϵ, the achievable lossy coding rate can be significantly lower than coding it as a uniform distribution.[8] Also notice that, with the notion of rate distortion, the difficulty discussed in Example 4.3 also disappears: We can choose two rational numbers that are close enough to each of the two irrational numbers. The resulting coding scheme will have a finite complexity.

Example 4.5. Sometimes, one may face an opposite situation when we want to fix the coding rate first and try to find a coding scheme that minimizes the distortion. For example, suppose that we only want to use a fixed number of codes for points sampled from a distribution, and we want to know how to design the codes such that the average or maximum distortion is minimized during the encoding/decoding scheme. For example, given a uniform distribution on a unit square, we wonder how precisely we can encode points drawn from this

[7] Note again, even if we can encode these vectors with this coding rate, we cannot decode them with an arbitrary precision.

[8] Nevertheless, for this discrete uniform distribution, when ϵ is small enough, we always have $H(\boldsymbol{X}) \approx \mathcal{R}_\epsilon(\boldsymbol{X})$.

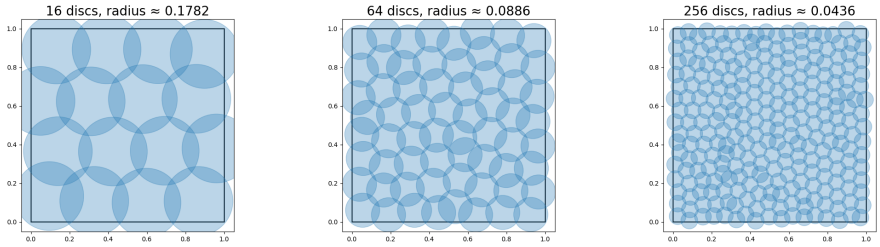


Figure 4.3: Approximations to the optimal solutions for 2^4, 2^6, and 2^8 discs covering a square, along with the corresponding radii, calculated using a heuristic optimization algorithm.

distribution, with say n bits. This problem is equivalent to asking what is the minimum radius (i.e., distortion) such that we can cover the unit square with 2^n discs of this radius. Figure 4.3 shows approximately optimal coverings of a square with $n = 4, 6, 8$, so that $2^n = 16, 64, 256$ discs, respectively. Notice that the optimal radii of the discs decreases as the number of discs 2^n increases. ■

It turns out to be a notoriously hard problem to obtain closed-form expressions for the rate distortion function (4.1.9) for general distributions $p(\boldsymbol{x})$. However, as Example 4.5 suggests, there are important special cases where the *geometry* of the support of the distribution $p(\boldsymbol{x})$ can be linked to the rate distortion function and hence to the optimal coding rate at distortion level ϵ. In fact, this example can be generalized to any setting where the support of $p(\boldsymbol{x})$ is a sufficiently regular compact set—including low-dimensional distributions—and $p(\boldsymbol{x})$ is uniformly distributed on its support. This covers a vast number of cases of practical interest. We formalize this notion in the following result, which establishes this property for a special case.

Theorem 4.1. *Suppose that $\boldsymbol{x}$ is a random variable such that its support $K \doteq \text{Supp}(\boldsymbol{x})$ is a compact set. Define the covering number $\mathcal{N}_\epsilon(K)$ as the minimum number of balls of radius ϵ that can cover K, i.e.,*

$$\mathcal{N}_\epsilon(K) \doteq \min\left\{ n \in \mathbb{N} \colon \exists \boldsymbol{p}_1, \ldots, \boldsymbol{p}_n \in K \ \text{s.t.}\ K \subseteq \bigcup_{i=1}^{n} B_\epsilon(\boldsymbol{p}_i) \right\}, \tag{4.1.14}$$

where $B_\epsilon(\boldsymbol{p}) = \{\boldsymbol{\xi} \in \mathbb{R}^D \mid \|\boldsymbol{\xi} - \boldsymbol{p}\|_2 \le \epsilon\}$ is the Euclidean ball of radius ϵ centered at $\boldsymbol{p}$. Then it holds

$$\mathcal{R}_\epsilon(\boldsymbol{x}) \le \log_2 \mathcal{N}_\epsilon(K). \tag{4.1.15}$$

If, in addition, $\boldsymbol{x}$ is uniformly distributed on K and K is a union of mutually orthogonal low-dimensional hyperspheres,[9] *then a matching lower bound holds:*

$$\mathcal{R}_\epsilon(\boldsymbol{x}) \ge \log_2 \mathcal{N}_\epsilon(K) - O(D). \tag{4.1.16}$$

[9] In fact, it is possible to treat highly irregular K, such as compact manifolds or even fractals, with a parallel result, but its statement becomes far more technical: cf. Riegler et al. [RBK18; RKB23]. We give a simple proof in Section B.3 which shows the result for

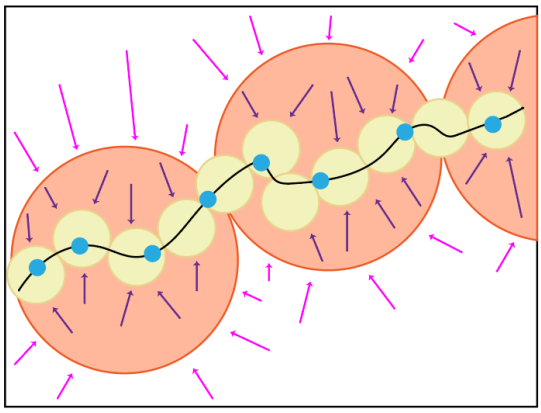


Figure 4.4: **The approximation of a low-dimensional distribution by ϵ balls.** We can see that as the ϵ parameter shrinks, the union of ϵ-balls approximates the support of the true distribution (black) increasingly well. Furthermore, the associated denoisers (whose input-output mapping is given by the provided arrows) obtained by approximating the true distribution by a mixture of Gaussians, each with covariance $(\epsilon^2/D)\boldsymbol{I}$, increasingly well-approximate the true denoisers. At large ϵ, such denoisers do not point near the true distribution at all, whereas at small ϵ they closely approximate the true denoisers. Theorem 4.1 establishes that this approximation characterizes the rate distortion function at small distortions ϵ, unifying the parallel approaches of coding rate minimization and denoising for learning low-dimensional distributions without pathologies.

Proof. A proof of this theorem is beyond the scope of this book and we defer it to Section B.3. □

The upper bound in Theorem 4.1 requires only that $\text{Supp}(\boldsymbol{x})$ is a compact set, while our proof of the lower bound requires stronger regularity assumptions: specifically, that $\boldsymbol{x}$ is uniformly distributed on $\text{Supp}(\boldsymbol{x})$ and that $\text{Supp}(\boldsymbol{x})$ is a mixture of low-rank Gaussians (with appropriate normalization).[10] Under these assumptions, the implication of Theorem 4.1 can be summarized as follows: for sufficiently accurate coding of the distribution of $\boldsymbol{x}$ (i.e., as $\epsilon \to 0$), the minimum rate distortion coding framework is equivalent to the sphere packing problem on the support of $\boldsymbol{x}$. So the rate distortion can be thought of as a "probability-aware" way to approximate the support of the distribution of $\boldsymbol{x}$ by a mixture of many small balls (Figure 4.4).

unions of orthogonal hyperspheres, an assumption that is equivalent to a mixture of low-rank Gaussians with orthogonal supports when the dimension of each subspace is sufficiently large. See Section B.3 for further discussion of this assumption.

[10] Later in this Chapter, in Section 4.2.3, we will see an objective function whose maxima are normalized mixtures of low-rank Gaussians that correspond to the input data in a certain way. Chapter 5 will further develop this framework for deep learning, showing that the structural assumption we make to prove the lower bound is not overly restrictive.

We now discuss another connection between this and the denoising-diffusion-entropy complexity hierarchy we discussed earlier in this chapter.

Remark 4.3. The key ingredient in the proof of the lower bound in Theorem 4.1 is an important result from information theory known as the *Shannon lower bound* for the rate distortion, named after Claude Shannon, who first derived it in a special case [Sha59]. It asserts the following estimate for the rate distortion function, for any random variable $\boldsymbol{x}$ with a density $p(\boldsymbol{x})$ and finite expected squared norm [LZ94]:

$$\mathcal{R}_\epsilon(\boldsymbol{x}) \geq h(\boldsymbol{x}) - \log_2 \operatorname{vol}(B_\epsilon) - C_D, \tag{4.1.17}$$

where $C_D > 0$ is a constant depending only on D. Moreover, this lower bound is actually sharp as $\epsilon \to 0$: that is,

$$\lim_{\epsilon \to 0} \mathcal{R}_\epsilon(\boldsymbol{x}) - [h(\boldsymbol{x}) - \log_2 \operatorname{vol}(B_\epsilon) - C_D] = 0. \tag{4.1.18}$$

So when the distortion ϵ is small, we can think solely in terms of the Shannon lower bound, rather than the (generally intractable) optimization problem defining the rate distortion (4.1.9).

The Shannon lower bound is the bridge between the coding rate, entropy minimization/denoising, and geometric sphere packing approaches for learning low-dimensional distributions. Notice that in the special case of a uniform density $p(\boldsymbol{x})$, (4.1.17) becomes

$$\mathcal{R}_\epsilon(\boldsymbol{x}) \geq -\int_K \frac{1}{\operatorname{vol}(K)} \log_2 \frac{1}{\operatorname{vol}(K)} \mathrm{d}\boldsymbol{\xi} - \log_2 \operatorname{vol}(B_\epsilon) - C_D \tag{4.1.19}$$

$$= \log_2 \operatorname{vol}(K) / \operatorname{vol}(B_\epsilon) - C_D. \tag{4.1.20}$$

The ratio $\operatorname{vol}(K)/\operatorname{vol}(B_\epsilon)$ approximates the number of ϵ-balls needed to cover K by a worst-case argument, which is accurate for sufficiently regular sets K when ϵ is small (see Section B.3 for details). Meanwhile, recall the Gaussian denoising model $\boldsymbol{x}_\epsilon = \boldsymbol{x} + \epsilon \boldsymbol{g}$ from earlier in the Chapter, where $\boldsymbol{g} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ is independent of $\boldsymbol{x}$. Interestingly, the differential entropy of the joint distribution $(\boldsymbol{x}, \epsilon \boldsymbol{g})$ can be calculated as

$$h(\boldsymbol{x}, \epsilon \boldsymbol{g}) = -\int p(\boldsymbol{\xi}) p(\boldsymbol{\gamma}) \log_2 p(\boldsymbol{\xi}) p(\boldsymbol{\gamma}) \mathrm{d}\boldsymbol{\xi} \mathrm{d}\boldsymbol{\gamma} \tag{4.1.21}$$

$$= h(\boldsymbol{x}) + h(\epsilon \boldsymbol{g}). \tag{4.1.22}$$

We have seen the Gaussian entropy calculated in Equation (3.1.4): when ϵ is small, it is equal, up to additive constants, to the quantity $-\log_2 \operatorname{vol}(B_\epsilon)$ we have seen in the Shannon lower bound. In certain special cases (e.g., data supported on incoherent low-rank subspaces), when ϵ is small and the support of $\boldsymbol{x}$ is sufficiently regular, the distribution of $\boldsymbol{x}_\epsilon$ can even be well-approximated locally by the product of the distributions $p(\boldsymbol{x})$ and $p(\boldsymbol{g})$, justifying the above computation. Hence the Gaussian denoising process yields yet another interpretation of the Shannon lower bound, as arising from the entropy of a noisy version of $\boldsymbol{x}$, with noise level proportional to the distortion level ϵ.

Thus, this finite rate distortion approach via sphere covering re-enables or generalizes all previous measures of complexity of the distribution, allowing us to differentiate between and rank different distributions in a unified way. These interrelated viewpoints are visualized in Figure 4.4.

For a general distribution at finite distortion levels, it is typically impossible to find its rate distortion function in an analytical form. One must often resort to numerical computation[11]. Nevertheless, as we will see, in our context we often need to know the rate distortion as an explicit function of a set of data points or their representations. This is because we want to use the coding rate as a measure of the goodness of the representations. An explicit analytical form makes it easy to determine how to transform the data distribution to improve the representation. So, we should work with distributions whose rate distortion functions take explicit analytical forms. To this end, we start with the simplest, and also the most important, family of distributions.

4.1.3 Lossy Coding Rate for a Low-Dimensional Gaussian

Now suppose we are given a set of data samples in $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$ from any distribution.[12] We would like to come up with a constructive scheme that can encode the data up to certain precision, say

$$\boldsymbol{x}_i \mapsto \hat{\boldsymbol{x}}_i, \quad \text{subject to} \quad \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|_2 \leq \epsilon. \tag{4.1.23}$$

Notice that this is a sufficient, explicit, and interpretable condition which ensures that the data are encoded such that $\frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|_2^2 \leq \epsilon^2$. This latter inequality is exactly the rate distortion constraint for the provided empirical distribution and its encoding. For example, in Example 4.5, we used this simplified criterion to explicitly find the minimum distortion and explicit coding scheme for a given coding rate.

Without loss of generality, let us assume the mean of $\boldsymbol{X}$ is zero, i.e., $\frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i = \boldsymbol{0}$. Without any prior knowledge about the nature of the distribution behind $\boldsymbol{X}$, we may view $\boldsymbol{X}$ as sampled from a Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$ with the covariance[13]

$$\boldsymbol{\Sigma} = \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top. \tag{4.1.24}$$

Notice that geometrically $\boldsymbol{\Sigma}$ characterizes an ellipsoidal region where most of the samples $\boldsymbol{x}_i$ reside.

We may view $\hat{\boldsymbol{X}} = [\hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_N]$ as a noisy version of $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$:

$$\hat{\boldsymbol{x}}_i = \boldsymbol{x}_i + \boldsymbol{w}_i, \tag{4.1.25}$$

[11] Interested readers may refer to [Bla72] for a classic algorithm that computes the rate distortion function numerically for a discrete distribution.

[12] Or these data points could be viewed as an (empirical) distribution themselves.

[13] It is known that given a fixed variance, the Gaussian achieves the maximal entropy. That is, it gives an upper bound for what the worst case could be in terms of possible coding rate.

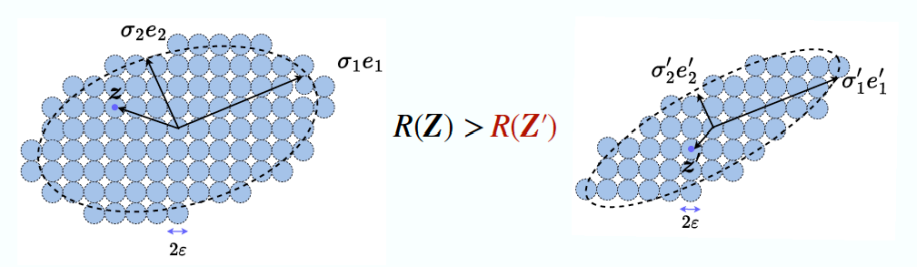


Figure 4.5: Covering the region spanned by the data vectors using ϵ-balls. The larger the volume of the space, the more balls are needed, hence the more bits are needed to encode and enumerate the balls. Each real-valued vector $\boldsymbol{x}$ can be encoded as the number of the ball which it falls into.

where $\boldsymbol{w}_i \sim \mathcal{N}(\boldsymbol{0}, \epsilon^2 \boldsymbol{I}/D)$ is a Gaussian noise independent of $\boldsymbol{x}_i$. Then the covariance of $\hat{\boldsymbol{x}}_i$ is given by

$$\hat{\boldsymbol{\Sigma}} = \mathbb{E}\left[\hat{\boldsymbol{x}}_i \hat{\boldsymbol{x}}_i^\top\right] = \frac{\epsilon^2}{D}\boldsymbol{I} + \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top. \tag{4.1.26}$$

Note that the volume of the region spanned by the vectors $\boldsymbol{x}_i$ is proportional to the square root of the determinant of the covariance matrix

$$\text{volume}(\hat{\boldsymbol{x}}_i) \propto \sqrt{\det\left(\hat{\boldsymbol{\Sigma}}\right)} = \sqrt{\det\left(\frac{\epsilon^2}{D}\boldsymbol{I} + \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top\right)}. \tag{4.1.27}$$

The volume spanned by each random vector $\boldsymbol{w}_i$ is proportional to

$$\text{volume}(\boldsymbol{w}_i) \propto \sqrt{\det\left(\frac{\epsilon^2}{D}\boldsymbol{I}\right)}. \tag{4.1.28}$$

To encode vectors that fall into the region spanned by $\hat{\boldsymbol{x}}_i$, we can cover the region with non-overlapping balls of radius ϵ, as illustrated in Figure 4.5. When the volume of the region spanned by $\hat{\boldsymbol{x}}_i$ is significantly larger than the volume of the ϵ-ball, the total number of balls that we need to cover the region is approximately equal to the ratio of the two volumes:

$$\#\,\epsilon\text{-balls} \approx \frac{\text{volume}(\hat{\boldsymbol{x}}_i)}{\text{volume}(\boldsymbol{w}_i)} = \sqrt{\det\left(\boldsymbol{I} + \frac{D}{N\epsilon^2}\boldsymbol{X}\boldsymbol{X}^\top\right)}. \tag{4.1.29}$$

If we use binary numbers to label all the ϵ-balls in the region of interest, the total number of binary bits needed is thus

$$\mathcal{R}_\epsilon(\boldsymbol{X}) \approx \log_2(\#\,\epsilon\text{-balls}) \approx R_\epsilon(\boldsymbol{X}) \doteq \frac{1}{2}\log\det\left(\boldsymbol{I} + \frac{D}{N\epsilon^2}\boldsymbol{X}\boldsymbol{X}^\top\right). \tag{4.1.30}$$

Notice that from Example 4.4 one can show that the right-hand side of this formula is the same as the rate distortion (up to precision ϵ) of a Gaussian source with covariance matrix $\frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top + (\epsilon^2/D)\boldsymbol{I}$ (e.g., the empirical covariance matrix of the data plus a noise term). In this way, a geometric argument about the volume of the enclosed vectors and sphere packing translates into an information-theoretic argument about the rate distortion of a perturbed Gaussian source.

Example 4.6. Figure 4.5 shows an example of a 2D distribution with an ellipsoidal support – approximating the support of a 2D Gaussian distribution. The region is covered by small balls of size ϵ. All the balls are numbered from 1 to say n. Then given any vector $\boldsymbol{x}$ in this region, we only need to determine to which ϵ-ball center it is the closest, denoted as $\text{ball}_\epsilon(\boldsymbol{x})$. To remember $\boldsymbol{x}$, we only need to remember the number of this ball, which takes $\log(n)$ bits to store. If we need to decode $\boldsymbol{x}$ from this number, we simply take $\hat{\boldsymbol{x}}$ as the center of the ball. This leads to an explicit encoding and decoding scheme:

$$\boldsymbol{x} \longrightarrow \text{ball}_\epsilon(\boldsymbol{x}) \longrightarrow \hat{\boldsymbol{x}} = \text{center of ball}_\epsilon(\boldsymbol{x}). \tag{4.1.31}$$

One may refer to these ball centers as "codes" of a code book or a dictionary for the encoding scheme. It is easy to see that the accuracy of this (lossy) encoding-decoding scheme is about the radius of the ball ϵ. Clearly $\mathcal{R}_\epsilon(\boldsymbol{Z})$ is the average number of bits required to encode the ball number of each vector $\boldsymbol{z}$ with this coding scheme, and hence can be called the *coding rate* associated with this scheme. ■

From the above derivation, we know that the coding rate $\mathcal{R}_\epsilon(\boldsymbol{X})$ is (approximately) achievable with an explicit encoding (and decoding) scheme. It has two interesting properties:

- First, one may notice that $R_\epsilon(\boldsymbol{X})$ closely resembles the rate distortion function of a Gaussian source [CT91] (see Example 4.4). Indeed, when ϵ is small, the above expression is a close approximation to the rate distortion of a Gaussian source, as pointed out by [MDH+07a].

- Second, the same closed-form coding rate $R_\epsilon(\boldsymbol{X})$ can be derived as an approximation of $\mathcal{R}_\epsilon(\boldsymbol{X})$ if the data $\boldsymbol{X}$ are assumed to be from a linear subspace. This can be shown by properly quantifying the singular value decomposition (SVD) of $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ and constructing a lossy coding scheme for vectors in the subspace spanned by $\boldsymbol{U}$ [MDH+07a].

In our context, the closed-form expression $R_\epsilon(\boldsymbol{X})$ is rather fundamental: it is the coding rate associated with an explicit and natural lossy coding scheme for data drawn from either a Gaussian distribution or a linear subspace. As we will see in the next chapter, this formula plays an important role in understanding the architecture of deep neural networks.

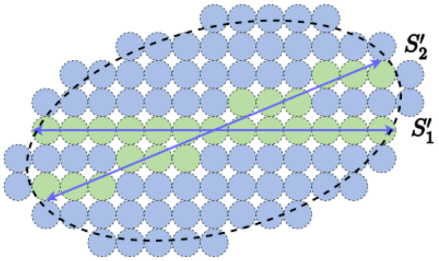


Figure 4.6: Comparison of two lossy coding schemes for data that are distributed around two subspaces. One is to pack (blue) ϵ-balls for the entire space spanned by the two subspaces; the other is to pack balls only in a tabular neighborhood around the two subspaces. The latter obviously has a much smaller code book and results in a much lower coding rate for samples on the subspaces.

4.1.4 Clustering a Mixture of Low-Dimensional Gaussians

As we have discussed before, the given dataset $\boldsymbol{X}$ often has low-dimensional intrinsic structures. Hence, encoding it as a general Gaussian would be very redundant. If we can identify those intrinsic structures in $\boldsymbol{X}$, we could design much better coding schemes that give much lower coding rates. Or equivalently, the codes used to encode such $\boldsymbol{X}$ can be compressed. We will see that compression gives a unifying computable way to identify such structures. In this section, we demonstrate this important idea with the most basic family of low-dimensional structures: a mixture of (low-dimensional) Gaussians or subspaces.

Example 4.7. Figure 4.6 shows an example in which the data $\boldsymbol{X}$ are distributed around two subspaces (or low-dimensional Gaussians). If they are viewed and coded together as one single Gaussian, the associated discrete (lossy) code book, represented by all the blue balls, is obviously very redundant. We can try to identify the locations of the two subspaces, denoted by S_1 and S_2, and design a code book that only covers the two subspaces, i.e., the green balls. If we can correctly partition samples in the data $\boldsymbol{X}$ into the two subspaces: $\boldsymbol{X} = [\boldsymbol{X}_1, \boldsymbol{X}_2]\boldsymbol{\Pi}$ with $\boldsymbol{X}_1 \in S_1$ and $\boldsymbol{X}_2 \in S_2$, where $\boldsymbol{\Pi}$ denotes a permutation matrix, then the resulting coding rate for the data will be much lower. This gives a more parsimonious, hence more desirable, representation of the data. ■

So, more generally speaking, if the data are drawn from any mixture of subspaces or low-dimensional Gaussians, it would be desirable to identify those components and encode the data based on the intrinsic dimensions of those components. It turns out that we do not lose much generality by assuming that the data are drawn from a mixture of low-dimensional Gaussians. This is because a mixture of Gaussians can closely approximate most general distributions [BDS16].

The clustering problem. Now for this specific family of distributions, how can we effectively and efficiently identify those low-dimensional components from a set of samples

$$\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N], \tag{4.1.32}$$

drawn from them? In other words, given the whole data set $\boldsymbol{X}$, we want to partition, or cluster, it into multiple, say K, subsets:

$$\boldsymbol{X}\boldsymbol{\Pi} = [\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_K], \tag{4.1.33}$$

where each subset consists of samples drawn from only one low-dimensional Gaussian or subspace and $\boldsymbol{\Pi}$ is a permutation matrix to indicate membership of the partition. Note that, depending on the situation, the partition could be either deterministic or probabilistic. As shown in [MDH+07b], for a mixture of Gaussians, probabilistic partition does not lead to a lower coding rate. So for simplicity, we here consider a deterministic partition only.

Clustering via lossy compression. The main difficulty in solving the above clustering problem is that we normally do not know the number of clusters K, nor do we know the dimension of each component. There has been a long history for the study of this clustering problem. The textbook [VMS16] gives a systematic and comprehensive coverage of different approaches to this problem. To find an effective approach to this problem, we first need to understand and clarify why we want to cluster. In other words, what exactly do we gain from clustering the data, compared with not to? How do we measure the gain? From the perspective of data compression, a correct clustering should lead to a more efficient encoding (and decoding) scheme.

For any given data set $\boldsymbol{X}$, there are already two obvious encoding schemes as the baseline. They represent two extreme ways to encode the data:

- Simply view all the samples together drawn as from one single Gaussian. The associated coding rate is, as derived before, given by:

$$\mathcal{R}_\epsilon(\boldsymbol{X}) \approx R_\epsilon(\boldsymbol{X}) = \frac{1}{2}\log\det\left(\boldsymbol{I} + \frac{D}{N\epsilon^2}\boldsymbol{X}\boldsymbol{X}^\top\right). \tag{4.1.34}$$

- Simply memorize all the samples separately by assigning a different number to each sample. The coding rate would be:

$$\mathcal{R}_0(\boldsymbol{X}) = \log(N). \tag{4.1.35}$$

Note that either coding scheme can become the "optimal" solution for certain (extreme) choice of the quantization error ϵ:

1. *Lazy Regime*: If we choose ϵ to be extremely large, all samples in $\boldsymbol{X}$ can be covered by a single ball. The rate is $\lim_{\epsilon\to\infty}\mathcal{R}_\epsilon \to \frac{1}{2}\log\det(\boldsymbol{I}) = 0$.
2. *Memorization Regime*: If ϵ is extremely small, every sample in $\boldsymbol{X}$ is covered by a different ϵ-ball, hence the total is N. The rate is $\lim_{\epsilon\to 0}\mathcal{R}_\epsilon \to \log(N)$.

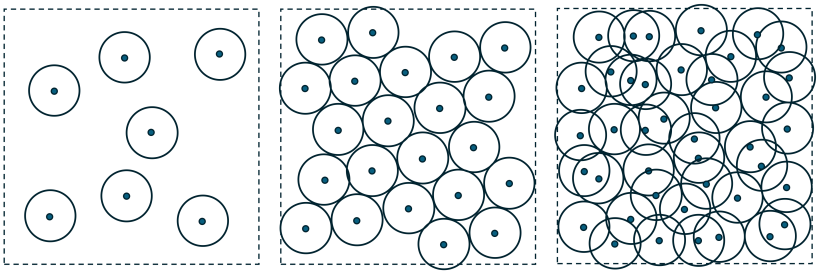


Figure 4.7: A number of random samples on a 2D plane. Consider an ϵ-disc assigned to each sample with the sample as its center. The density of the samples increases from left to right.

Note that the first scheme corresponds to the scenario when one does not care about anything interesting about the distribution at all. One does not want to spare any bit for anything informative. We call this the "lazy regime." The second scheme corresponds to the scenario when one wants to decode every sample with an extremely high precision. So one would better "memorize" every sample. We call this the "memorization regime."

Notice that these asymptotics in terms of ϵ are exactly those discussed when introducing the lossy coding rate as a quantity that realistically-trained diffusion models approximate, as in the end of Section 3.3. In the current situation, we *explicitly* code the distribution up to precision ϵ; in the case of diffusion models, we *implicitly* code the distribution. Certain properties of the scheme such as memorization and generalization manifest similarly in both cases, and indeed one can study one case to understand the other.

Example 4.8. To see when the memorization regime is preferred or not, let us consider a number, say N, of samples randomly distributed in a unit area on a 2D plane.[14] Imagine we try to design a lossy coding scheme with a fixed quantization error ϵ. This is equivalent to putting an ϵ-disc around each sample, as shown in Figure 4.7. When N is small, the chance that all the discs overlap with each other is zero. A codebook of size N is necessary and optimal in this case. When N or the density reaches a certain critical value N_c, with high probability all the discs start to overlap and connect into one cluster that covers the whole plane—this phenomenon is known as continuum "percolation" [Gil61; MM12]. When N becomes larger than this value, the discs overlap heavily. The number N of discs becomes very redundant because we only want to encode points on the plane up to the given precision ϵ. The number of discs needed to cover all the samples is much less than N.[15] ■

Both the lazy and memorization regimes are somewhat trivial and perhaps are of little theoretical or practical interest. Either scheme would be far from

[14] Say the points are drawn by a Poisson process with density N points per unit area.
[15] In fact, there are efficient algorithms to find such a covering [BBF+01].

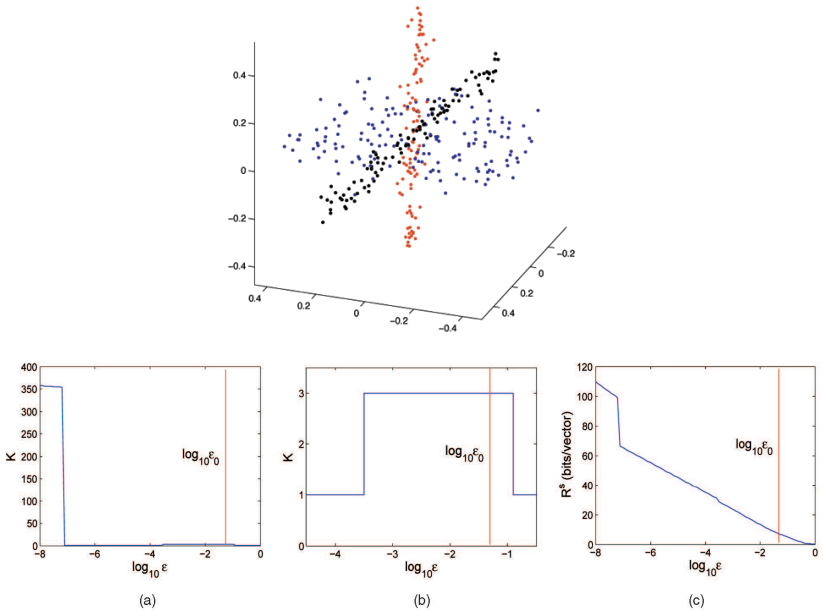


Figure 4.8: Top: 358 noisy samples drawn from two lines and one plane in $\mathbb{R}^3$. Bottom: the effect of varying ϵ on the clustering result and the coding rate. The red line marks the variance ϵ_0 of the Gaussian noise added to the samples. Note that the bottom left and middle plots show the same quantity with different axis locations and scales.

optimal when used to encode a large number of samples drawn from a distribution that has a *compact and low-dimensional support*. The interesting regime exists in between these two.

Example 4.9. Figure 4.8 shows an example with noisy samples drawn from two lines and one plane in $\mathbb{R}^3$. As we notice from the plot (c) on the right, the optimal coding rate decreases monotonically as we increase ϵ, as anticipated from the property of the rate distortion function. The plots (a) and (b) show, when varying ϵ from very small (near zero) to very large (towards infinite), the optimal number of clusters when the coding rate is minimal. We can clearly see the lazy regime and the memorization regime on the two ends of the plots. But one can also notice in plot (b), when the quantization error ϵ is chosen to be around the level of the true noise variance ϵ_0, the optimal number of clusters is the "correct" number three that represents two lines and one plane. We informally refer to this middle regime as the "generalization regime". Notice that a sharp phase transition takes place between these regimes.[16] ■

From the above discussion and examples, we see that, when the quantiza-

[16]So far, to our best knowledge, there is no rigorous theoretical justification for these phase transition behaviors.

tion error relative to the sample density[17] is in a proper range, minimizing the lossy coding rate would allow us to uncover the underlying (low-dimensional) distribution of the sampled data. *Hence, quantization, started as a choice of practicality, seems to be becoming necessary for learning a continuous distribution from its empirical distribution with finite samples.* Although a rigorous theory for explaining this phenomenon remains elusive, here, for learning purposes, we care about how to exploit the phenomenon to design algorithms that can find the correct distribution.

Let us use the simple example shown in Figure 4.6 to illustrate the basic ideas. If one can partition all samples in $\boldsymbol{X}$ into two clusters in $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, with N_1 and N_2 samples respectively, then the associated coding rate would be[18]

$$R_\epsilon^c(\boldsymbol{X} \mid \boldsymbol{\Pi}) = \frac{N_1}{N} R_\epsilon(\boldsymbol{X}_1) + \frac{N_2}{N} R_\epsilon(\boldsymbol{X}_2), \tag{4.1.36}$$

where we use $\boldsymbol{\Pi}$ to indicate membership of the partition. If the partition respects the low-dimensional structures of the distribution, in this case $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ belonging to the two subspaces respectively, then the resulting coding rate should be significantly smaller than the above two basic schemes:

$$R_\epsilon^c(\boldsymbol{X} \mid \boldsymbol{\Pi}) \ll R_\epsilon(\boldsymbol{X}), \quad R_\epsilon^c(\boldsymbol{X} \mid \boldsymbol{\Pi}) \ll R_0(\boldsymbol{X}). \tag{4.1.37}$$

In general, we can cast the clustering problem into an optimization problem that minimizes the coding rate:

$$\min_{\boldsymbol{\Pi}} \left\{ R_\epsilon^c(\boldsymbol{X} \mid \boldsymbol{\Pi}) \doteq \sum_{k=1}^{K} \frac{N_k}{N} R_\epsilon(\boldsymbol{X}_k) \right\}. \tag{4.1.38}$$

Optimization strategies to cluster. The remaining question is how we optimize the above coding rate objective to find the optimal clusters. There are three natural approaches to this objective:

1. We may start with the whole set $\boldsymbol{X}$ as a single cluster (i.e. the lazy regime) and then search (say randomly) to partition it so that it would lead to a smaller coding rate.

2. Inversely, we may start with each sample $\boldsymbol{x}_i$ as its own cluster (i.e. the memorization regime) and search to merge clusters that would result in a smaller coding rate.

3. Alternatively, if we could represent (or approximate) the membership $\boldsymbol{\Pi}$ as some continuous parameters, we may use optimization methods such as gradient descent (GD).

[17] or the sample density relative to the quantization error

[18] We here ignore some overhead bits needed to encode the membership for each sample, say via the Huffman coding.

The first approach is not so appealing computationally as the number of possible partitions that one needs to try is exponential in the number of samples. For example, the number of partitions of $\boldsymbol{X}$ into two subsets of equal size is $\binom{N}{N/2}$ which explodes as N becomes large. We will explore the third approach in the next Chapter 5. There, we will see how the role of deep neural networks, transformers in particular, is connected with the coding rate objective.

The second approach was originally suggested in the work of [MDH+07b]. It demonstrates the benefit of being able to evaluate the coding rate efficiently (say with an analytical form). With it, the (low-dimensional) clusters of the data can be found rather efficiently and effectively via the principle of minimizing coding length (MCL). Note that for a cluster $\boldsymbol{X}_k$ with N_k samples, the length of binary bits needed to encode all the samples in $\boldsymbol{X}_k$ is given by:[19]

$$L(\boldsymbol{X}_k) = N_k R_\epsilon(\boldsymbol{X}_k). \tag{4.1.39}$$

If we have two clusters $\boldsymbol{X}_k$ and $\boldsymbol{X}_l$, if we want to code the samples as two separate clusters, the length of binary bits needed is

$$L^c(\boldsymbol{X}_k, \boldsymbol{X}_l) = N_k R_\epsilon(\boldsymbol{X}_k) + N_l R_\epsilon(\boldsymbol{X}_l) - N_k \log \frac{N_k}{N_k + N_l} - N_l \log \frac{N_l}{N_k + N_l}.$$

The last two terms are the number of bits needed to encode the memberships of samples according to the Huffman code.

Then, given any two separate clusters $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, we can decide whether to merge them or not based on the difference between the two coding lengths:

$$L(\boldsymbol{X}_k \cup \boldsymbol{X}_l) - L^c(\boldsymbol{X}_k, \boldsymbol{X}_l) \tag{4.1.40}$$

is positive or negative and $\boldsymbol{X}_k \cup \boldsymbol{X}_l$ denotes the union of the sets of samples in $\boldsymbol{X}_k$ and $\boldsymbol{X}_l$. If it is positive , it means the coding length would become smaller if we merge the two clusters into one. This simple fact leads to the following clustering algorithm proposed by [MDH+07b]:

Note that this algorithm is tractable as the total number of (pairwise) comparisons and merges is about $O(N^2 \log N)$. However, due to its greedy nature, there is no theoretical guarantee that the process will converge to the globally optimal clustering solution. Nevertheless, as reported in [MDH+07b], in practice, this seemingly simple algorithm works extremely well. The clustering results plotted in Figure 4.8 were actually computed by this algorithm.

Example 4.10 (Image Segmentation). The above measure of coding length and the associated clustering algorithm assume the data distribution is a mixture of (low-dimensional) Gaussians. Although this seems somewhat idealistic, the measure and algorithm can already be very useful and even powerful in scenarios when the model is (approximately) valid.

[19] In fact, a more accurate estimate of the coding length is $L(\boldsymbol{X}_k) = (N_k + D)R_\epsilon(\boldsymbol{X}_k)$ where the extra bits are used to encode the basis of the subspace [MDH+07b]. Here we omit this overhead for simplicity.

Algorithm 4.1 Pairwise Steepest Descent of Coding Length

Input: N data points $\{\boldsymbol{x}_i\}_{i=1}^N$
Output: A set $\mathcal{C}$ of clusters

procedure PairwiseSteepestDescentOfCodingLength($\{\boldsymbol{x}_i\}_{i=1}^N$)
  `# Initialize` N `clusters` $\boldsymbol{X}_k$ `with one element each`
  $\mathcal{C} \leftarrow \{\{\boldsymbol{x}_i\}\}_{i=1}^N$
  while $|\mathcal{C}| > 1$ **do**
    `# If no bits are saved by any merging`
    if $\min_{\boldsymbol{X}_k, \boldsymbol{X}_l \in \mathcal{C}} [L(\boldsymbol{X}_k \cup \boldsymbol{X}_l) - L^c(\boldsymbol{X}_k, \boldsymbol{X}_l)] \geq 0$ **then**
      `# Early return` $\mathcal{C}$ `and exit`
      return $\mathcal{C}$
    else
      `# Merge clusters which save the most bits`
      $\boldsymbol{X}_{k^*}, \boldsymbol{X}_{l^*} \leftarrow \arg\min_{\boldsymbol{X}_k, \boldsymbol{X}_l \in \mathcal{C}} [L(\boldsymbol{X}_k \cup \boldsymbol{X}_l) - L^c(\boldsymbol{X}_k, \boldsymbol{X}_l)]$
      `# Remove unmerged clusters and add back the merged one`
      $\mathcal{C} \leftarrow [\mathcal{C} \setminus \{\boldsymbol{X}_{k^*}, \boldsymbol{X}_{l^*}\}] \cup \{\boldsymbol{X}_{k^*} \cup \boldsymbol{X}_{l^*}\}$
    end if
  end while
  `# If all merges yield savings, return one cluster`
  return $\mathcal{C}$
end procedure

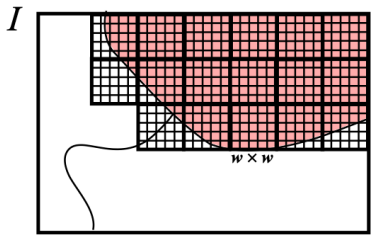


Figure 4.9: Image patches with a size of $w \times w$ pixels.

For example, a natural image typically consists of multiple regions with nearly homogeneous textures. If we take many small windows from each region, they should resemble samples drawn from a (low-dimensional) Gaussian, as illustrated in Figure 4.9. Figure 4.10 shows the results of image segmentation based on applying the above clustering algorithm to the image patches directly. More technical details regarding customizing the algorithm to the image segmentation problem can be found in [MRY+11]. ■

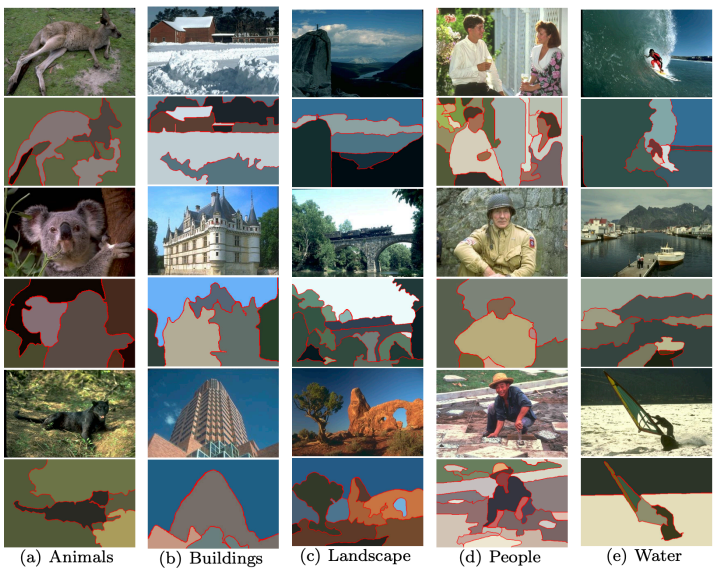


Figure 4.10: Segmentation results based on the clustering algorithm applied to the image patches.

4.2 Transformation via Maximizing Information Gain

So far in this chapter, we have discussed how to identify a distribution with low-dimensional structures through the principle of compression. As we have seen from the previous two sections, computational compression can be realized through either the denoising operation or clustering. Figure 4.11 illustrates this concept with our favorite example.

It is important to note that the clustering algorithm and the coding rate formulas developed in the previous section assumed that the data distribution is *already* (approximately) a mixture of low-dimensional Gaussians. In practice, the raw data—images, audio, text—rarely satisfy this assumption directly: their intrinsic structures are typically *nonlinear*. The purpose of this section is to develop a framework for *transforming* data so that their representations do take the form of a mixture of incoherent low-dimensional subspaces, at which point the tools from the preceding section become directly applicable. That is, the preceding section answers "how to compress and cluster data that are already Gaussian," while this section answers "how to transform data so that they become Gaussian." As we will see in Chapter 5, deep networks are precisely the computational mechanism for realizing such transformations, with the network's architecture derived from optimizing the coding rate objective developed here.

Indeed, the ultimate goal for identifying a data distribution is to use it to facilitate certain subsequent tasks such as segmentation, classification, or generation (of images). Hence, how the resulting distribution is "represented" matters tremendously with respect to how information related to these subsequent tasks can be efficiently and effectively retrieved and utilized. This naturally raises a fundamental question: *what makes a representation truly "good" for downstream use?* In the following, we will explore the essential properties that a meaningful and useful representation should possess, and how these properties can be explicitly characterized and pursued via maximizing information gain.

How to measure the goodness of representations. One may view a given dataset as samples of a random vector $\boldsymbol{x}$ with a certain distribution in a high-dimensional space, say $\mathbb{R}^D$. Typically, the distribution of $\boldsymbol{x}$ has a much lower intrinsic dimension than the ambient space. Generally speaking, *learning a representation* refers to learning a continuous mapping, say $f(\cdot)$, that transforms $\boldsymbol{x}$ to a so-called *feature vector* $\boldsymbol{z}$ in another (typically lower-dimensional) space, say $\mathbb{R}^d$, where $d < D$. It is hopeful that through such a mapping

$$\boldsymbol{x} \in \mathbb{R}^D \xrightarrow{f(\boldsymbol{x})} \boldsymbol{z} \in \mathbb{R}^d, \tag{4.2.1}$$

the low-dimensional intrinsic structures of $\boldsymbol{x}$ are identified and represented by $\boldsymbol{z}$ in a more compact and structured way so as to facilitate subsequent tasks such as classification or generation. The feature $\boldsymbol{z}$ can be viewed as a (learned)

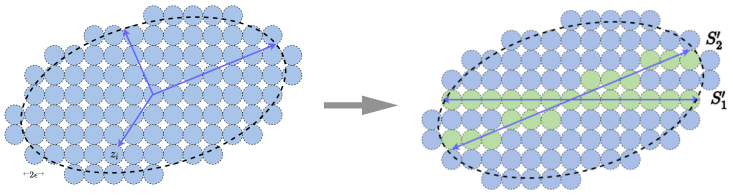


Figure 4.11: Identifying a low-dimensional distribution with two subspaces (left) via denoising or clustering, starting from a generic random Gaussian distribution (right).

compact code for the original data $\boldsymbol{x}$, so the mapping f is also called an *encoder*. The fundamental question of representation learning is

What is a principled and effective measure for the goodness of representations?

Conceptually, the quality of a representation $\boldsymbol{z}$ depends on how well it identifies the most relevant and sufficient information of $\boldsymbol{x}$ for subsequent tasks and how efficiently it represents this information. For a long time, it was believed and argued that the "sufficiency" or "goodness" of a learned feature representation should be defined in terms of a specific task. For example, $\boldsymbol{z}$ just needs to be sufficient for predicting the class label $\boldsymbol{y}$ in a classification problem. Below, let us start with the classic problem of image classification and argue why such a notion of a task-specific "representation" is limited and needs to be generalized.

4.2.1 Representation Learning from Classified Data

Suppose that $\boldsymbol{x} \in \mathbb{R}^D$ is a random vector drawn from a mixture of K (component) distributions $\mathcal{D} = \{\mathcal{D}_k\}_{k=1}^K$. Given a finite set of i.i.d. samples $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{D \times N}$ of the random vector $\boldsymbol{x}$, we *seek a good representation* through a continuous mapping $f(\boldsymbol{x}) : \mathbb{R}^D \to \mathbb{R}^d$ that captures intrinsic structures of $\boldsymbol{x}$ and best facilitates the subsequent classification task.[20] To ease the task of learning distribution $\mathcal{D}$, in the popular supervised classification setting, a true class label (or a code word for each class), usually represented by a one-hot vector $\boldsymbol{y}_i \in \mathbb{R}^K$, is given for each sample $\boldsymbol{x}_i$.

Encoding class information via cross entropy. Extensive studies have shown that for many practical datasets (e.g., images, audio, and natural languages), the (encoding) mapping from the data $\boldsymbol{x}$ to its class label $\boldsymbol{y}$ can be

[20] Classification is the domain where deep learning demonstrated the initial success, sparking the explosive interest in deep networks. Although our study focuses on classification, we believe the ideas and principles can be naturally generalized to other settings, such as regression.

effectively modeled by training a deep network,[21] here denoted as

$$f(\boldsymbol{x}, \theta) : \boldsymbol{x} \mapsto \boldsymbol{y}$$

with network parameters $\theta \in \Theta$, where Θ denotes the parameter space. For the output $f(\boldsymbol{x}, \theta)$ to match well with the label $\boldsymbol{y}$, we like to minimize the *cross-entropy loss* over a training set $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$:[22]

$$\min_{\theta \in \Theta} \mathbb{E}[\mathrm{CE}(\boldsymbol{y} \parallel f(\boldsymbol{x}, \theta))] \approx -\frac{1}{N} \sum_{i=1}^{N} \langle \boldsymbol{y}_i, \log\left(f(\boldsymbol{x}_i, \theta)\right) \rangle. \tag{4.2.2}$$

The optimal network parameters θ are typically found by optimizing the above objective through an efficient gradient descent scheme, with gradients computed via back propagation, as described in Section A.2.3 of Chapter A.

Despite its effectiveness and enormous popularity, there are two serious limitations with this approach: (1) It aims only to predict the labels $\boldsymbol{y}$ even if they might be mislabeled. Empirical studies show that deep networks, used as a "black box," can even fit random labels [ZBH+17]. (2) With such an end-to-end data fitting, despite plenty of empirical efforts in trying to interpret the so-learned features, it is not clear to what extent the intermediate features learned by the network capture the intrinsic structures of the data that make meaningful classification possible in the first place. The precise geometric and statistical properties of the learned features are also often obscured, which leads to the lack of interpretability and subsequent performance guarantees (e.g., generalizability, transferability, and robustness, etc.) in deep learning. Therefore, *one of the goals of this section is to address such limitations by reformulating the objective towards learning explicitly meaningful and useful representations for the data $\boldsymbol{x}$, not limited to classification.*

Minimal discriminative features via information bottleneck. One popular approach to interpret the role of deep networks is to view outputs of intermediate layers of the network as selecting certain latent features $\boldsymbol{z} = f(\boldsymbol{x}, \theta) \in \mathbb{R}^d$ of the data that are discriminative among multiple classes. Learned representations $\boldsymbol{z}$ then facilitate the subsequent classification task for predicting the class label $\boldsymbol{y}$ by optimizing a classifier $g(\boldsymbol{z})$:

$$\boldsymbol{x} \xrightarrow{\quad f(\boldsymbol{x}, \theta) \quad} \boldsymbol{z} \xrightarrow{\quad g(\boldsymbol{z}) \quad} \boldsymbol{y}. \tag{4.2.3}$$

We know from (4.1.13) that *the mutual information* between two random variables, say $\boldsymbol{x}, \boldsymbol{z}$, is defined to be

$$I(\boldsymbol{x}; \boldsymbol{z}) = H(\boldsymbol{x}) - H(\boldsymbol{x} \mid \boldsymbol{z}), \tag{4.2.4}$$

[21] Here let us not worry about yet which network we should use here and why. The purpose here is to consider any empirically tested deep network. We will leave the justification of the network architectures to the next chapter.

[22] Here we interpret $\boldsymbol{y}$ as a particular *one-hot* probability distribution over $[K] = \{1, 2, \ldots, K\}$, and $f(\boldsymbol{x}, \theta) \in \Delta([K]) \subseteq \mathbb{R}^K$ to also be a probability distribution over the same set, which can therefore be compared via the cross-entropy despite being computationally represented as vectors in $\mathbb{R}^K$.

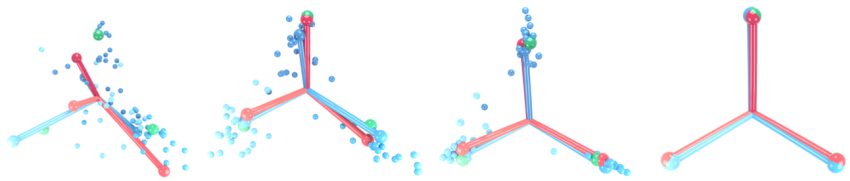


Figure 4.12: Evolution of penultimate layer outputs of a VGG13 neural network when trained on the CIFAR-10 dataset with 3 randomly selected classes. Figure from [PHD20].

where $H(\boldsymbol{x}|\boldsymbol{z})$ is the conditional entropy of $\boldsymbol{x}$ given $\boldsymbol{z}$. The mutual information is also known as *the information gain*: It measures how much the entropy of the random variable $\boldsymbol{x}$ can be reduced once $\boldsymbol{z}$ is given. Or equivalently, it measures how much information $\boldsymbol{z}$ contains about $\boldsymbol{x}$. The *information bottleneck* (IB) formulation [TZ15] further hypothesizes that the role of the network is to learn $\boldsymbol{z}$ as the minimal sufficient statistics for predicting $\boldsymbol{y}$. Formally, it seeks to maximize the mutual information $I(\boldsymbol{z}, \boldsymbol{y})$ between $\boldsymbol{z}$ and $\boldsymbol{y}$ while minimizing the mutual information $I(\boldsymbol{x}, \boldsymbol{z})$ between $\boldsymbol{x}$ and $\boldsymbol{z}$:

$$\max_{\theta \in \Theta} \text{IB}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \doteq I(\boldsymbol{z}; \boldsymbol{y}) - \beta I(\boldsymbol{x}; \boldsymbol{z}) \quad \text{s.t. } \boldsymbol{z} = f(\boldsymbol{x}, \theta), \tag{4.2.5}$$

where $\beta > 0$.

Given one can overcome some caveats associated with this framework [KTV18], such as how to accurately evaluate mutual information with finite samples of degenerate distributions, this framework can be helpful in explaining certain behaviors of deep networks. For example, recent work [PHD20] indeed shows that the representations learned via the cross-entropy loss (4.2.2) exhibit a *neural collapse* phenomenon. That is, features of each class are mapped to a one-dimensional vector whereas all other information of the class is suppressed, as illustrated in Figure 4.12.[23]

Remark 4.4 (Neural Collapse). Neural collapse refers to a phenomenon observed in deep neural networks trained for classification, where the learned feature representations and classifier weights exhibit highly symmetric and structured behavior during the terminal phase of training [PHD20; YWZ+22; ZDZ+21]. Specifically, within each class, features collapse to their class mean, and across classes, these means become maximally separated, forming a simplex equiangular configuration. The linear classifier aligns with the class mean up to rescaling. Additionally, the last-layer classifier converges to choosing whichever class has the nearest train class mean. Neural collapse reveals deep connections between optimization dynamics, generalization, and geometric structures arising in supervised learning.

[23] We will observe this phenomenon in experiments with real imagery data later in Section 4.3.1.

From the above example of classification, we see that the so-learned representation $\boldsymbol{z}$ of $\boldsymbol{x}$ gives a very simple encoder that essentially maps each class of data to only one code word: the one-hot vector representing each class. From the lossy compression perspective, such an encoder is too lossy to preserve information in the data distribution. Other information, such as that useful for tasks such as image segmentation, completion, or generation, is severely lost in such a supervised learning process.

To remedy this situation, we want to learn a different encoding scheme such that the resulting feature representation $\boldsymbol{z}$ can capture as much information about the data distribution as possible, not limited to that useful for classification alone. More precisely, we want the representation $\boldsymbol{z} = f(\boldsymbol{x}, \theta)$ to be such that there exists a decoder $g(\boldsymbol{z}, \eta)$ so that:

$$\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2 \leq \epsilon \quad \text{s.t.} \quad \hat{\boldsymbol{x}} = g(f(\boldsymbol{x})). \tag{4.2.6}$$

We will formally introduce such an encoding-decoding scheme as an "autoencoder" and such a representation $\boldsymbol{z}$ as a "consistent" representation in Chapter 6. For now, the reader may simply view this as a generalization to the quantization code $\hat{\boldsymbol{x}}$ introduced in the preceding section for the rate distortion (in the native data $\boldsymbol{x}$ domain).

In the formulation of information bottleneck (4.2.5), it does not require the learned feature representation $\boldsymbol{z}$ to be consistent. Hence, at least it should be fixed with an objective for a (consistent) representation learning (RL):

$$\begin{aligned} &\max_{\theta,\eta} \ \mathrm{RL}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \doteq I(\boldsymbol{z}; \boldsymbol{y}) - \beta I(\boldsymbol{x}; \boldsymbol{z}) \\ &\text{s.t.} \quad \boldsymbol{z} = f(\boldsymbol{x}, \theta), \quad \|\boldsymbol{x} - g(\boldsymbol{z}, \eta)\|_2 \leq \epsilon. \end{aligned} \tag{4.2.7}$$

Solving the above problem in the most general setting is a very daunting task. In general, we do not even know what classes of encoders or decoders could ensure the consistency[24] Also, since we do not know the distribution of the representation $\boldsymbol{z}$, computing the mutual information $I(\boldsymbol{z}; \boldsymbol{y})$ can be very challenging, if not impossible.

Before we study how to approach this problem in general settings (in later Chapters), for the rest of this chapter, we will demonstrate how to solve it, in some limited way, for the special setting when the data distribution consists of clear (low-dimensional) classes.

4.2.2 Linear Discriminative Representations

Whether the given data $\boldsymbol{X}$ of a mixed distribution $\mathcal{D}$ can be effectively classified or clustered depends on how separable (or discriminative) the component distributions $\mathcal{D}_k$ are (or can be made). One popular working assumption is that the distribution of each class has relatively *low-dimensional* intrinsic structures. Hence we may assume that the distribution $\mathcal{D}_k$ of each class has a support on a

[24] In practice, people tend to learn them, separately from the above objective, through trial and error, as we will discuss more in Chapter 6.

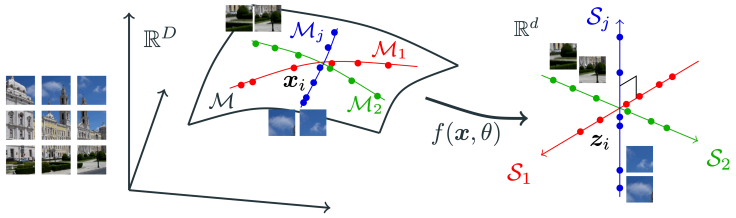


Figure 4.13: The distribution $\mathcal{D}$ of high-dimensional data $\boldsymbol{x} \in \mathbb{R}^D$ is supported on a manifold $\mathcal{M}$ and its classes on low-dimensional submanifolds $\mathcal{M}_k$. We aim to learn a mapping $f(\boldsymbol{x}, \theta)$ parameterized by θ such that $\boldsymbol{z}_i = f(\boldsymbol{x}_i, \theta)$ lie on a union of maximally uncorrelated subspaces $\{\mathcal{S}_k\}$.

low-dimensional submanifold, say $\mathcal{M}_k$ with dimension $d_k \ll D$, and the distribution $\mathcal{D}$ of $\boldsymbol{x}$ is supported on the mixture of those submanifolds, $\mathcal{M} = \cup_{k=1}^{K} \mathcal{M}_k$, in the high-dimensional ambient space $\mathbb{R}^D$, as illustrated before in Figure 4.1.

Not only do we need to identify the low-dimensional distribution, but we also want to represent the distribution in a form that best facilitates subsequent tasks such as classification, clustering, and conditioned generation (as we will see in the future). To do so, we require our learned feature representations to have the following properties:

1. *Within-Class Compressible:* Features of samples from the same class should be strongly *correlated* in the sense that they belong to a low-dimensional linear subspace.

2. *Between-Class Discriminative:* Features of samples from different classes should be highly *uncorrelated* and belong to different incoherent low-dimensional linear subspaces.

3. *Maximally Diverse Representation:* Dimension (or variance) of the features of each class should be *as large as possible* as long as they are incoherent to the other classes.

We refer to such a representation the *linear discriminative representation* (LDR).

Notice that the first property aligns well with the objective of the classic *principal component analysis* (PCA) that we have discussed in Section 2.1.1. It essentially says that features of samples that belong to the same class[25] should become or be made highly correlated. In other words, feature representation should be *contractive* for samples that are similar.

The second property requires that for samples that belong to different classes or are believed to be dissimilar, their features should be made (geometrically) incoherent or (statistically) independent, as illustrated before in Figure 4.1. Figure 4.13 illustrates when the data distribution is actually a mixture of low-dimensional submanifolds. Through compression (denoising or clustering), we

[25] including augmented instances of the same samples

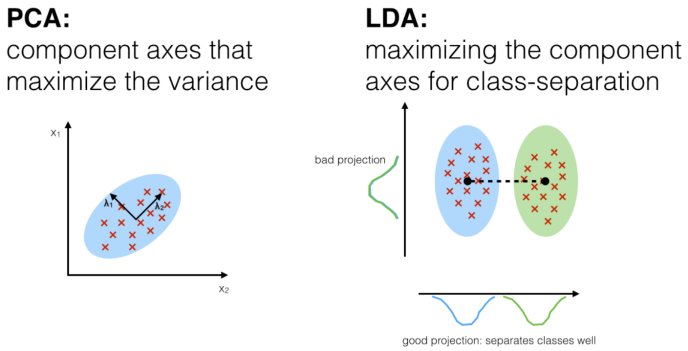


Figure 4.14: Comparison between PCA and LDA. Figures adopted from https://sebastianraschka.com/Articles/2014_python_lda.html.

first identify that the true data distribution (left). We then would like to transform the distribution so that the submanifolds eventually become mutually incoherent/independent linear subspaces (right).

Remark 4.5 (Linear Discriminative Analysis). One may notice that the second criterion resembles that of the classic *linear discriminant analysis* (LDA) [HTF09]. Linear discriminant analysis (LDA) [HTF09] is a supervised dimensionality reduction technique that aims to find a linear projection of data that maximizes class separability. Specifically, given labeled data, LDA seeks a linear transformation that projects high-dimensional inputs onto a lower-dimensional space where the classes are maximally separated. Note that PCA is an unsupervised method that projects data onto directions of maximum variance without considering class labels. While PCA focuses purely on preserving global variance structure, LDA explicitly exploits label information to enhance discriminative power; see the comparison in Figure 4.14. But LDA only performs a linear projection of the original data. As we will see LDR can be more general: it seeks a, probably nonlinear, transformation to make the resulting representation linear and incoherent.

Remark 4.6 (Contrastive Learning). One may notice that the conventional *contrastive learning* [HCL06; HFW+19; OLV18] aims to achieve similar goals for feature representation as the first two properties stipulate. For data that belong to k different classes, a randomly chosen pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is of high probability belonging to different classes if k is large.[26] Hence it is desirable that the representation $\boldsymbol{z}_i = f(\boldsymbol{x}_i, \boldsymbol{\theta})$ of a sample $\boldsymbol{x}_i$ should be highly incoherent to those $\boldsymbol{z}_j$ of other samples $\boldsymbol{x}_j$ whereas coherent to feature of its transformed version $\tau(\boldsymbol{x}_i)$, denoted as $\boldsymbol{z}(\tau(\boldsymbol{x}_i))$ for τ in certain augmentation set $\mathcal{T}$ in consideration. Hence it was proposed heuristically that, to promote discriminativeness of the

[26] For example, when $k \geq 100$, a random pair is of probability 99% belonging to different classes.

learned representation, one may seek to minimize the so-called *contrastive loss*:

$$\min_{\boldsymbol{\theta}} -\log \frac{\exp(\langle \boldsymbol{z}_i, \boldsymbol{z}(\tau(\boldsymbol{x}_i))\rangle)}{\sum_{j\neq i}\exp(\langle \boldsymbol{z}_i, \boldsymbol{z}_j\rangle)}, \tag{4.2.8}$$

which is small whenever the inner product $\langle \boldsymbol{z}_i, \boldsymbol{z}(\tau(\boldsymbol{x}_i))\rangle$ is large and $\langle \boldsymbol{z}_i, \boldsymbol{z}_j\rangle$ is small for $i \neq j$. Such contrastive loss, and many of its variants, has been widely used in popular (self-supervised) feature learning methods such as the DINO-type features [CMM+21; ODM+24] for visual data. As we will see, based on the lossy coding framework studied in the previous section, we will be able to arrive at a principled objective function that promotes the two properties and at the same time significantly simplifies the practice such as learning the DINO-type features (as we will feature in great detail in Chapter 8).

The third property is also important because we want the learned features to reveal all possible causes of why one class is different from all other classes. For example, to tell "apple" from "orange", we care not only about color but also shape and the leaves. Ideally, the dimension of each subspace $\{\mathcal{S}_k\}$ should be equal to that of the corresponding submanifold $\mathcal{M}_k$. This property will be important if we would like the map $f(\boldsymbol{x}, \theta)$ to be *invertible* for tasks such as image generation. For example, if we draw different sample points from the feature subspace for "apple", we should be able to decode them, via an inverse map $g(\boldsymbol{z}, \eta)$, to generate diverse images of apples. Such a pair of mutually inverse mappings f and g makes an autoencoding which we will study in great detail in Chapter 6. Notice that feature representations learned from minimizing the cross entropy (4.2.2) normally do not have this property, as they suffer the so-called *neural collapse* phenomenon discussed earlier (see Remark 4.4).

In general, although the intrinsic structures of each class/cluster may be low-dimensional, they are by no means simply linear (or Gaussian) in their original representation $\boldsymbol{x}$ and they need to be made linear first, through some nonlinear transformation.[27] Therefore, overall, we use the nonlinear transformation $f(\boldsymbol{x}, \theta)$ to seek a representation of the data such that the subspaces that represent all the classes are maximally incoherent linear subspaces. To be more precise, we want to learn a mapping $\boldsymbol{z} = f(\boldsymbol{x}, \theta)$ that maps each of the submanifolds $\mathcal{M}_k \subset \mathbb{R}^D$ (Figure 4.13 left) to a *linear* subspace $\mathcal{S}_k \subset \mathbb{R}^d$ (Figure 4.13 right). To some extent, the resulting multiple subspaces $\{\mathcal{S}_k\}$ can be viewed as discriminative *generalized principal components* [VMS16] or, if orthogonal, *independent components* [HO00a] of the resulting features $\boldsymbol{z}$ for the original data $\boldsymbol{x}$. As we will see in the next Chapter 5, deep networks precisely play the role of modeling and realizing this nonlinear transformation from the data distribution to linear discriminative representations.

4.2.3 The Principle of Maximal Coding Rate Reduction

Although the three properties—*between-class discriminative*, *within-class compressible*, and *maximally diverse representation*—for linear discriminative repre-

[27] We will discuss how this can be done explicitly in Chapter 6.

sentations (LDRs) are all highly desired properties of the learned representation $\boldsymbol{z}$, they are by no means easy to obtain: Are these properties compatible so that we can expect to achieve them all at once? If so, is there a *simple but principled* objective that can measure the goodness of the resulting representations in terms of all these properties? The key to these questions is to find a principled "measure of compactness" or "information gain" for the distribution of a random variable $\boldsymbol{z}$ or from its finite samples $\{\boldsymbol{z}_i\}_{i=1}^N$. Such a measure should directly and accurately characterize intrinsic geometric or statistical properties of the distribution, in terms of its intrinsic dimension or volume. Unlike the cross entropy (4.2.2) or information bottleneck (4.2.5), such a measure should not depend exclusively on class labels so that it can work in more general settings such as supervised, self-supervised, semi-supervised, and unsupervised settings (as we will see in later sections).

Without loss of generality, assume that the distribution $\mathcal{D}$ of the random vector $\boldsymbol{x}$ is supported on a mixture of distributions, i.e., $\mathcal{D} = \cup_{k=1}^K \mathcal{D}_k$, where each $\mathcal{D}_k \subset \mathbb{R}^D$ has a low intrinsic dimension in the high-dimensional ambient space $\mathbb{R}^D$. Let $\boldsymbol{X}_k \in \mathbb{R}^{D \times N_k}$ denote the data matrix whose columns are samples drawn from the distribution $\mathcal{D}_k$, where N_k denotes the number of samples for each $k = 1, \ldots, K$. Then, we use $\boldsymbol{X} = [\boldsymbol{X}_1, \ldots, \boldsymbol{X}_K] \in \mathbb{R}^{D \times N}$ to denote all the samples, where $N = \sum_{k=1}^K N_k$. Recall that we also use $\boldsymbol{x}_i$ to denote the i-th sample of $\boldsymbol{X}$, i.e., $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$. Under an encoding mapping:

$$\boldsymbol{x} \xrightarrow{\quad f(\boldsymbol{x}) \quad} \boldsymbol{z}, \tag{4.2.9}$$

the input samples are mapped to $\boldsymbol{z}_i = f(\boldsymbol{x}_i)$ for each $i = 1, \ldots, N$. With an abuse of notation, we also write $\boldsymbol{Z}_k = f(\boldsymbol{X}_k)$ and $\boldsymbol{Z} = f(\boldsymbol{X})$. Therefore, we have $\boldsymbol{Z} = [\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_K]$ and $\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots \boldsymbol{z}_N]$.

On one hand, for learned features to be discriminative, features of different classes/clusters are preferred to be *maximally incoherent* to each other. Hence, they together should span a space of the largest possible volume (or dimension) and the coding rate of the whole set $\boldsymbol{Z}$ should be as large as possible. On the other hand, learned features of the same class/cluster should be highly correlated and coherent. Hence, each class/cluster should only span a space (or subspace) of a very small volume and the coding rate should be as small as possible. Now, we will introduce how to measure the coding rate of the learned features.

Coding rate of features. Notably, a practical challenge in evaluating the coding rate is that the underlying distribution of the feature representations $\boldsymbol{Z}$ is typically unknown. To address this, we may approximate the features $\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N]$ as samples drawn from a multivariate Gaussian distribution. Under this assumption, as discussed in Section 4.1.3, the compactness of the features $\boldsymbol{Z}$ *as a whole* can be measured in terms of the average coding length per sample, referred to as the *coding rate*, subject to a precision level $\epsilon > 0$ (see (4.1.30)) defined as follows:

$$R_\epsilon(\boldsymbol{Z}) = \frac{1}{2} \operatorname{logdet}\left(\boldsymbol{I} + \frac{d}{N\epsilon^2}\boldsymbol{Z}\boldsymbol{Z}^\top\right). \tag{4.2.10}$$

On the other hand, we hope that a nonlinear transformation $f(\boldsymbol{x})$ maps each class-specific submanifold $\mathcal{M}_k \subset \mathbb{R}^D$ to a maximally incoherent linear subspace $\mathcal{S}_k \subset \mathbb{R}^d$ such that the learned features $\boldsymbol{Z}$ lie in a union of low-dimensional subspaces. This structure allows for a more accurate evaluation of the coding rate by analyzing each subspace separately. Recall that the columns of $\boldsymbol{Z}_k$ denotes the features of the samples in $\boldsymbol{X}_k$ for each $k = 1, \ldots, K$. The coding rate for the features in $\boldsymbol{Z}_k$ can be computed as follows:

$$R_\epsilon(\boldsymbol{Z}_k) = \frac{N_k}{2N} \log\det\left(\boldsymbol{I} + \frac{d}{N_k\epsilon^2}\boldsymbol{Z}_k\boldsymbol{Z}_k^\top\right) \tag{4.2.11}$$

Then, the sum of the average coding rates of features in each class is

$$R_\epsilon^c(\boldsymbol{Z}) \doteq \sum_{k=1}^{K} R_\epsilon(\boldsymbol{Z}_k), \tag{4.2.12}$$

Therefore, a good representation $\boldsymbol{Z}$ of $\boldsymbol{X}$ is the one that achieves a large difference between the coding rate for the whole and that for all the classes:

$$\Delta R_\epsilon(\boldsymbol{Z}) \doteq R_\epsilon(\boldsymbol{Z}) - R_\epsilon^c(\boldsymbol{Z}). \tag{4.2.13}$$

Notice that, as per our discussions earlier in this chapter, this difference can be interpreted as the amount of "information gained" by identifying the correct low-dimensional clusters $\boldsymbol{Z}_k$ within the overall set $\boldsymbol{Z}$.

If we choose our feature mapping $f(\cdot)$ to be a deep neural network $f(\cdot, \theta)$ with network parameters θ, the overall process of the feature representation and the resulting rate reduction can be illustrated by the following diagram:

$$\boldsymbol{X} \xrightarrow{\quad f(\boldsymbol{x},\theta) \quad} \boldsymbol{Z} \xrightarrow{\quad \epsilon \quad} \Delta R_\epsilon(\boldsymbol{Z}). \tag{4.2.14}$$

Note that ΔR_ϵ is *monotonic* in the scale of the features $\boldsymbol{Z}$. To ensure fair comparison across different representations, it is essential to *normalize the scale* of the learned features. This can be achieved by either imposing the Frobenius norm of each class $\boldsymbol{Z}_k$ to scale with the number of features in $\boldsymbol{Z}_k \in \mathbb{R}^{d\times N_k}$, i.e., $\|\boldsymbol{Z}_k\|_F^2 = N_k$, or by normalizing each feature to be on the unit sphere, i.e., $\boldsymbol{z}_i \in \mathbb{S}^{d-1}$, where N_k denotes the number of samples in the k-th class. This formulation offers a natural justification for the need for "batch normalization" in the practice of training deep neural networks [IS15].

Once the representations are comparable, the goal becomes to learn a set of features $\boldsymbol{Z} = f(\boldsymbol{X}, \theta)$ such that they maximize the reduction between the coding rate of all features and that of the sum of features w.r.t. their classes:

$$\begin{aligned} &\max_\theta \Delta R_\epsilon\big(\boldsymbol{Z}\big) \doteq R_\epsilon(\boldsymbol{Z}) - R_\epsilon^c(\boldsymbol{Z}), \\ &\text{s.t.} \quad \boldsymbol{Z} = f(\boldsymbol{X}, \theta),\ \|\boldsymbol{Z}_k\|_F^2 = N_k,\ k = 1, \ldots, K. \end{aligned} \tag{4.2.15}$$

We refer to this as the principle of *maximal coding rate reduction* (MCR^2), as an embodiment of Aristotle's famous quote:

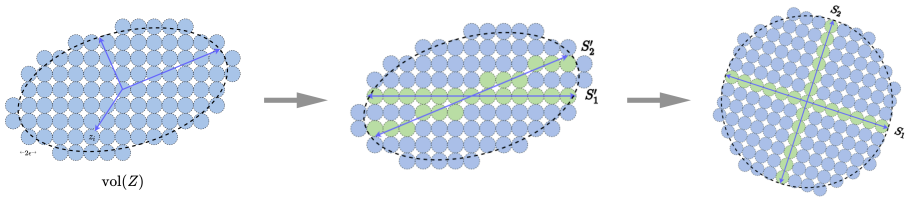


Figure 4.15: After identifying the low-dimensional data distribution, we would like to further transform the data distribution to a more informative structured representation: R is the number of ϵ-balls covering the whole space and R^c is the sum of the numbers for all the subspaces (the green balls). ΔR is their difference (the number of blue balls).

"*The whole is greater than the sum of its parts.*"

To learn the best representation, we require that *the whole is maximally greater than the sum of its parts*. To a large extent, the quantity ΔR can be viewed as a computable realization to the mutual information term $I(\boldsymbol{z}; \boldsymbol{y})$ introduced in the objective (4.2.7).

Example 4.11. To further elucidate the ideas, let us examine the example shown in Figure 4.15. From a compression perspective, the representation on the right is *the most compact one* in the sense that the difference between the coding rate when all features are encoded as a single Gaussian (blue) and that when the features are properly clustered and encoded as two separate subspaces (green) is maximal.[28] ■

Note that the above MCR^2 principle is designed for supervised learning problems, where the group memberships (or class labels) are known. There are a variety of ways to use the principle, or suitable generalizations, in ways which circumvent this requirement for class labels; we will discuss these later in this Chapter and in the next Chapter 5. To get a first taste, we extend the principle to unsupervised learning problems by introducing a membership matrix, which encodes the (potentially soft) assignment of each data point to latent groups or clusters. Specifically, let $\boldsymbol{\Pi} = \{\boldsymbol{\Pi}_k\}_{k=1}^K \subset \mathbb{R}^{N\times N}$ be a set of diagonal matrices whose diagonal entries encode the membership of the N samples into K classes. That is, $\boldsymbol{\Pi}$ lies in a simplex $\Omega \doteq \{\boldsymbol{\Pi} : \boldsymbol{\Pi}_k \geq \mathbf{0} : \sum_{k=1}^K \boldsymbol{\Pi}_k = \boldsymbol{I}_N\}$. Then, we can define the average coding rate with respect to the partition $\boldsymbol{\Pi}$ as

$$R_\epsilon^c(\boldsymbol{Z} \mid \boldsymbol{\Pi}) \doteq \sum_{k=1}^K \frac{\operatorname{tr}(\boldsymbol{\Pi}_k)}{2N} \log\det\left(\boldsymbol{I} + \frac{d}{\operatorname{tr}(\boldsymbol{\Pi}_k)\epsilon^2}\boldsymbol{Z}\boldsymbol{\Pi}_k\boldsymbol{Z}^\top\right). \tag{4.2.16}$$

When $\boldsymbol{Z}$ is given, $R_\epsilon^c(\boldsymbol{Z}|\boldsymbol{\Pi})$ is a concave function of $\boldsymbol{\Pi}$. Then the MCR^2 principle for unsupervised learning problems becomes as follows:

$$\max_{\boldsymbol{\Pi},\theta} \Delta R_\epsilon\big(\boldsymbol{Z} \mid \boldsymbol{\Pi}\big) \doteq R_\epsilon(\boldsymbol{Z}) - R_\epsilon^c(\boldsymbol{Z} \mid \boldsymbol{\Pi})$$

[28] Intuitively, the ratio between the "volume" of the whole space spanned by all features and that actually occupied by the features is maximal.

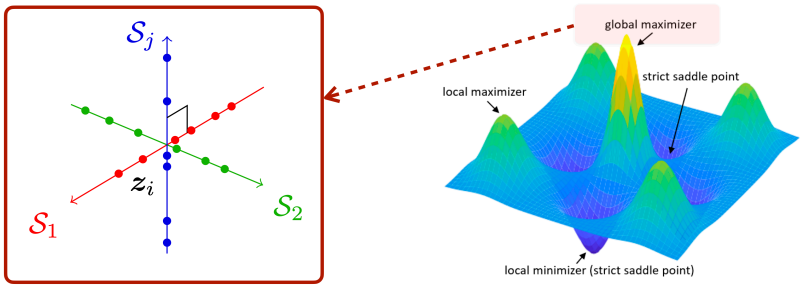


Figure 4.16: **Local optimization landscape:** According to Theorem 4.2, the global maximum of the rate reduction objective corresponds to a solution with mutually incoherent subspaces.

$$\text{s.t.} \quad \boldsymbol{Z} = f(\boldsymbol{X}, \theta),\ \|\boldsymbol{Z}\boldsymbol{\Pi}_k\|_F^2 = N_k,\ k = 1, \ldots, K,\ \boldsymbol{\Pi} \in \Omega. \tag{4.2.17}$$

Compared to (4.2.15), the formulation here allows for the joint optimization of both the group memberships and the network parameters. In particular, when $\boldsymbol{\Pi}$ is fixed to a group membership matrix that assigns N data points into K groups, Problem (4.2.17) can recover Problem (4.2.15).

Remark 4.7 (Computing Coding Rates). It is a classical fact that computing the log-determinant of an $n \times n$ matrix takes $\mathcal{O}(n^3)$ time to compute. Hence computing the coding rates ΔR_ϵ or R_ϵ (etc.) seem intractable at the scale of modern data analysis. There are several ways around this, which we will use without comment in the experiments that power the rest of the book.

- *Use the structure of the coding rate.* Suppose that $\boldsymbol{Z}$ is an element of $\mathbb{R}^{d \times n}$. Then it holds

  $$\operatorname{logdet}(\boldsymbol{I} + \alpha \boldsymbol{Z}\boldsymbol{Z}^\top) = \operatorname{logdet}(\boldsymbol{I} + \alpha \boldsymbol{Z}^\top \boldsymbol{Z}) \tag{4.2.18}$$

  so computing the matrix product and computing the coding rates requires time on the order of $\min\{d^2 n + d^3, dn^2 + n^3\}$.

- *Use the symmetry of the Gramian.* Note that $\boldsymbol{Z}\boldsymbol{Z}^\top$ (and similarly $\boldsymbol{Z}^\top \boldsymbol{Z}$) is a symmetric matrix. Hence it is relatively cheap (though still cubic time) to compute its eigenvalues $\lambda_i(\boldsymbol{Z}\boldsymbol{Z}^\top)$. Then the log-determinant can be expressed as follows:

  $$\operatorname{logdet}(\boldsymbol{I} + \alpha \boldsymbol{Z}\boldsymbol{Z}^\top) = \sum_{i=1}^{d} \log(1 + \alpha \lambda_i(\boldsymbol{Z}\boldsymbol{Z}^\top)), \tag{4.2.19}$$

  (and similarly for $\boldsymbol{Z}^\top \boldsymbol{Z}$). However, this method is made redundant by the next method, which uses more structure of the input and performs better.

- *Use the positive definiteness of the input to the log determinant.* Since $\boldsymbol{Z}\boldsymbol{Z}^\top$ (and similarly $\boldsymbol{Z}^\top \boldsymbol{Z}$) is symmetric positive semidefinite, the quantity

$\boldsymbol{I} + \alpha \boldsymbol{Z}\boldsymbol{Z}^\top$ is symmetric positive definite. This means we can compute the Cholesky decomposition $\boldsymbol{L}\boldsymbol{L}^\top = \boldsymbol{I} + \alpha \boldsymbol{Z}\boldsymbol{Z}^\top$, where $\boldsymbol{L} \in \mathbb{R}^{d\times d}$ is a lower-triangular matrix with positive diagonal entries. Then

$$\operatorname{logdet}(\boldsymbol{I} + \alpha \boldsymbol{Z}\boldsymbol{Z}^\top) = 2\sum_{i=1}^{d} \log(L_{ii}). \tag{4.2.20}$$

This method tends to be the cheapest in practice, especially when also strategically deciding whether to compute $\boldsymbol{Z}\boldsymbol{Z}^\top$ or $\boldsymbol{Z}^\top\boldsymbol{Z}$, though it remains at cubic time complexity.

4.2.4 Optimization Properties of Coding Rate Reduction

In this subsection, we study the optimization properties of the MCR^2 function by analyzing its optimal solutions and the structure of its optimization landscape. To get around the technical difficulty introduced by the neural networks, we consider a simplified version of Problem (4.2.15) as follows:

$$\max_{\boldsymbol{Z}} \; R_\epsilon(\boldsymbol{Z}) - R_\epsilon^c(\boldsymbol{Z}) \qquad \text{s.t.} \quad \|\boldsymbol{Z}_k\|_F^2 = N_k, \; k = 1, \ldots, K. \tag{4.2.21}$$

In theory, the MCR^2 principle (4.2.21) benefits from great generalizability and can be applied to representations $\boldsymbol{Z}$ of *any* distributions as long as the rates R_ϵ and R_ϵ^c for the distributions can be accurately and efficiently evaluated. The optimal representation $\boldsymbol{Z}^*$ should have some interesting geometric and statistical properties. We here reveal nice properties of the optimal representation with the special case of subspaces, which have many important use cases in machine learning. When the desired representation for $\boldsymbol{Z}$ is multiple subspaces, the rates R_ϵ and R_ϵ^c in (4.2.21) are given by (4.2.10) and (4.2.12), respectively. At the maximal rate reduction, MCR^2 achieves its optimal representations, denoted as $\boldsymbol{Z}^* = [\boldsymbol{Z}_1^*, \ldots, \boldsymbol{Z}_K^*]$ with $\operatorname{rank}(\boldsymbol{Z}_k^*) \le d_k$. One can show that $\boldsymbol{Z}^*$ has the following desired properties (see [YCY+20] for a formal statement and detailed proofs).

Theorem 4.2 (**Characterization of Global Optimal Solutions**). *Suppose $\boldsymbol{Z}^* = [\boldsymbol{Z}_1^*, \ldots, \boldsymbol{Z}_K^*]$ is a global optimal solution of Problem (4.2.21). The following statements hold:*

- Between-Class Discriminative*: As long as the ambient space is adequately large ($d \ge \sum_{k=1}^K d_k$), the subspaces are all orthogonal to each other,* i.e.*, $(\boldsymbol{Z}_k^*)^\top \boldsymbol{Z}_l^* = \boldsymbol{0}$ for $k \ne l$.*
- Maximally Diverse Representation*: As long as the coding precision is adequately high, i.e., $\epsilon^4 < c \cdot \min_k \left\{ \frac{N_k}{N} \frac{d^2}{d_k^2} \right\}$, where $c > 0$ is a constant. Each subspace achieves its maximal dimension, i.e. $\operatorname{rank}(\boldsymbol{Z}_k^*) = d_k$. In addition, the largest $d_k - 1$ singular values of $\boldsymbol{Z}_k^*$ are equal.*

This theorem indicates that the MCR^2 principle promotes embedding of data into multiple independent subspaces (as illustrated in Figure 4.16), with features distributed *isotropically* in each subspace (except for possibly one dimension). Notably, this theorem also confirms that the features learned by the MCR^2 principle exhibit the desired low-dimensional discriminative properties discussed in Section 4.2.2. In addition, among all such discriminative representations, it prefers the one with the highest dimensions in the ambient space. This is substantially different from the objective of information bottleneck (4.2.5).

Regularized MCR^2. The above theorem characterizes properties of the global optima of the rate reduction objectives. What about other optima, such as local ones? Due to the constraints of the Frobenius norm, it is a difficult task to analyze Problem (4.2.21) from an optimization-theoretic perspective. Therefore, we consider the Lagrangian formulation of (4.2.21). This can be viewed as a tight relaxation or even an equivalent problem of (4.2.21) whose optimal solutions agree under specific settings of the regularization parameter; see [WLP+24, Proposition 3.2]. Specifically, the formulation we study, referred to henceforth as the *regularized MCR^2 problem*, is as follows:

$$\max_{\boldsymbol{Z}} \; R_\epsilon(\boldsymbol{Z}) - R_\epsilon^c(\boldsymbol{Z}) - \frac{\lambda}{2}\|\boldsymbol{Z}\|_F^2, \tag{4.2.22}$$

where $\lambda > 0$ is the regularization parameter. Although the program (4.2.22) is highly nonconcave and involves matrix inverses in its gradient computation, we can still explicitly characterize its local and global optima as follows (see [WLP+24, Theorem 3.1] for detailed proofs).

Theorem 4.3 (**Local and Global Optima**). *Let N_k denote the number of training samples in the k-th class for each $k \in \{1, \dots, K\}$, $N_{\max} \doteq \max\{N_1, \dots, N_K\}$, $\alpha = d/(N\epsilon^2)$, and $\alpha_k = d/(N_k\epsilon^2)$ for each $k \in \{1, \dots, K\}$. Given a coding precision $\epsilon > 0$, if the regularization parameter satisfies*

$$\lambda \in \left(0, \frac{d(\sqrt{N/N_{\max}} - 1)}{N(\sqrt{N/N_{\max}} + 1)\epsilon^2}\right], \tag{4.2.23}$$

then the following statements hold:
*(i) (***Local maximizers***) $\boldsymbol{Z}^* = [\boldsymbol{Z}_1^*, \dots, \boldsymbol{Z}_K^*]$ is a local maximizer of Problem (4.2.22) if and only if the k-th block admits the following decomposition*

$$\boldsymbol{Z}_k^* = \left(\frac{\eta_k + \sqrt{\eta_k^2 - 4\lambda^2 N/N_k}}{2\lambda\alpha_k}\right)^{1/2} \boldsymbol{U}_k \boldsymbol{V}_k^\top, \tag{4.2.24}$$

where (a) $r_k = \operatorname{rank}(\boldsymbol{Z}_k^*)$ satisfies $r_k \in [0, \min\{N_k, d\})$ and $\sum_{k=1}^K r_k \le \min\{N, d\}$, (b) $\boldsymbol{U}_k \in \mathcal{O}^{d\times r_k}$ satisfies $\boldsymbol{U}_k^\top \boldsymbol{U}_l = \boldsymbol{0}$ for all $k \neq l$, $\boldsymbol{V}_k \in \mathcal{O}^{N_k \times r_k}$, and (c) $\eta_k := (\alpha_k - \alpha) - \lambda(N/N_k + 1)$ for each $k \in \{1, \dots, K\}$.
*(ii) (***Global maximizers***) $\boldsymbol{Z}^* = [\boldsymbol{Z}_1^*, \dots, \boldsymbol{Z}_K^*]$ is a global maximizer of Problem*

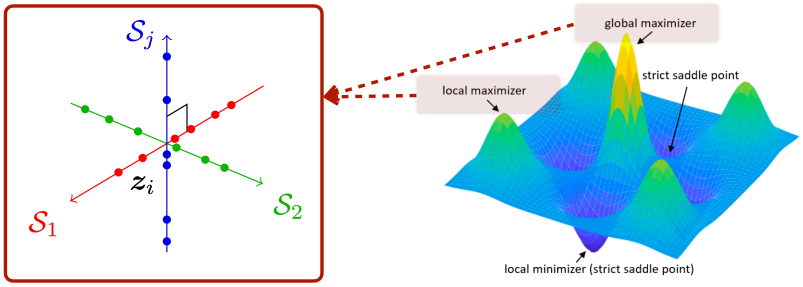


Figure 4.17: **Global optimization landscape:** According to [LSJ+16; SQW15], Theorems 4.3 and 4.4, both global and local maxima of the (regularized) rate reduction objective correspond to a solution with mutually incoherent subspaces. All other critical points are strict saddle points.

(4.2.22) *if and only if (a) it satisfies all the above conditions and* $\sum_{k=1}^{K} r_k = \min\{N, d\}$*, and (b) for all* $k \neq l \in [K]$ *satisfying* $N_k < N_l$ *and* $r_l > 0$*, we have* $r_k = \min\{N_k, d\}$.

This theorem explicitly characterizes the local and global optima of problem (4.2.22). Intuitively, this shows that the features represented by each local maximizer of Problem (4.2.22) are low-dimensional and discriminative. Although we have characterized the local and global optimal solutions in Theorem 4.3, it remains unknown whether these solutions can be efficiently computed using GD to solve the problem (4.2.22), since GD may get stuck at other critical points such as a saddle point. Fortunately, [LSJ+16; SQW15] showed that if a function is twice continuously differentiable and satisfies the *strict saddle property*, i.e., each critical point is either a local maximizer or a strict saddle point[29], GD converges to a local maximizer almost surely with random initialization. We investigate the global optimization landscape of the problem (4.2.22) by characterizing all its critical points as follows (see [WLP+24, Theorem 3.3] for detailed proofs).

Theorem 4.4 (**Benign Global Optimization Landscape**). *Given a coding precision* $\epsilon > 0$*, if the regularization parameter satisfies* (4.2.23)*, it holds that any critical point* $\boldsymbol{Z}$ *of the problem* (4.2.22) *is either a local maximizer or a strict saddle point.*

Together, the above two theorems show that the learned features associated with each local maximizer of the rate reduction objective — not just global maximizers — are structured as incoherent low-dimensional subspaces. Furthermore, the (regularized) rate reduction objective (4.2.15) has a very benign landscape with only local maxima and strict saddles as critical points, as illustrated in

[29] We say that a critical point is a strict saddle point of Problem (4.2.22) if it has a direction with strictly positive curvature [SQW15]. This includes classical saddle points with strictly positive curvature as well as strict local minimizers.

Figure 4.17. According to [LSJ+16; SQW15], Theorems 4.3 and 4.4 imply that low-dimensional and discriminative representations (LDRs) can be efficiently found by applying (stochastic) gradient descent to the rate reduction objective (4.2.15) from random initialization. These results also indirectly explain why in Section 4.3.1, if the chosen network is expressive enough and trained well, the resulting representation typically gives an incoherent linear representation that likely corresponds to the globally optimal solution. Interested readers are referred to [WLP+24] for proofs.

4.3 Learning Representations for Imagery Data

4.3.1 Supervised Representation Learning

We here present how the MCR^2 objective (4.2.15) helps learn better representations than the cross entropy (CE) (4.2.2) for image classification. Here, for simplicity,[30] we adopt the popular neural network architecture, the ResNet-18 [HZR+16b], to model the feature mapping $\boldsymbol{z} = f(\boldsymbol{x}, \theta)$. We optimize the neural network parameters θ via the backpropagation algorithm (detailed in Section A.2.3) to maximize the coding rate reduction. More implementation details of this experiment can be found in [CYY+22].

Figure 4.18 illustrates how the two rates and their difference (for both training and test data) evolve over epochs of training: After an initial phase, R_ϵ gradually increases while R_ϵ^c decreases, indicating that features $\boldsymbol{Z}$ are expanding as a whole while each class $\boldsymbol{Z}_k$ is being compressed.

We first evaluate the performance with the CIFAR-10 image classification dataset [KH+09] and the results are shown in the table of Figure 4.19. As we can see, the MCR^2 objective achieves a comparable performance as the cross entropy (CE) loss. In addition, the MCR^2 objective is more robust to corruptions in the class labels since it learns a joint (subspace) representation for each class instead of trying to fit the labels of individual samples.

Figure 4.20 shows the distribution of singular values of the learned features $\boldsymbol{Z}_k$ per class. From the distribution of the singular values shown in Figure 4.20a, we see that the features $\boldsymbol{z}$ learned from MCR^2 for each class are distributed on a subspace of dimension around 10-12, whereas the features from cross entropy Figure 4.20b are of much lower dimension.[31]

Figure 4.21 shows the cosine similarities between the learned features sorted by class. We also compare the similarities of the learned features by using the cross-entropy (CE) (4.2.2) and the MCR^2 objective (4.2.15). From the plots, one can clearly see that the features learned by the MCR^2 objective for different classes are much more incoherent (or independent) than features learned by using cross-entropy loss. Within each class, the features also are less coherent

[30] We will rigorously study and justify how design or derive a proper deep architecture from and for the MCR^2 objective in Chapter 5.

[31] If one continue to train with more iterations, the CE features will may even collapse to a single dimension, known as the neural collapse phenomenon discussed earlier in Remark 4.4.

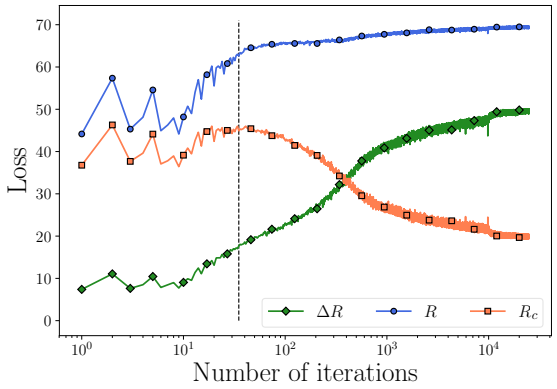


Figure 4.18: Evolution of R_ϵ, R_ϵ^c, ΔR_ϵ during the training process.

	Ratio=0.0	Ratio=0.1	Ratio=0.2	Ratio=0.3	Ratio=0.4	Ratio=0.5
CE Training	0.939	0.909	0.861	0.791	0.724	0.603
MCR^2 Training	**0.940**	**0.911**	**0.897**	**0.881**	**0.866**	**0.843**

Figure 4.19: Comparison of classification accuracy between training with the cross entropy (CE) loss or the MCR^2 objective, with different ratios of the class labels being randomly corrupted.

hence more diverse, as indicated by the distribution of their singular values shown in Figure 4.20a.

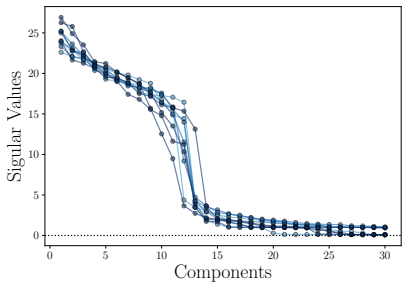
We visualize the images that are mapped along different singular vectors in the feature subspace of each class, which we call the "principal" images. Figure 4.22 shows these principal images along each of the top-10 singular vectors. As we can see, the images along each singular vector are very similar in their appearances and structures and images along different singular vectors are very distinct from each other.

Despite the fact that the MCR^2 objective seems to allow us to learn better representations, there has been an apparent lack of justification of the network architectures used in the above experiments. It is yet unclear why the network adopted here (the ResNet-18) is suitable for representing the map $f(\boldsymbol{x}, \theta)$, let alone for interpreting the layer operators and parameters θ learned inside. In Chapter 5, *we will show how to derive network architectures and components entirely as a "white box" from the desired objective (say the rate reduction).*

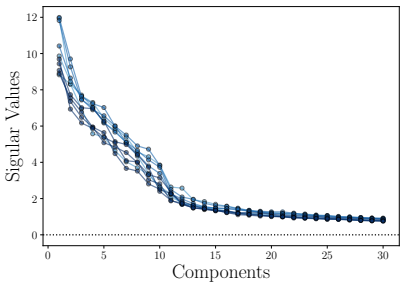
4.3.2 Unsupervised Representation Learning

As we have seen in the above section, the class information makes learning a structured representation of the (mixed) data distribution relatively easy. However, such information is often not available, at least not at a massive scale.

Now we present how to extend the MCR^2 principle to the unsupervised set-

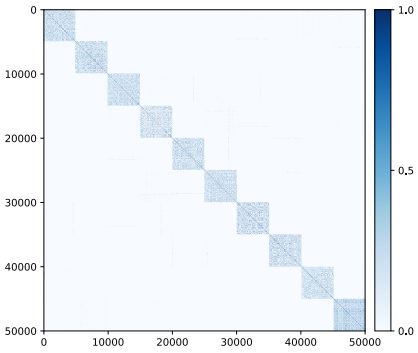


(a) PCAs of features for individual classes learned by MCR^2.



(b) PCAs of features for individual classes learned by Cross Entropy.

Figure 4.20: Comparison of the principal components of learned features from MCR^2 versus those from cross entropy.



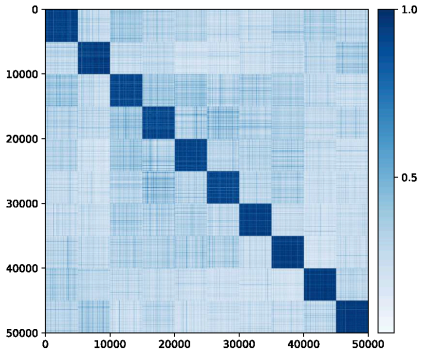
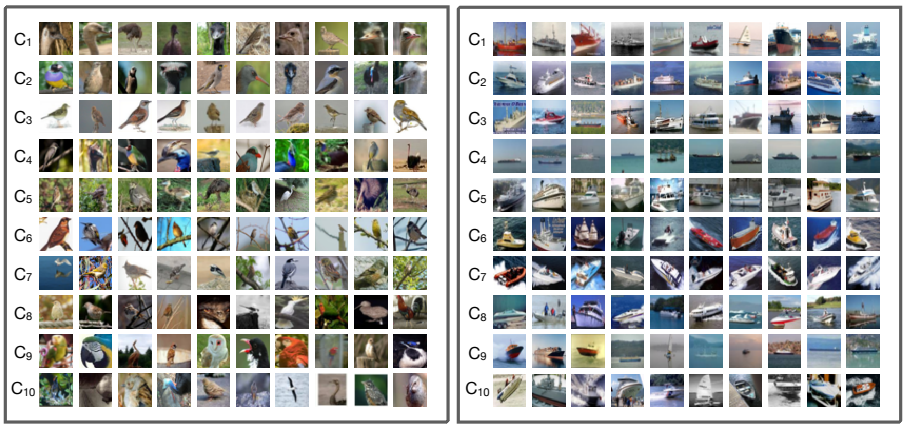


Figure 4.21: Cosine similarity between learned features by using the MCR^2 objective (**left**) and CE loss (**right**).

ting. Contrary to supervised learning where the class labels are known, in unsupervised learning, the group memberships of the data samples are unknown. In this case, we can apply certain class of augmentations or transformations, say τ with a distribution P_τ, to each sample. For example, for images, augmentations typically include translations, rotations, or cropping of each image sample. We may view the augmented samples of each data point as belonging to the same group or cluster.

More specifically, given a set of N data samples $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$, we generate m augmented views $\{\tau_i\}_{i=1}^m$ for each data point $\boldsymbol{x}_k$, denoted as $\tilde{\boldsymbol{X}}_k = [\tau_1(\boldsymbol{x}_k), \tau_2(\boldsymbol{x}_k), \ldots, \tau_m(\boldsymbol{x}_k)]$. We treat the augmented samples $\tilde{\boldsymbol{X}}_i$ as the i-th class with $K = N$ groups/clusters in total. Now, we can apply the MCR^2 principle as we did before in the supervised learning case. Specifically, we learn the



(a) Bird (b) Ship

Figure 4.22: Top-10 "principal" images for class - "Bird" and "Ship" in the CIFAR-10 image dataset.

feature mapping $f(\boldsymbol{x}, \theta)$ by maximizing the coding rate reduction as follows:

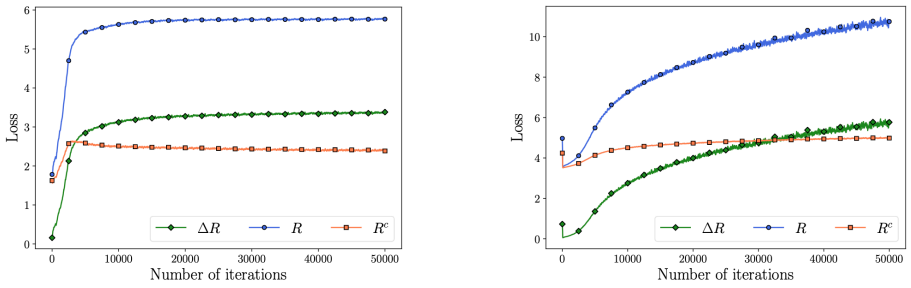
$$\begin{aligned} &\max_{\theta} \Delta R_\epsilon\big(\boldsymbol{Z}\big) \doteq R_\epsilon(\boldsymbol{Z}) - R_\epsilon^c(\boldsymbol{Z}), \\ &\text{s.t.} \quad \boldsymbol{Z} = f(\tilde{\boldsymbol{X}}, \theta),\ \|\boldsymbol{Z}_k\|_F^2 = m,\ k = 1, \ldots, N. \end{aligned} \tag{4.3.1}$$

Here, $\tilde{\boldsymbol{X}} = [\tilde{\boldsymbol{X}}_1, \ldots, \tilde{\boldsymbol{X}}_N]$ and $\boldsymbol{Z}_k = f(\tilde{\boldsymbol{X}}_k, \theta)$ denotes the features of the augmented samples of $\boldsymbol{x}_k$ for each $k = 1, \ldots, N$.

Let us again adopt ResNet-18 as the feature mapping $f(\boldsymbol{x}, \theta)$ and optimize the network parameters θ to maximize the unsupervised objective (4.3.1). After training on the CIFAR-10 dataset, we visualize the evolution of the two rates and their difference (for both training and test data) over epochs of training in Figure 4.23a. Similar to the supervised learning case, after an initial phase, R_ϵ gradually increases while R_ϵ^c decreases, indicating that features $\boldsymbol{Z}$ are expanding as a whole while each cluster $\boldsymbol{Z}_k$ is being compressed. Compared to the supervised case shown in Figure 4.18, we can observe that, in the unsupervised case, the total coding rate R_ϵ expands rapidly and the overall MCR^2 objective saturates at a local optimum. To alleviate this, we can alter the optimization strategy by controlling the dynamics of R_ϵ and R_ϵ^c. Specifically, we can rescale the rates by replacing the coding rate $R_\epsilon(\boldsymbol{Z})$ in (4.3.1) with

$$\tilde{R}_\epsilon(\boldsymbol{Z}) \doteq \frac{1}{2\gamma_1} \log\det\left(\boldsymbol{I} + \frac{\gamma_2 d}{Nm\epsilon^2}\boldsymbol{Z}\boldsymbol{Z}^\top\right).$$

By setting $\gamma_1 = \gamma_2 = N$, we arrive at a modified learning objective, referred to as MCR^2-CTRL, whose training dynamics are shown in Figure 4.23b. As we can see, with this controlled MCR^2 objective, features are first compressed and then gradually expanded, leading to a higher value of the overall objective $\Delta R_\epsilon(\boldsymbol{Z})$.



(a) MCR^2 (b) MCR^2-ctrl

Figure 4.23: Evolution of the rates of MCR^2 in the training process for unsupervised learning on CIFAR-10.

Table 2: Clustering results on CIFAR10, CIFAR100, and STL10 datasets.

Dataset	Metric	K-Means	JULE	RTM	DEC	DAC	DCCM	MCR^2-Ctrl
CIFAR10	NMI	0.087	0.192	0.197	0.257	0.395	0.496	**0.630**
	ACC	0.229	0.272	0.309	0.301	0.521	0.623	**0.684**
	ARI	0.049	0.138	0.115	0.161	0.305	0.408	**0.508**
CIFAR100	NMI	0.084	0.103	-	0.136	0.185	0.285	**0.387**
	ACC	0.130	0.137	-	0.185	0.237	0.327	**0.375**
	ARI	0.028	0.033	-	0.050	0.087	0.173	**0.178**
STL10	NMI	0.124	0.182	-	0.276	0.365	0.376	**0.446**
	ACC	0.192	0.182	-	0.359	0.470	0.482	**0.491**
	ARI	0.061	0.164	-	0.186	0.256	0.262	**0.290**

Figure 4.24: Comparison of clustering performance accuracy between training with the MCR^2-ctrl objective and other unsupervised learning algorithms on CIFAR-10, CIFAR-100, and STL-10 datasets.

To assess the quality of the learned representations and verify whether they have good subspace structures, we adopt a subspace clustering algorithm EnSC [YLR+16]. For evaluation metrics, we consider normalized mutual information (NMI), clustering accuracy (ACC), and adjusted rand index (ARI) [32]. For comparison, we consider methods including JULE [YPB16], RTM [NMM19], DEC [XGF16], DAC [CWM+17], and DCCM [WLW+19]. The clustering results are summarized in Figure 4.24. As we can see, the features learned by maximizing the MCR^2-ctrl objective achieve better clustering accuracy than those learned by other algorithms.

There is one main drawback though: the MCR^2-based unsupervised learning method requires a costly log det operation of $\mathsf{O}(d^3)$ complexity *per sample* when computing R_ϵ^c. This quickly becomes computationally prohibitive as we try to scale to larger datasets. Instead of minimizing the exact coding rate for each

[32] Refer to the Appendix in [YCY+20] for their specific definitions.

Method	Model	Epochs	20-NN	Linear Probing
DINO	ViT-B	100	72.9	76.3
SimDINO	ViT-B	100	**74.9**	**77.3**
DINO	ViT-L	100	–	–
SimDINO	ViT-L	100	**75.6**	**77.4**

Table 4.1: **Classification performance** on ImageNet-1k test set for DINO and SimDINO, using both k-nearest neighbor accuracy ($k = 20$) and linear probing. SimDINO outperforms DINO in both accuracy and training stability. DINO training on ViT-L is unstable and quickly diverges, while SimDINO trains stably.

group of augmented samples $\tilde{\boldsymbol{X}}_i$, we can approximate the effect of compressing features in each group by maximizing the *cosine similarity* between the features of the augmented samples:[33]

$$\begin{aligned} &\max_{\theta} R_\epsilon(\boldsymbol{Z}) + \frac{\gamma}{N}\sum_{k=1}^{N}\sum_{i\neq j} \boldsymbol{z}_{k,j}^\top \boldsymbol{z}_{k,i}, \\ &\text{s.t.} \quad \boldsymbol{z}_{i,j} = f(\tau_j(\boldsymbol{x}_i), \theta), \ \ \|\boldsymbol{z}_{i,j}\|_2^2 = 1, \end{aligned} \tag{4.3.2}$$

where $\gamma > 0$ is a hyperparameter that controls the strength of the cosine similarity term, as it differs from the coding rate in magnitude. This objective avoids the costly $\log\det$ operation and is much more scalable. In fact, this objective is closely related to popular (unsupervised) *contrastive learning* methods such as SimCLR [CKN+20] and DINO [CTM+21], which have been engineered to learn high-quality representations from massive amount of images.

As we will show later in Section 8.2.2 of Chapter 8, this objective (4.3.2) can be implemented very efficiently to learn a highly structured representation $\boldsymbol{Z}$ that is highly linear and discriminative. The learned feature representation leads to state-of-the-art performance on large real-world datasets comparable or even better than features learned by DINO, but without the heavy engineering efforts needed for DINO. Hence, we refer to this objective as *Simplified DINO* (SimDINO). As a preview, Table 4.1 show that SimDINO outperforms DINO on ImageNet-1k classification using both k-nearest neighbor accuracy and linear probing (i.e. training a linear classifier on pretrained models). This demonstrates that the objective leads to highly structured and linearized representations. We can also utilize these representations for downstream tasks such as unsupervised segmentation[34], as illustrated in Figure 4.25. More implementation details and experimental results of SimDINO will be presented in Chapter 8.

[33] We leave as an exercise for the readers to verify under what conditions that minimizing the per class coding rate $R^c(\boldsymbol{Z})$ is (approximately) equivalent to maximizing the cosine similarity among the features $\boldsymbol{Z}$.

[34] Specifically, we adopt MaskCut [WGY+23], an effective unsupervised approach of extracting features from a frozen vision backbone for object detection and instance segmentation.

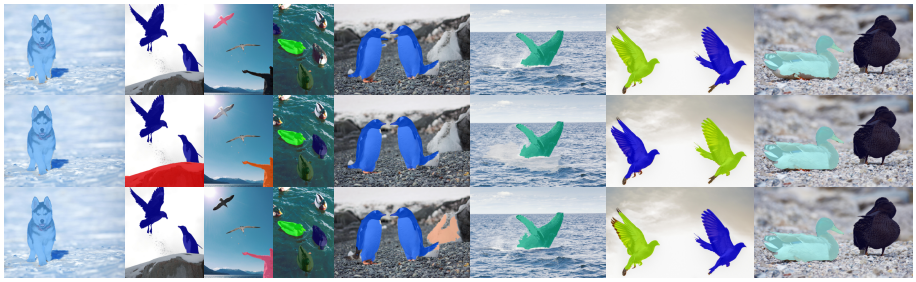


Figure 4.25: **Visualization of unsupervised segmentation results** from DINO ViT-B/16 (row 1), SimDINO ViT-B/16 (row 2) and SimDINO ViT-L/16 (row 3).

4.4 Summary and Notes

Key messages. In this Chapter, we have studied basic concepts and ideas behind how to learn a distribution from finite samples and obtain a *computable* encoding and decoding scheme for the distribution. For distributions that are continuous and possibly low-dimensional, we have argued that such a scheme is necessarily *lossy* – hence the importance of the concept of rate distortion. The noise introduced through distortion, due to quantization, plays a related but different role from the noise we have studied in the preceding chapter. One may think of the noise considered in Chapter 3 as "outside" of the submanifold of the support of the distribution whereas the noise considered in this chapter is "inside" the support. Informally, we may refer to the diffusion/denoising process outside of the support as "continuation" and the diffusion/denoising process inside the support as "percolation." A more unified analysis and understanding of their compounded effect is due for future study.

In this chapter, we introduced for the first time the concept of learning a *representation* for a distribution. We have considered arguably the simplest case, the case of classification. If one views the classification label $\boldsymbol{y}$ as a fixed simple (grossly lossy) code for the data $\boldsymbol{x}$, we may seek a consistent[35] representation $\boldsymbol{z} = f(\boldsymbol{x})$ of the data $\boldsymbol{x}$ such that it tends to maximize the mutual information between the representation and this simple code:

$$\max I(\boldsymbol{z}, \boldsymbol{y}) = H(\boldsymbol{z}) - H(\boldsymbol{z} \mid \boldsymbol{y}). \tag{4.4.1}$$

In addition, we require such a representation $\boldsymbol{z}$ is not only consistent to $\boldsymbol{x}$ but also much more *structured* than the native data distribution of $\boldsymbol{x}$ – classes are linearly discriminative. This would make the learned representation much more efficient to use than the native data distribution for subsequent tasks such as recognition, segmentation and completion. The maximal coding rate

[35] That is, we can faithfully recover $\boldsymbol{x}$ from $\boldsymbol{z}$ as we will make the notion of "consistent" representations clear later in Chapter 6.

reduction objective introduced is a natural and computable realization of these requirements. In Chapter 7, we will see how to generalize this objective to a much broader context when $\boldsymbol{y}$ could be a more general lossy code (say image caption) or a random variable related to $\boldsymbol{x}$.

4.5 Exercises and Extensions

Exercise 4.1. Please show the following properties of the $\log\det(\cdot)$ function.

1. Show that
$$f(\boldsymbol{X}) = \log\det(\boldsymbol{X})$$
is a concave function. (**Hint:** The function $f(\boldsymbol{x})$ is concave if and only if the one-dimensional function $f(\boldsymbol{x}+t\boldsymbol{h})$ is concave for all $\boldsymbol{x}$ and $\boldsymbol{h}$.)

2. Show that
$$\log\det(\boldsymbol{I}+\boldsymbol{X}^\top\boldsymbol{X}) = \log\det(\boldsymbol{I}+\boldsymbol{X}\boldsymbol{X}^\top)$$

3. Let $\boldsymbol{A}\in\mathbb{R}^{n\times n}$ be a positive definite matrix. Please show that:
$$\log\det(\boldsymbol{A}) = \sum_{i=1}^{n}\log(\lambda_i), \tag{4.5.1}$$
where $\lambda_1,\lambda_2,\ldots,\lambda_n$ are the eigenvalues of $\boldsymbol{A}$.

4. Show that the gradient of the $\log\det(\cdot)$ function is given by
$$\nabla\log\det(\boldsymbol{A}) = \boldsymbol{A}^{-1},$$
for a positive definite matrix $\boldsymbol{A}$.

Exercise 4.2. Let $\boldsymbol{Z} = [\boldsymbol{Z}_1,\ldots,\boldsymbol{Z}_K]\in\mathbb{R}^{d\times N}$, where each $\boldsymbol{Z}_k\in\mathbb{R}^{d\times N_k}$ and $N=\sum_{k=1}^{K}N_k$.

1. Please show that
$$\log\det\left(\boldsymbol{I}+\alpha\boldsymbol{Z}\boldsymbol{Z}^\top\right) \le \sum_{k=1}^{K}\log\det\left(\boldsymbol{I}+\alpha\boldsymbol{Z}_k\boldsymbol{Z}_k^\top\right),$$
where $\alpha>0$.

2. Please derive the necessary and sufficient condition under which the equality holds.

Exercise 4.3. Let $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ be two vectors of unit length. Let $\boldsymbol{Z}=[\boldsymbol{z}_1,\boldsymbol{z}_2]$ Show that the following three objectives are equivalent:

- $\max \boldsymbol{z}_1^\top \boldsymbol{z}_2$.
- $\min \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2^2$.
- $\max \log\det(\boldsymbol{I} + \boldsymbol{Z}\boldsymbol{Z}^\top)$.

However, they may no longer be exactly equivalent when there are more than two vectors in a high-dimensional space. The $\log\det$ is more general as it measures the total volume spanned by all the vectors in the high-dimensional space.

Exercise 4.4. Please derive the rate-distortion function for the Gaussian random vector introduced in Example 4.4.