Chapter 10

Unsupervised Learning and Self-supervised Learning

We venture into unsupervised learning by first studying classical (and analytically tractable) approaches to unsupervised learning. Classical unsupervised learning usually consists of specifying a latent variable model and fitting using the expectation-maximization (EM) algorithm. However, so far we do not have a comprehensive theoretical analysis for the convergence of EM algorithms because fundamentally analyzing EM algorithms involves understanding non-convex optimization. Most analysis of EM only applies to special cases (e.g., see Xu et al. [2016], Daskalakis et al. [2016]) and it is not clear whether any of the results can be extended to more realistic, complex setups, without a fundamentally new technique for understanding nonconvex optimization. Instead, we will analyze a family of algorithms which are broadly referred to as spectral methods or tensor methods, which are a particular application of the method of moments [Pearson, 1894] with the algorithmic technique of tensor decomposition [Anandkumar et al., 2015]. While the spectral method appears to be not as empirically sample-efficient as EM, it has provable guarantees and arguably is more reliable than EM given the provable guarantees.

After discussing the basics of classical unsupervised learning, we will move on to modern applications of deep learning. In particular, we'll focus on theoretical interpretations of contrastive learning, which is a class of successful self-supervised learning algorithms in computer vision.

10.1 Method of Moments for mixture models

We begin by formally describing the unsupervised learning problem. First, assume that we are studying a family of distributions P_θ parameterized by $\theta \in \Theta$, where P_θ can be described by a latent variable model. Then, given data $x^{(i)}, ..., x^{(n)}$ that is sampled i.i.d. from some distribution in $\{P_\theta\}_{\theta\in\Theta}$, our goal is to recover the true θ.

Perhaps the most well-studied latent variable model in machine learning is the mixture of Gaussians. We consider the following model for the mixture of k d-dimensional Gaussians. Let

$$\theta = ((\mu_1, \cdots, \mu_k), (p_1, \cdots, p_k)), \tag{10.1}$$

where $\mu_i \in \mathbb{R}^d$ is the mean of the i-th component and p is a vector of probabilities belonging to the k-simplex, which represents the mixture coefficient for clusters. Formally, for $\Delta(k) \triangleq \{p : \|p\|_1 = 1, p \geq 0, p \in \mathbb{R}^k\}$,

$$p = (p_1, \cdots, p_k) \in \Delta(k). \tag{10.2}$$

We then sample $x \sim P_\theta$ in a two-step approach:

$$\begin{aligned} i &\sim \text{categorical}(p), \\ x &\sim \mathcal{N}(\mu_i, I). \end{aligned} \tag{10.3}$$

Here i is called the latent variable since we only observe x. Here we assume the covariances of the Gaussians to be identity, but they can also be parameters that are to be learned.

There are many other latent variables that could be defined via a similar generative process, such as Hidden Markov Models, Independent Component Analysis, which we will discuss later.

10.1.1 Warm-up: mixture of two Gaussians

We first study a simple case: the mixture of two Gaussians. In this case, $k = 2$, and we assume $p_1 = p_2 = \frac{1}{2}$. For simplicity, we also assume $\mu_1 = -\mu_2$, that is, the means of the two Gaussians are symmetric around the origin. To simplify our notation, let $\mu_1 = \mu$ and $\mu_2 = -\mu$. These assumptions yield the following model for x:

$$x \sim \frac{1}{2}\mathcal{N}(\mu, I) + \frac{1}{2}\mathcal{N}(-\mu, I). \tag{10.4}$$

To implement the moment method, we need to complete the following two tasks:

1. Estimate the moment(s) of x using empirical samples.
2. Recover parameters from the moment(s) of x.

The first moment of x is

$$\begin{aligned} M_1 &\triangleq \mathbb{E}[x] && (10.5)\\ &= \frac{1}{2}\mathbb{E}[x|i=1] + \frac{1}{2}\mathbb{E}[x|i=2] && (10.6)\\ &= \frac{1}{2}\mu + \frac{1}{2}(-\mu) && (10.7)\\ &= 0. && (10.8) \end{aligned}$$

Therefore, the first moment provides no information about μ. We compute the second moment as

$$\begin{aligned} M_2 &\triangleq \mathbb{E}[xx^\top] && (10.9)\\ &= \frac{1}{2}\mathbb{E}[xx^\top|i=1] + \frac{1}{2}\mathbb{E}[xx^\top|i=2] && (10.10) \end{aligned}$$

To compute these expectations, consider an arbitrary $Z \sim \mathcal{N}(\mu, I)$. Then,

$$\begin{aligned} \mathbb{E}[ZZ^\top] &= \mathbb{E}[Z]\,\mathbb{E}[Z]^\top + \mathrm{Cov}(Z) && (10.11)\\ &= \mu\mu^\top + I && (10.12) \end{aligned}$$

Recognizing that this second moment calculation is the same for both Gaussians in our mixture, we obtain:

$$\begin{aligned} M_2 &= \frac{1}{2}(\mu\mu^\top + I) + \frac{1}{2}(\mu\mu^\top + I) && (10.13)\\ &= \mu\mu^\top + I && (10.14) \end{aligned}$$

Since the second moment provides information about μ, we can complete the two tasks required for the moment method using the second moment.

If we had access to infinite data, then we can compute the exact second moment $M_2 = \mu\mu^\top + I$. Then, we can recover μ by evaluating the top eigenvector and eigenvalue of M_2.[1] The top eigenvector and eigenvalue of M_2 is $\bar{\mu} \triangleq \frac{\mu}{\|\mu\|_2}$ and $\|\mu\|_2^2 + 1$, respectively.

In practice, however, we do not have infinite data. In that case, we need to estimate the second moment by an empirical average.

$$\widehat{M_2} = \frac{1}{n}\sum_{i=1}^{n} x^{(i)}x^{(i)\top} \tag{10.15}$$

[1] This approach is known as the spectral method.

* Eigenvalue of $A = \mu\mu^\top$

$Ax = \mu(\mu^\top x)$

$\mathrm{Im}(A) \subseteq \mathrm{span}(\mu) \Rightarrow \mathrm{rank}(A) \le 1$

μ is an eigenvector of A with eigenvalue $\|\mu\|_2^2$

Adding I to the matrix shifts all eigenvalue by 1

We can then recover μ by evaluating the top egivenvector and eigenvalue of $\widehat{M}_2$. However, we need this algorithm to be robust to errors, i.e., similar estimates, $\widehat{M}_2$, of the second moment should yield similar estimates of μ. Fortunately, most algorithms we might use for obtaining the top eigenvector and eigenvalue are robust, so we can limit our attention to the infinite data case. Having outlined the moment method approach to the mixture of two Gaussians problem, we study a generalization of this problem in the sequel.

10.1.2 Mixture of Gaussians with more components via tensor decomposition

The general moment method for solving latent variable models is summarized by the following steps.

1. Compute $M_1 = \mathbb{E}[x]$, $M_2 = \mathbb{E}[xx^\top]$, $M_3 = \mathbb{E}[x \otimes x \otimes x]$, $M_4 = \cdots$. Note that $x \otimes x \otimes x$ is in $\mathbb{R}^{d \times d \times d}$ and $(x \otimes x \otimes x)_{ijk} = x_i \cdot x_j \cdot x_k$. For example, $M_{3,ijk} = \mathbb{E}[x_i x_j x_k]$.
2. Design as algorithm $A(M_1, M_2, M_3, \dots)$ that outputs θ.
3. Show that A is robust to errors in our moment estimates, i.e., we apply A to $(\widehat{M}_1, \widehat{M}_2, \widehat{M}_3, ...)$ in reality.

In the sequel, we instantiate this paradigm for mixtures of k Gaussians ($k \geq 3$).

For the simplicity of demonstrating the idea, we assume $p_1 = \cdots = p_k = \frac{1}{k}$, i.e. $i \overset{\text{unif}}{\sim} [k]$, and $x \sim \mathcal{N}(\mu_i, I)$. Equivalently,

$$x \sim \frac{1}{k} \sum_{i=1}^{k} \mathcal{N}(\mu_i, I). \tag{10.16}$$

In this example, we only describe steps (1) and (2) in the general algorithm described above.

We first evaluate the first and second moments. The first moment follows from

$$\begin{aligned} M_1 &= \mathbb{E}[x] && (10.17)\\ &= \sum_{i=1}^{k} \frac{1}{k} \mathbb{E}[x|i] && (10.18)\\ &= \frac{1}{k} \sum_{i=1}^{k} \mu_i, && (10.19) \end{aligned}$$

and the second moment is computed as

$$\begin{aligned} M_2 &= \mathbb{E}[xx^\top] && (10.20)\\ &= \sum_{i=1}^{k} \frac{1}{k} \mathbb{E}[xx^\top|i] && (10.21)\\ &= \sum_{i=1}^{k} \frac{1}{k} (\mu_i \mu_i^\top + I) && (10.22)\\ &= \frac{1}{k} \sum_{i=1}^{k} \mu_i \mu_i^\top + I. && (10.23) \end{aligned}$$

Second moments do not suffice

Can we recover $\mu = (\mu_1, ..., \mu_k)$ from $\frac{1}{k}\sum_{i=1}^{k} \mu_i$ and $\frac{1}{k}\sum_{i=1}^{k} \mu_i \mu_i^\top$? Unfortunately, in most of the cases when $k \geq 3$, the first and second moments are not sufficent to recover μ.

One reason is the so-called "missing rotation information" problem. Let

$$U = \begin{bmatrix} | & & | \\ \mu_1 & \cdots & \mu_k \\ | & & | \end{bmatrix} \in \mathbb{R}^{d \times k} \tag{10.24}$$

denote the matrix we aim to recover. Then, consider some rotation matrix $R \in \mathbb{R}^{k \times k}$. We consider U versus UR:

$$\frac{1}{k}\sum_{i=1}^{k} \mu_i \mu_i^\top = \frac{1}{k} UU^\top \tag{10.25}$$

$$= \frac{1}{k}(UR)(UR)^\top \qquad (RR^\top = I) \tag{10.26}$$

This result proves that the second moment is invariant to rotations. To prove a similar claim for the first moment, we also constrain our choice of R such that

$$R \cdot \vec{1} = \vec{1} \tag{10.27}$$

Then,

$$\frac{1}{k}\sum_{i=1}^{k} \mu_i = \frac{1}{k} U \cdot \vec{1} \tag{10.28}$$

$$= \frac{1}{k} UR \cdot \vec{1} \tag{10.29}$$

Therefore, the first and second moments of U and UR are indistinguishable, and we must consider the third moment in order to identify U.

Computing the third moment

The third moment is the tensor $\mathbb{E}[x \otimes x \otimes x] \in \mathbb{R}^{d \times d \times d}$. To express this expectation in terms of tractable quantities, we can condition on the Gaussian observed and average:

$$\mathbb{E}[x \otimes x \otimes x] = \frac{1}{k}\sum_{i=1}^{k} \mathbb{E}[x \otimes x \otimes x \mid i] \tag{10.30}$$

Each term in the sum now corresponds to the third moment for some multivariate Gaussian. Fortunately, Lemma 10.1 suggests a formula for estimating its value.

Lemma 10.1. *Suppose $z \in \mathcal{N}(v, I)$. Then,*

$$\mathbb{E}[z \otimes z \otimes z] = v \otimes v \otimes v + \sum_{l=1}^{d} \mathbb{E}[z] \otimes e_l \otimes e_l + \sum_{l=1}^{d} e_l \otimes \mathbb{E}[z] \otimes e_l + \sum_{l=1}^{d} e_l \otimes e_l \otimes \mathbb{E}[z] \tag{10.31}$$

where $e_1, \ldots, e_d$ denote the canonical basis vectors.

This lemma suggests that we can compute $v \otimes v \otimes v$ from a linear combination of $\mathbb{E}[z \otimes z \otimes z]$ and $\mathbb{E}[z]$. Also note that $\mathbb{E}[z] = v$. .

Proof. We compute the third moment element-wise. That is,

$$(\mathbb{E}[z \otimes z \otimes z])_{ijk} = \mathbb{E}[z_i z_j z_k] \tag{10.32}$$

$$= \mathbb{E}[(v_i + \xi_i) \cdot (v_j + \xi_j) \cdot (v_k + \xi_k)] \qquad (z = v + \xi, \xi \sim \mathcal{N}(0, I)) \tag{10.33}$$

$$= v_i v_j v_k + \underbrace{\mathbb{E}[v_i v_j \xi_k] + \mathbb{E}[v_i \xi_j v_k] + \mathbb{E}[\xi_i v_j v_k]}_{=0} \quad \mathbb{E}[v_i v_j \xi_k] = v_i v_j \mathbb{E}[\xi_k] = 0$$

$$+ \mathbb{E}[v_i \xi_j \xi_k] + \mathbb{E}[v_j \xi_i \xi_k] + \mathbb{E}[v_k \xi_i \xi_j] + \mathbb{E}[\xi_i \xi_j \xi_k] \quad \mathbb{E}[\xi_i \xi_k] = \delta_{ik} \tag{10.34}$$

$$\begin{cases} \mathbb{E}[\xi_i \xi_k] = \mathbb{E}[\xi_i]\mathbb{E}[\xi_k] = 0 & i \neq k \\ \mathrm{Var}(\xi_i) + \mathbb{E}[\xi_i]^2 = 1 + 0 = 1 & i = k \end{cases}$$

To explicitly compute the last four terms in (10.34), we note that:

$$\mathbb{E}[\xi_i \xi_k] = \begin{cases} 0 & \text{if } i \neq k \\ \mathbb{E}[\xi_i^2] = 1 & \text{if } i = k \end{cases} \tag{10.35}$$

and that for any choice of i, j, and k,

$$\mathbb{E}[\xi_i \xi_j \xi_k] = 0. \tag{10.36}$$

(Symmetrical distribution)
Another proof might use MGF

Therefore,

$$(\mathbb{E}[z \otimes z \otimes z])_{ijk} = v_i v_j v_k + v_i \mathbf{1}[j = k] + v_j \mathbf{1}[i = k] + v_k \mathbf{1}[i = j] \tag{10.37}$$

Since $(\sum_{l=1}^d v \otimes e_l \otimes e_l)_{ijk} = \sum_{l=1}^d v_i e_{lj} e_{lk} = v_i \mathbb{I}(j = k)$, we have proven that:

$$\mathbb{E}[z \otimes z \otimes z] = v \otimes v \otimes v + \sum_{l=1}^d v \otimes e_l \otimes e_l + \sum_{l=1}^d e_l \otimes v \otimes e_l + \sum_{l=1}^d e_l \otimes e_l \otimes v. \tag{10.38}$$

□

We can now apply Lemma 10.1 to compute the third moment of the mixture of k Gaussians. In particular,

$$\mathbb{E}[x \otimes x \otimes x] = \frac{1}{k} \sum_{i=1}^k \mathbb{E}[x \otimes x \otimes x \mid i] \tag{10.39}$$

↓ Lemma 10.1

$$= \frac{1}{k} \sum_{i=1}^k \left(\mu_i \otimes \mu_i \otimes \mu_i + \sum_{l=1}^d \mu_i \otimes e_l \otimes e_l + \sum_{l=1}^d e_l \otimes \mu_i \otimes e_l + \sum_{l=1}^d e_l \otimes e_l \otimes \mu_i \right) \tag{10.40}$$

$$= \frac{1}{k} \sum_{i=1}^k \mu_i \otimes \mu_i \otimes \mu_i + \sum_{l=1}^d \frac{1}{k} \left(\sum_{i=1}^k \mu_i \right) \otimes e_l \otimes e_l + \sum_{l=1}^d e_l \otimes \frac{1}{k} \left(\sum_{i=1}^k \mu_i \right) \otimes e_l$$
$$+ \sum_{l=1}^d e_l \otimes e_l \otimes \frac{1}{k} \left(\sum_{i=1}^k \mu_i \right) \tag{10.41}$$

$$= \frac{1}{k} \sum_{i=1}^k \mu_i \otimes \mu_i \otimes \mu_i + \sum_{l=1}^d \mathbb{E}[x] \otimes e_l \otimes e_l + \sum_{l=1}^d e_l \otimes \mathbb{E}[x] \otimes e_l + \sum_{l=1}^d e_l \otimes e_l \otimes \mathbb{E}[x] \tag{10.42}$$

$$\tag{10.43}$$

For notational convenience, let

$$a^{\otimes 3} \triangleq a \otimes a \otimes a. \tag{10.44}$$

So far, we have shown how to compute $\frac{1}{k} \sum_{i=1}^k \mu_i^{\otimes 3}$ from $\mathbb{E}[x^{\otimes 3}]$ and $\mathbb{E}[x]$. In the sequel, we will formalize the remaining problem, recovering $\{\mu_i\}_{i=1}^k$ from $\frac{1}{k} \sum_{i=1}^k \mu_i^{\otimes 3}$, as the tensor decomposition problem, and discuss efficient algorithms for it.

Tensor decomposition Recovering the Gaussian means, $\{\mu_i\}_{i=1}^k$, from the third mixture moment, $\frac{1}{k} \sum_{i=1}^k \mu_i^{\otimes 3}$, is a special case of the general tensor decomposition problem. That problem is set up as follows: Assume that $a_1, \cdots a_k \in \mathbb{R}^d$ are unknown. Then, given $\sum_{i=1}^k a_i^{\otimes 3}$, our goal is to reconstruct the a_i vectors.

Before we present some standard results on tensor decomposition, we first describe some basic facts about tensors. Much like matrices, tensors have an associated rank. For example, $a \otimes b \otimes c$ is a rank-1 tensor. In general, the rank of a tensor T is the minimum k such that T can be decomposed as

$$T = \sum_{i=1}^k a_i \otimes b_i \otimes c_i. \tag{10.45}$$

for some $\{a_i\}_{i=1}^k, \{b_i\}_{i=1}^k, \{c_i\}_{i=1}^k$. Another difference between tensors and matrices is that the former objects do not have the typical rotational invariance. In particular, consider applying a right rotation $R \in \mathbb{R}^{k \times k}$ to the matrix

$$A = \begin{bmatrix} | & & | \\ a_1 & \cdots & a_k \\ | & & | \end{bmatrix} \in \mathbb{R}^{d \times k} \tag{10.46}$$

and get

$$\widetilde{A} = AR = \begin{bmatrix} | & & | \\ \tilde{a}_1 & \cdots & \tilde{a}_k \\ | & & | \end{bmatrix} \in \mathbb{R}^{d \times k} \tag{10.47}$$

Then,

$$\sum_{i=1}^k a_i a_i^\top = AA^\top = (AR) \cdot (AR)^\top = \sum_{i=1}^k \tilde{a}_i \tilde{a}_i^\top \tag{10.48}$$

However, there is no tensor analogue to the rotation invariance result above. But tensors do maintain an interesting (and useful) permutation invariance; that is, $T = \sum_{i=1}^k a_i^{\otimes 3}$ is invariant to permutations of the indices of a_i. Or in other words, let $P \in \mathbb{R}^{k \times k}$ be a permutation matrix suppose, and let

$$\widetilde{A} = AP = \begin{bmatrix} | & & | \\ \tilde{a}_1 & \cdots & \tilde{a}_k \\ | & & | \end{bmatrix} \in \mathbb{R}^{d \times k} \tag{10.49}$$

Then,

$$\sum_{i=1}^k a_i^{\otimes 3} = \sum_{i=1}^k \tilde{a}_i^{\otimes 3} \tag{10.50}$$

The lack of rotation invariance in the sense above and the existence of permutation invariance make tensor decomposition computationally challenging as well as powerful.

We now summarize some standard results regarding tensor decomposition for $T = \sum_{i=1}^k a_i^{\otimes 3}$. The results for decomposing the asymmetric version $T = \sum_{i=1}^k a_i \otimes b_i \otimes$ are largely analogous. We will not prove these results here.

0. In the most general case, recovering the a_i's from T is computationally hard. Any procedure will either fail to find a unique a_i or it fails to find a_i *efficiently*.

1. In the orthogonal case, i.e. $a_1, \ldots, a_k$ are orthogonal vectors, each a_i is a global maximizer of

$$\max_{\|x\|_2 = 1} T(x, x, x) = \sum_{i,j,k} T_{ijk} x_i x_j x_k \tag{10.51}$$

   We can heuristically think of a_i as eigenvectors of T and there exists an algorithm to compute a_i in poly-time.

2. In the independent case, i.e. $a_1, \ldots, a_k$ are linearly independent. Jennrich's algorithm can be used to efficiently recover $\{a_i\}_{i=1}^k$.

Results 1 and 2 above both involve the so-called "under-complete" case ($k \leq d$), e.g., when the number of Gaussians in the mixture is smaller than the dimension of the data. Next, we describe certain overcomplete cases for which efficient tensor decomposition is possible.

Tensor Decomposition (Stanford CS369H)

1. Tensor Rank

Def. $\text{rank}(T) = \min\{r : T = \sum_{i=1}^{r} a_i \otimes b_i \otimes c_i\}$

Ex. The rank of a tensor $T \in \mathbb{R}^{n \times n \times n}$ is trivially at most n^3.

Can obtain a better upper bound of n^2. How?

Slice T into n matrices of size $n \times n$!

$$T = \sum_{k=1}^{n} \sum_{j=1}^{n} u_{k,j} \otimes v_{k,j} \otimes e_k$$

① In general, rank $\Omega(n^2)$ tensor decomposition is NP-hard.
② For tensors of rank up to $3n/2$, reasonable conditions for having a unique decomposition are known.
③ Efficient algorithms are known for rank up to n.

2. Jenrich's algorithm

Let T be a tensor that can be decomposed as $T = \sum_{i=1}^{r} a_i^{\otimes 3}$, where the a_i's are orthogonal.

① Pick a random vector $v \in \mathbb{R}^n$.

Compute $M := Tv = \sum_{i=1}^{r} (a_i \otimes a_i) \langle a_i, v \rangle$

Since the a_i's are orthogonal, considering $u_i = a_i$, $\sigma_i = \langle a_i, v \rangle$

$M = \sum_{i=1}^{r} \sigma_i u_i u_i^T$ is exactly same as SVD!

Moreover, $P(\langle a_i, v \rangle = \langle a_j, v \rangle) = 0$ since v is random.
With high probability, all singular values are different.

② Use SVD to compute

$$M = \sum_{i=1}^{r} \lambda_i (w_i \otimes w_i)$$

③ Find coefficients $\alpha_1, \cdots \alpha_r$ s.t. $M = \sum_{i=1}^{r} \alpha_i w_i^{\otimes 3}$

* $a^{\otimes 3} = (\lambda a) \otimes (\lambda a) \otimes (\lambda a) \cdot \lambda^{-3}$

Ex. $a_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $a_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

$T = a_1^{\otimes 3} + a_2^{\otimes 3} \Rightarrow T_{111} = 1,\ T_{222} = 1,\ \text{else: } 0$

Sol. $v = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$

$M_{jk} = \sum_{l=1}^{2} T_{jkl} v_l$

$M_{11} = T_{111} v_1 + T_{112} v_2 = 1 \cdot 2 + 0 \cdot (-1) = 2$

$M_{22} = T_{221} v_1 + T_{222} v_2 = 0 \cdot 2 + 1 \cdot (-1) = -1$

$M_{12} = M_{21} = 0$

$M = \begin{pmatrix} 2 & 0 \\ 0 & -1 \end{pmatrix}$

$\Rightarrow w_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, w_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \alpha_1 = 1, \alpha_2 = 1$

3. Suppose $a_1^{\otimes 2}, \ldots, a_k^{\otimes 2}$ are independent for $k \leq d^2$. Then, applying Result 2, we can recover a_i from $\sum_{i=1}^k (a_i^{\otimes 2})^{\otimes 3} = \sum_{i=1}^k (a_i^{\otimes 6}) \in \mathbb{R}^{d^6}$.
4. Excluding an algebraic set of measure 0, we can use the FOOBI algorithm to recover a_i from the fourth-order tensor $\sum_{i=1}^k a_i^{\otimes 4}$ when $k \leq d^2$. A robust version of the FOOBI algorithm is described in Ma et al. 2016.
5. Assume a_i are *randomly* generated unit vectors. Then, for the third order tensor, k can be large as $d^{1.5}$ Ma et al. 2016, Schramm and Steurer 2017.

In summary, the moment method is a recipe in which we first compute high order moments (i.e. tensors), assume that these estimates are noiseless, and decompose these tensors to recover the latent variables. Though we do not discuss these results here, there is an extensive literature analyzing the robustness of the moment method to error in the moment estimates. Last, we note that even though we only explicitly analyze the mixture of Gaussians model here, latent variable models amenable to analysis by the moment method include ICA, Hidden Markov Models, topic models, etc.

10.2 Graph Decomposition and Spectral Clustering

Introduced by Shi and Malik 2000 and Ng et al. 2001, spectral clustering learns a model for the data points using a *pairwise* notion of similarity. Formally, assume that we are given n data points $x^{(1)}, \ldots, x^{(n)}$ as well as a similarity matrix $G \in \mathbb{R}^{n \times n}$ such that

$$G_{ij} = \rho(x^{(i)}, x^{(j)}) \tag{10.52}$$

where ρ is some measure of similarity that assigns larger values to more similar pairs of points.

For example, $x^{(i)}$ could denote images for which $\rho(x^{(i)}, x^{(j)})$ measures the semantic similarity. Alternatively, $x^{(i)}$ might be users of a social network and $\rho(x^{(i)}, x^{(j)}) = 1$ if $x^{(i)}$ and $x^{(j)}$ are friends (hence usually share similar interests / jobs / $\cdots$).

We note that in moment methods, $\mathbb{E}[xx^\top]$ captures pairwise information between coordinates / dimensions, whereas matrix G here captures pairwise information between datapoints.

Our goal is to cluster the data points by viewing G as a graph. For instance, in the social network example, we can naturally view G as the adjacency matrix of an *unweighted* graph, where $G_{ij} \in \{0, 1\}$ defines the edges. Then, the clustering problem is to partition the graph into distinct neighborhoods, i.e., components that are as separate from each other as possible. As we will see repeatedly in the sequel, the eigendecomposition of G is closely related to this graph paritioning / clustering problem.

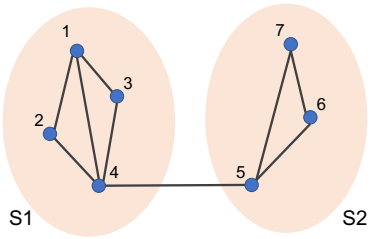

Figure 10.1: A demonstration of graph partitioning. Sets S_1 and S_2 form a good partition of the graph since there's only one edge between them.

10.2.1 Stochastic block model

In the stochastic block model (SBM), G is assumed to be generated randomly from two hidden communities. Formally,

$$\{1, \cdots n\} = S \cup \bar{S}, \tag{10.53}$$

where S and $\bar{S}$ partition $[n]$. Assume $|S| = \frac{n}{2}$. We then assume the following generative model for G. If $i, j \in S$ or $i, j \in \bar{S}$, then

$$G_{ij} = \begin{cases} 1 & \text{w.p. } p \\ 0 & \text{w.p. } 1-p \end{cases}. \tag{10.54}$$

Otherwise, for i and j in distinct components,

$$G_{ij} = \begin{cases} 1 & \text{w.p. } q \\ 0 & \text{w.p. } 1-q \end{cases} \tag{10.55}$$

for $p > q$ (i.e., more likely to be connected if from the same group). For instance, S and $\bar{S}$ could mean two companies, and $i \in [n]$ is a user of a social network. Two users i, j are more likely to know each other if they are in the same company.

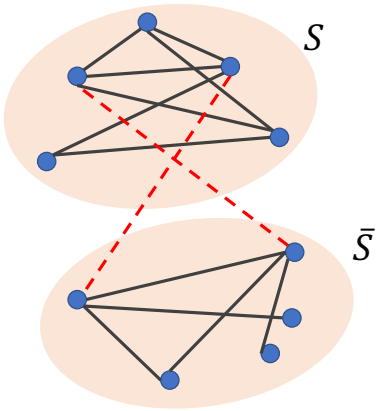

Figure 10.2: A graph generated by the stochastic block model with $p = \frac{2}{3}$ and $q = \frac{1}{5}$.

Our goal is then to recover S and $\bar{S}$ from G; the primary tool we use towards this goal is the eigendecomposition of G.

In some trivial cases, it is not necessary to eigendecompose G to recover the two hidden communities. Suppose, for instance, that $p = 0.5$ and $q = 0$. Then, the graph represented by G will contain two connected components that correspond to S and $\bar{S}$.

As a warm-up to motivate our approach, we eigendecompose $\bar{G} = \mathbb{E}[G]$. Observe that

$$\bar{G}_{ij} = \begin{cases} p & \text{if } i, j \text{ from the same community} \\ q & \text{o.w.} \end{cases}. \tag{10.56}$$

from (10.54), (10.55)

It is then easy to see that $\bar{G}$ is a matrix of rank 2:

$$\bar{G} = \left[\begin{array}{c|c} p \cdots p & q \cdots q \\ \vdots & \vdots \\ p \cdots p & q \cdots q \\ \hline q \cdots q & p \cdots p \\ \vdots & \vdots \\ q \cdots q & p \cdots p \end{array}\right]. \tag{10.57}$$

Lemma 10.2. *Let $\bar{G} = \mathbb{E}[G]$ for the stochastic block model. Then, letting $u_i(A)$ denote the i-th eigenvector of the matrix A,*

$$u_1(\bar{G}) = \vec{1} \tag{10.58}$$

$$u_2(\bar{G}) = [\underbrace{1, \dots, 1}_{|S|}, \underbrace{-1, \dots, -1}_{|\bar{S}|}]^\top \tag{10.59}$$

where $u_2(\bar{G})$ has $|S|$ entries of 1 and $|\bar{S}|$ entries of -1.

Proof. We begin by directly proving (10.58).

$$\bar{G} \cdot \vec{1} = \begin{bmatrix} \frac{pn}{2} + \frac{qn}{2} \\ \vdots \\ \frac{pn}{2} + \frac{qn}{2} \end{bmatrix} \tag{10.60}$$

$$= \frac{p+q}{2} \cdot n \cdot \vec{1}. \tag{10.61}$$

More generally, $\vec{1}$ is the top eigenvector for any matrix with fixed row sum or any graph with uniform degree (i.e., regular graph).

Next, we prove (10.59). Let

$$G' = \left[\begin{array}{c|c} \begin{matrix} r \cdots r \\ \vdots \\ r \cdots r \end{matrix} & 0 \\ \hline 0 & \begin{matrix} r \cdots r \\ \vdots \\ r \cdots r \end{matrix} \end{array}\right] \tag{10.62}$$

for $r = p - q$. To precisely define G', we note that G' is block diagonal with two blocks of size $|S|$ and $|\bar{S}|$, respectively. Then,

$$\bar{G} = \vec{1}\vec{1}^\top q + G'. \tag{10.63}$$

Thus,

$$G' \cdot u = \left[\begin{array}{c|c} \begin{matrix} r \cdots r \\ \vdots \\ r \cdots r \end{matrix} & 0 \\ \hline 0 & \begin{matrix} r \cdots r \\ \vdots \\ r \cdots r \end{matrix} \end{array}\right] \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \\ -1 \\ \vdots \\ -1 \end{bmatrix} = r \cdot \frac{n}{2} \cdot u. \tag{10.64}$$

Then, because $u \perp \vec{1}$, we can combine (10.63) and (10.64) to obtain

$$\bar{G} \cdot u = G' \cdot u = r \cdot \frac{n}{2} \cdot u = \frac{p-q}{2} \cdot n \cdot u \tag{10.65}$$

Thus, u has eigenvalue $\frac{p-q}{2} \cdot n$. □

Remark 10.3. Lemma 10.2 shows that

$$\bar{G} = \frac{p+q}{2} \cdot \vec{1}\vec{1}^\top + \frac{p-q}{2} \cdot uu^\top. \tag{10.66}$$

$A = \sum_{i=1}^{k} \lambda_i \frac{v_i v_i^\top}{\|v_i\|^2}$ (spectral decomposition)

More generally, when we have more than two clusters in the graph, G' is block diagonal with more than two blocks. In this setting, the eigenvectors will still align with the blocks. We illustrate this below for a generic block diagonal matrix. Let

$$A = \left[\begin{array}{c|c|c} \begin{matrix} 1 \cdots 1 \\ \vdots \\ 1 \cdots 1 \end{matrix} & 0 & 0 \\ \hline 0 & \begin{matrix} 1 \cdots 1 \\ \vdots \\ 1 \cdots 1 \end{matrix} & 0 \\ \hline 0 & 0 & \begin{matrix} 1 \cdots 1 \\ \vdots \\ 1 \cdots 1 \end{matrix} \end{array}\right] \tag{10.67}$$

Then, the three eigenvectors of A are

$$\begin{bmatrix} 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \tag{10.68}$$

Furthermore, the rows of the matrix formed by the three eigenvectors given by (10.68) clearly give the cluster IDs of the vertices in G. Note also that permutations of A will result in equivalent permutations in the coordinates of each of the three eigenvectors.

Next, we relate this observation to the result in Lemma 10.2. While there are no negative values in the eigenvectors given in (10.68), we observe that any linear combination of eigenvectors is also an eigenvector, so recovering a solution that look more like (10.59) is straightforward. Indeed, taking linear combinations of the eigenvectors defined above shows that there is an alternative eigenbasis that includes the all-ones vector, $\vec{1}$. How for this choice of A, the all-ones vector is not the unique top eigenvector. For that to be the case, we require background noise in $\bar{G}$.

vector with largest eigenvalue

In reality, we only observe G. In the sequel, we will show that in terms of the spectrum, $G \approx \mathbb{E}[G]$. Formally, we will leverage earlier concentration results to prove that $\|G - \mathbb{E}[G]\|_{\text{op}}$ is small. Concretely, then,

operator norm
$\|T\| = \sup_{\|v\|=1} \|T(v)\|$

$$G = (G - \mathbb{E}[G]) + \mathbb{E}[G] \tag{10.69}$$

$$= (G - \mathbb{E}[G]) + \frac{p+q}{2} \cdot \vec{1}\vec{1}^\top + \frac{p-q}{2} \cdot uu^\top \tag{10.70}$$

Rearranging, we obtain that:

$$G - \frac{p+q}{2} \cdot \vec{1}\vec{1}^\top = (G - \mathbb{E}[G]) + \frac{p-q}{2} \cdot uu^\top \tag{10.71}$$

We then hope that $G - \mathbb{E}[G]$ is a small perturbation, so that the top eigenvector of $G - \frac{p+q}{2} \cdot \vec{1}\vec{1}^\top$ is close to u. Namely, it suffices to show that

$$\|G - \mathbb{E}[G]\|_{\text{op}} \ll \left\| \frac{p-q}{2} \cdot uu^\top \right\|_{\text{op}}. \tag{10.72}$$

We will start by proving the following lemma.

Lemma 10.4. *With high probability,*

$$\|G - \mathbb{E}[G]\|_{\text{op}} \leq O(\sqrt{n \log n}) \,. \tag{10.73}$$

Note that this concentration inequality is different from the ones we have seen in the course so far because both G and $\mathbb{E}[G]$ are matrices, not scalars. Our goal will be to turn the quantity on the LHS into something that we are familiar with.

Proof. The key idea is that we can use uniform convergence, after noting that

$$\|G - \mathbb{E}[G]\|_{\text{op}} = \max_{\|v\|_2=1} \left|v^\top (G - \mathbb{E}[G])v\right| \qquad \because G - \mathbb{E}[G] \text{ is symmetric} \tag{10.74}$$

$$= \max_{\|v\|_2=1} \left|v^\top G v - v^\top \mathbb{E}[G] v\right| \tag{10.75}$$

$$= \max_{\|v\|_2=1} \left| \sum_{i,j\in[n]} v_i v_j G_{ij} - \mathbb{E}\left[\sum_{i,j\in[n]} v_i v_j G_{ij}\right] \right| \,. \tag{10.76}$$

Now, the quantity inside the max is the difference between the sum of independent random variables and their expectation, which we are familiar with. We can use brute force discretization to deal with the max. First, note that for a fixed v with $\|v\|_2 = 1$, we can use Hoeffding's inequality to find that

$$\Pr\left(\left| \sum_{i,j\in[n]} v_i v_j G_{ij} - \mathbb{E}\left[\sum_{i,j\in[n]} v_i v_j G_{ij}\right] \right| \geq \epsilon \right) \leq \exp(-\frac{\epsilon^2}{2}) \,. \tag{10.77}$$

Then, we choose $\epsilon = O(\sqrt{n \log n})$, take a discretization of the unit ball with granularity $1/n^{O(1)}$ (which yields a cover of cardinality $\exp(n \log n)$), and take a union bound over this discretized set to achieve the desired result. □

Remark 10.5. Comparing this bound to $\frac{p-q}{2} \cdot n$, we can deduce that if $p - q \gg \frac{\sqrt{\log n}}{\sqrt{n}}$, then we can recover the vector u approximately. Via a post-processing step that we do not discuss here, u can actually be recovered exactly.

10.2.2 Clustering the worst-case graph

Given a graph $G(V, E)$ where V denotes the set of vertices and E the set of edges, we define the *conductance* of a component S as

$$\phi(S) \triangleq \frac{|E(S, \bar{S})|}{\text{Vol}(S)} \tag{10.78}$$

where $E(S, \bar{S})$ is the total number of edges between S and $\bar{S}$, and $\text{Vol}(S)$ is the total number of edges connecting to S. To be precise, let A be the adjacency matrix of G,

$$E(S, \bar{S}) = \sum_{i\in S, j\in\bar{S}} A_{ij} \tag{10.79}$$

$$\text{Vol}(S) = \sum_{i\in S, j\in[n]} A_{ij} \,. \tag{10.80}$$

Intuitively, conductance captures how separated S and $\bar{S}$ are.

Since $\text{Vol}(S) \geq E(S, \bar{S})$, it follows that $\phi(S) \leq 1$. Next, observe that $\text{Vol}(S) + \text{Vol}(\bar{S}) = \text{Vol}(V)$. Then, if $\text{Vol}(S) \leq \text{Vol}(V)/2$, it must be the case that $\text{Vol}(S) \leq \text{Vol}(\bar{S})$ and therefore $\phi(S) \geq \phi(\bar{S})$. In some sense, thus suggests that the conductance of a set $\bar{S}$ with volume larger than $\text{Vol}(V)/2$ could be misleading, because

the conductance of the other part could be larger. Therefore, typically people only consider the conductance of a smaller part of the partition.

Next, we define $\phi(G)$ to be the *sparsest cut* of G:

$$\phi(G) = \min_{S:\mathrm{Vol}(S)\le \mathrm{Vol}(V)/2} \phi(S) \,. \tag{10.81}$$

One may wonder why we need to normalize by $\mathrm{Vol}(S)$ in the definition of conductance. The reason is that $E(S,\bar{S})$ itself is typically minimized when S is small. Thus, without this normalization, the sparsest cut would not be very meaningful. For instance, suppose the graph G contains N nodes and can be divided into two halves each containing $N/2$ nodes, and every node is connected to all the other nodes in the same half, but is connected to only 2 nodes in the other half (as shown in Figure 10.3). Then, we can consider two subsets S_1 and S_2, where S_1 contains half the nodes, and S_2 contains two nodes in the same half. It's easy to see that $E(S_1,\bar{S}_1) = \frac{N}{2}\cdot 2 > E(S_2,\bar{S}_2) = \frac{N}{2}$. However, the conductance of S_1 is $\phi(S_1) = \frac{E(S_1,\bar{S}_1)}{\mathrm{Vol}(S_1)} = \frac{\frac{N}{2}\cdot 2}{\frac{N}{2}\cdot(\frac{N}{2}-1)+\frac{N}{2}\cdot 2} \approx \frac{4}{N}$, whereas the conductance of S_2 is $\phi(S_2) = \frac{E(S_2,\bar{S}_2)}{\mathrm{Vol}(S_2)} = \frac{\frac{N}{2}}{N+2} \approx \frac{1}{2}$. Thus, when n is large, S_1 is a sparser cut than S_2 under $\phi(\cdot)$.

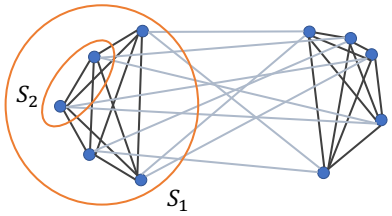

Figure 10.3: A demonstration of the sparsest cut. S_1 is a sparser cut than S_2.

Our goal is to find an approximate sparsest cut $\hat{S}$ such that $\phi(\hat{S}) \approx \phi(G)$.[2] Our main technique is eigendecomposition or spectral clustering, though in the literature much more advanced and better algorithms have been proposed, e.g., the famous ARV algorithm [Arora et al. 2009]. Let $d_i = \mathrm{Vol}(\{i\})$ be the degree of node i, and let $D = \mathrm{diag}(\{d_i\})$ be the diagonal matrix that contains the degrees d_i as entries. Furthermore, let

$$\bar{A} = D^{-\frac{1}{2}} A D^{\frac{1}{2}} \tag{10.82}$$

be the normalized adjacency matrix. This is equivalent to normalizing each element A_{ij} of the adjacency matrix by $\frac{1}{\sqrt{d_i d_j}}$ (i.e., $\bar{A}_{ij} = \frac{A_{ij}}{\sqrt{d_i d_j}}$). In most cases, it suffices to starting with considering G as a regular graph (whose degrees are all the same), because the proof for regular graph can oftentimes extend to general graph easily. Assuming G is a κ-regular graph, i.e. $d_i = \kappa$; then, this normalization simply scales A by $\frac{1}{\kappa}$.

Let $L = I - \bar{A}$ be the Laplacian matrix. Note that any eigenvector of L is also an eigenvector of $\bar{A}$. Suppose L has eigenvalues $\lambda_1 \le \ldots \le \lambda_n$ with corresponding eigenvectors $u_1 \ldots u_n$, then $\bar{A}$ has eigenvalues $1-\lambda_1 \ge \ldots \ge 1-\lambda_n$ with the same eigenvectors.

The following famous Cheeger's inequality relates the eigendecompositions to the graph partitioning.

Theorem 10.6 (Cheeger's inequality)**.** *The second eigenvalue of L, namely λ_2, is related to the sparsest cut $\phi(G)$ as follows:*

$$\frac{\lambda_2}{2} \le \phi(G) \le \sqrt{2\lambda_2} \,. \tag{10.83}$$

Moreover, we can find $\hat{S}$ such that $\phi(\hat{S}) \le \sqrt{2\lambda_2} \le 2\sqrt{\phi(G)}$ efficiently by rounding the second eigenvector. Suppose $u_2 = [\beta_1 \cdots \beta_n]^\top \in \mathbb{R}^n$ is the second eigenvector of L. Then we can choose a threshold $\tau = \beta_i$ and consider $\hat{S}_i = \{j \in [n] : \beta_j \le \tau\}$. At least one such $\hat{S}_i$ satisfies $\phi(\hat{S}_i) \le 2\sqrt{\phi(G)}$.

[2] Finding the exact sparsest cut is a NP-hard problem.

Note that this can be viewed as a general but weaker version of the theorem that we proved for stochastic block model. There is no randomized assumption; the conclusion is weaker than those for SBM; also the rounding algorithm to recover the cluster is also more complicated—one has to try multiple thresholding instead of using threshold 1/2.

We will refer the readers to Chung [2007] for the proof of the theorem. Here below we give a few basic lemmas that help build up intuitions on why eigendecompositions relate to graph decomposition.

First, one might wonder why this algorithm uses the second eigenvector of $\bar{A}$, but not the first eigenvector. As we have seen in the SBM case, the first eigenvector captures the background in some sense. Here for general graph, we see the same phenomenon. The top eigenvector is generally not that interesting as it only captures the "background density" of the graph. For instance, when A is κ-regular, $\vec{1}$ is the top eigenvector of A and is thus also the top eigenvector of $\bar{A} = \frac{1}{\kappa} \cdot A$. More generally, we have the following lemma:

Lemma 10.7. *The top eigenvector of $\bar{A}$ (respectively, the smallest eigenvector of L) is $u_1 = [\sqrt{d_1} \cdots \sqrt{d_n}]^\top$.*

Proof.

degree info

$$(\bar{A} \cdot u_1)_i = \sum_j \bar{A}_{ij} \sqrt{d_j} \tag{10.84}$$

$$= \sum_j \frac{A_{ij}}{\sqrt{d_i}\sqrt{d_j}} \sqrt{d_j} \tag{10.85}$$

$$= \frac{1}{\sqrt{d_i}} \sum_j A_{ij} \tag{10.86}$$

$$= \frac{d_i}{\sqrt{d_i}} = \sqrt{d_i}. \tag{10.87}$$

□

To understand why the eigenvectors of the Laplacian are related to the sparsest cut, we examine the quadratic form the Laplacian. In particular, we have the following lemma:

Lemma 10.8. *Let $v \in \mathbb{R}^N$ be a vector, L is the graph Laplacian. Then,*

$$v^\top L v = \frac{1}{2} \sum_{(i,j) \in E} \left(\frac{v_i}{\sqrt{d_i}} - \frac{v_j}{\sqrt{d_j}} \right)^2. \tag{10.88}$$

Proof.

$$v^\top L v = v^\top I v - v^\top \bar{A} v \tag{10.89}$$

$$= \sum_{i=1}^n v_i^2 - \sum_{i,j=1}^n v_i v_j \bar{A}_{ij} \tag{10.90}$$

$$= \sum_{i=1}^n v_i^2 - \sum_{i,j=1}^n v_i v_j \frac{A_{ij}}{\sqrt{d_i d_j}} \tag{10.91}$$

$$= \frac{1}{2} \cdot \left(2 \sum_{i=1}^n v_i^2 - 2 \sum_{(i,j) \in E} \frac{v_i}{\sqrt{d_i}} \cdot \frac{v_j}{\sqrt{d_j}} \right) \tag{10.92}$$

$$= \frac{1}{2} \sum_{(i,j) \in E} \left(\frac{v_i}{\sqrt{d_i}} - \frac{v_j}{\sqrt{d_j}} \right)^2. \tag{10.93}$$

$\sum_{(i,j)\in E} \frac{v_i^2}{d_i} = \sum_{i=1}^n d_i \cdot \frac{v_i^2}{d_i} = \sum_{i=1}^n v_i^2$

□

If G is κ-regular, then this becomes $v^\top Lv = \frac{1}{2\kappa}\sum_{(i,j)\in E}(v_i - v_j)^2$. Furthermore, suppose $v \in \{0,1\}$ is a binary vector with support $S = \text{supp}(v)$. Then,

$$\frac{1}{2\kappa}\sum_{(i,j)\in E}(v_i - v_j)^2 = \frac{1}{\kappa}E(S,\bar{S}) \tag{10.94}$$

$$= \frac{1}{\kappa}E(\text{supp}(v), \overline{\text{supp}}(v)) \ . \tag{10.95}$$

If $|\text{supp}(v)| \le n/2$, implying $\text{Vol}(S) \le \text{Vol}(V)/2$, then

$$\frac{v^\top Lv}{\|v\|_2^2} = \frac{\frac{1}{\kappa}E(S,\bar{S})}{\frac{1}{\kappa}\text{Vol}(S)} = \phi(S) \ . \tag{10.96}$$

The term $\frac{v^\top Lv}{\|v\|_2^2}$ is also called the *Rayleigh quotient*. This result nicely connects the abstract linear algebraic approach to the sparsest cut approach. Note that we only achieve an approximate sparsest cut because when we compute eigenvectors, we minimize the Rayleigh quotient *without any constraints on* v. By contrast, the sparsest cut minimizes the Rayleigh quotient subject to the constraint that $v \in \{0,1\}^n$. Proving Cheeger inequality essentially involves controlling the difference caused by real v vs binary v. We omit the proof of Cheeger's inequality because it's beyond the scope of this notes.

10.2.3 Applying spectral clustering

How do the ideas from the previous sections connect to our previous discussion of spectral clustering? Suppose that we have some raw data $x^{(1)} \cdots x^{(n)}$ that we'd like to group into k clusters. Ng et al. [2001] propose to define a weighted graph G such that each element

$$G_{ij} = \exp\left(-\frac{\|x^{(i)} - x^{(j)}\|_2^2}{2\sigma^2}\right) \tag{10.97}$$

represents a distance between two data points. Then, we compute the first k eigenvectors of the Laplacian L and arrange them into the columns of a matrix:

$$\begin{bmatrix} | & & | \\ u_1 & \cdots & u_k \\ | & & | \end{bmatrix} \in \mathbb{R}^{n\times k}. \tag{10.98}$$

The i-th row of this matrix (which we denote by v_i) is then a k-dimensional embedding of the i-th example. Note that this is usually a much lower-dimensional representation than the raw data. Finally, we can use k-means to cluster the embeddings $\{v_1, \ldots, v_n\}$.

Supplementary

In high dimensions, the issue with Ng et al. [2001]'s approach is that the training data points are all far away from each other. The Euclidean distance between points becomes meaningless, and so our definition of G does not make sense.

How do we solve this issue? HaoChen et al. [2021] propose a solution. They consider an infinite weighted graph $G(V,w)$, where w are the edge weights, and $V = \mathcal{X} \subseteq \mathbb{R}^n$ is the set of all possible data points. We define $w(x,x')$ to be large only if x and x' are very close in ℓ_2 distance. Now, the graph is more meaningful, because only data points that are small perturbations of each other have high connection weights. However, we do not have explicit access to this graph, and its eigenvectors are infinite-dimensional.

Now, suppose we have some eigenvector $u = (u_x)_{x\in\mathcal{X}}$. Rather than explicitly represent u_x, we represent u_x by $f_\theta(x)$ where f_θ is some parameterized model. Now, the challenge is to find θ such that $[f_\theta(x)]_{x\in\mathcal{X}}$ is the second smallest eigenvector of Laplacian of G. It turns out solving this problem gives a form of contrastive learning, which we will discuss in Section 10.3.2

Rayleigh Quotients (MIT 18.409)

Problem (constrained). Given a symmetrix matrix $A \in \mathbb{R}^{n \times n}$, find the extreme values of the associated quadratic form over the unit sphere in $\mathbb{R}^n$:

$$\max_{x \in \mathbb{R}^n} x^T A x \quad \text{subject to } \|x\|^2 = 1$$

Same Problem (unconstrained): $\max_{x \neq 0 \in \mathbb{R}^n} \frac{x^T A x}{x^T x}$: Rayleigh Constant

Thm. For any given symmetric matrix $A \in \mathbb{R}^{n \times n}$,

$$\max_{x \in \mathbb{R}^n : x \neq 0} \frac{x^T A x}{x^T x} = \lambda_{max} \quad (\text{when } x = \text{"largest" eigenvector of } A)$$

$$\min_{x \in \mathbb{R}^n : x \neq 0} \frac{x^T A x}{x^T x} = \lambda_{min} \quad (\text{when } x = \text{"smallest" eigenvector of } A)$$

Ex. $Q(x) = \frac{x_1^2 + 4x_2^2 + 4x_1x_2}{x_1^2 + x_2^2}$ corresponding to $A = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}$

- maximum value of $Q(x)$ is 5, achieved at $X_1 = \frac{1}{\sqrt{5}}(1,2)^T$
- minimum value is 0, achieved at $X_2 = \frac{1}{\sqrt{5}}(-2,1)^T$ (orthogonal to X_1)

Proof (Linear algebra approach).

Let $A = Q \Lambda Q^T$ be the spectral decomposition, where $Q = [q_1, \cdots, q_n]$ is orthogonal and $\Lambda = \text{diag}(\lambda_1, \cdots, \lambda_n)$ is diagonal with sorted diagonals from large to small. Then for any unit vector x,

$$x^T A x = x^T (Q \Lambda Q^T) x = (x^T Q) \Lambda (Q^T x) = y^T \Lambda y$$

where $y = Q^T x$ is also a unit vector:

$$\|y\|^2 = y^T y = x^T Q Q^T x = x x^T = 1$$

So the original problem becomes

$$\max_{y \in \mathbb{R}^n : \|y\|=1} \; y^T \Lambda y$$

write $y = (y_1, \cdots, y_n)^T$. It follows that $y^T \Lambda y = \sum_{i=1}^{n} \lambda_i y_i^2$ (subject to $y_1^2 + \cdots + y_n^2 = 1$)

Because $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$, when $y_1^2 = 1, y_2^2 = \cdots = y_n^2 = 0$ (i.e. $y = \pm e_1$), $y^T \Lambda y = \lambda_1$ is the maximum.

$x = Qy = Q(\pm e_1) = \pm q_1$ is the maximizer

Proof 2 (Multi-variable Calculus approach).

① $L(x, \lambda) = x^T A x - \lambda(\|x\|^2 - 1)$

② Compute the partial derivatives $\frac{\partial L}{\partial x}, \frac{\partial L}{\partial \lambda}$ and set them equal to zero

Def (Generalized Rayleigh Quotient)

Let B be a PD matrix with same size as A, then the quantity

$\frac{x^T A x}{x^T B x}$ is called a generalized Rayleigh quotient.

* The generalized Rayleigh quotient problem

The extreme values λ of the generalized Rayleigh quotient satisfy

$$Av = \lambda B v \iff B^{-1} A v = \lambda v$$

Brief Proof. $y = B^{1/2} x$

$$x^T B x = x^T B^{1/2} B^{1/2} x = y^T y$$

Substitute $x = B^{-1/2} y$ into the numerator (which converts the problem to a regular Rayleigh quotient problem)

10.3 Self-supervised Learning

10.3.1 Pretraining / self-supervised learning / foundation model basics

Self-supervised learning is widely used for pretraining modern models. These large pretrained models are also called foundation models Bommasani et al. 2021. One simplified setup / workflow contains the following two stages:

Pretraining on unlabeled, massive data. Let $\{x^{(1)}, \cdots, x^{(n)}\}$ be a dataset where $x^{(i)} \in \mathbb{R}^d$ is sampled from some pretraining data distribution $x^{(i)} \sim P_{\text{pre}}$. The goal is to learn a pretrained model $f_\theta : \mathbb{R}^d \to \mathbb{R}^k$, where k is the dimension of features / representations / embeddings, and θ is the model parameter. This model can be learned by minimizing certain pretrained loss function: $\hat{L}_{\text{pre}}(\theta) = \hat{L}_{\text{pre}}(x^{(1)}, \cdots, x^{(n)}; \theta)$, which sometimes is of the form $\hat{L}_{\text{pre}}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell_{\text{pre}}(x^{(i)}; \theta)$. We use $\hat{\theta} = \text{argmin}_\theta \hat{L}_{\text{pre}}(\theta)$ to denote the parameter learned during pretraining.

Adaptation. During adaptation, we have access to a set of labeled downstream task examples $\{(x^{(1)}_{\text{task}}, y^{(1)}_{\text{task}}), \cdots, (x^{(n_{\text{task}})}_{\text{task}}, y^{(n_{\text{task}})}_{\text{task}})\}$, where usually $n_{\text{task}} \ll n$. One popular adapataion method is *linear probe*, which uses $f_{\hat{\theta}}(x)$ as features / representations / embeddings, and train a linear classifier on downstream tasks. Concretely, the prediction on data x is $w^\top f_{\hat{\theta}}(x)$, where w is the linear head learned from $\min_w \hat{L}_{\text{task}}(w) = \frac{1}{n_{\text{task}}}\sum_{i=1}^{n_{\text{task}}} \ell_{\text{task}}(y^{(i)}_{\text{task}}, w^\top f_{\hat{\theta}}(x^{(i)}_{\text{task}}))$. Another popular adaptation method is *finetuning*, which also updates the parameter θ. Concretely, one initializes from $\theta = \hat{\theta}$, and solve $\min_{\theta,w} \hat{L}_{\text{task}}(w, \theta) = \frac{1}{n_{\text{task}}}\sum_{i=1}^{n_{\text{task}}} \ell_{\text{task}}(y^{(i)}_{\text{task}}, w^\top f_{\theta}(x^{(i)}_{\text{task}}))$.

Why does pretraining on unlabeled data with an unsupervised (self-supervised) loss help a wide range of downstream prediction tasks? There are many open questions to be answered in this field. For instance, we may ask: (1) how pretraining helps label efficiency of downstream tasks, (2) why pretraining can give "universal" representations, and (3) why does pretraining provide robustness to distribution shift.

10.3.2 Analysis of contrastive learning

Let $\bar{X}$ be the set of all natural images in the image domain, $\bar{P}_{\bar{X}}$ be the distribution over $\bar{X}$. Contrastive learning learns f_θ such that representations of augmentations of the same image are close, whereas augmentations of random images are far away (as visualized in Figure 10.4).

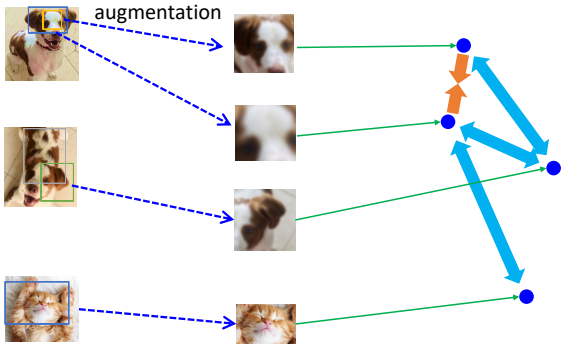

Figure 10.4: A demonstration of contrastive learning. Representations of augmentations of the same image are pulled close, whereas augmentations of random images are pushed far away.

Given a natural image $\bar{x} \in \bar{X}$, one can generate augmentations by random cropping, flipping, adding Gaussian noise or color transformation. We use $x \sim \mathcal{A}(\cdot|\bar{x})$ to denote the conditional distribution of augmentations given the natural image. For simplicity, we consider the case where Gaussian blurring is the

augmentation, we have

$$x \sim \mathcal{A}(\cdot|\bar{x}) \Leftrightarrow x = \bar{x} + \xi \qquad \xi \sim \mathcal{N}(0, \sigma^2 \mathcal{I}), \tag{10.99}$$

where $\|\xi\|_2 \approx \sigma\sqrt{d}$ should be $\ll \|\bar{x}\|$.

We say (x, x^+) is a *positive pair* if they are generated as follows: first sample $\bar{x} \sim \bar{P}_{\bar{X}}$, then sample $x \sim \mathcal{A}(\cdot|\bar{x})$ and $x^+ \sim \mathcal{A}(\cdot|\bar{x})$ independently. In other words, (x, x^+) are augmentations of the same natural image.

We say (x, x') is a *random pair / negative pair* if they are sampled as: first sample two natural images $\bar{x} \sim \bar{P}_{\bar{X}}$ and $\bar{x}' \sim \bar{P}_{\bar{X}}$, then sample augmentations $x \sim \mathcal{A}(\cdot|\bar{x})$ and $x' \sim \mathcal{A}(\cdot|\bar{x}')$.

The design principle for contrastive learning is to find θ such that $f_\theta(x)$, $f_\theta(x^+)$ are close, while $f_\theta(x), f_\theta(x')$ are far away Chen et al., 2020, Zbontar et al., 2021, He et al., 2020.

One example of contrastive learning is SimCLR Chen et al., 2020. Given B positive pairs $(x^{(1)}, x^{(1)+}), \cdots, (x^{(B)}, x^{(B)+})$, note that $(x^{(i)}, x^{(j)+})$ is a random pair if $i \neq j$, SimCLR defines the loss on the i-th pair as

$$\text{loss}_i = -\log \frac{\exp(f_\theta(x^{(i)})^\top f_\theta(x^{(i)+}))}{\exp(f_\theta(x^{(i)})^\top f_\theta(x^{(i)+})) + \sum_{j \neq i} \exp(f_\theta(x^{(i)})^\top f_\theta(x^{(j)+}))}. \tag{10.100}$$

Notice that $-\log \frac{A}{A+B}$ is decreasing in A but increasing in B, the loss above encourages $f_\theta(x^{(i)})^\top f_\theta(x^{(i)+})$ to be large, while $f_\theta(x^{(i)})^\top f_\theta(x^{(j)+})$ to be small.

In the rest of this section, we consider a variant of contrastive loss, proposed in HaoChen et al., 2021:

$$L(\theta) = -2 \mathop{\mathbb{E}}_{(x,x^+)\sim\text{positive}} f_\theta(x)^\top f_\theta(x^+) + \mathop{\mathbb{E}}_{(x,x')\sim\text{random}} \left(f_\theta(x)^\top f_\theta(x')\right)^2. \tag{10.101}$$

This contrastive loss follows the same design principle as other contrastive losses in the literature: suppose all representations have the same norm, then the first term encourages the representations of a positive pair to be closer while the second term encourages the representations of a random pair to be orthogonal to each other (hence far away). HaoChen et al., 2021 show that the loss function, though inspired by theoretical derivations, still perform competitively, nearly matching the SOTA methods.

We can also define the empirical loss on a set of tuples $(x^{(i)}, x^{+(i)}, x'^{(i)})$ sampled i.i.d. as follows: $\bar{x} \sim \bar{P}_{\bar{X}}, x^{(i)} \sim \mathcal{A}(\cdot|\bar{x}^{(i)}), x^{+(i)} \sim \mathcal{A}(\cdot|\bar{x}^{(i)})$, $\bar{x}' \sim \bar{P}_{\bar{X}}$, $x'^{(i)} \sim \mathcal{A}(\cdot|\bar{x}'^{(i)})$. The empirical loss is defined as

$$\hat{L}(\theta) = \sum_{i=1}^{n} \left[-2 f_\theta(x^{(i)})^\top f_\theta(x^{+(i)}) + \left(f_\theta(x^{(i)})^\top f_\theta(x'^{(i)})\right)^2 \right]. \tag{10.102}$$

Then the empirically learned parameter is $\hat{\theta} = \min_\theta \hat{L}(\theta)$.

Suppose the downatream task is binary classification with label set $\{1, -1\}$. We define the downstream loss as

$$\hat{L}_{\text{task}}(w, \theta) = \frac{1}{n_{\text{task}}} \sum_{i=1}^{n_{\text{task}}} \frac{1}{2} \left(y_{\text{task}}^{(i)} - w^\top f_\theta(x_{\text{task}}^{(i)})\right)^2. \tag{10.103}$$

We learn the linear head $\hat{w} = \text{argmin}_w \hat{L}_{\text{task}}(w, \hat{\theta})$, and the evaluate its performance on downstream population data:

$$L_{\text{task}}(\hat{w}, \hat{\theta}) = \mathbb{E}\left[\frac{1}{2}\left(y_{\text{task}} - \hat{w}^\top f_\theta(x_{\text{task}})\right)^2\right]. \tag{10.104}$$

Analysis pipeline. We give a summary of our analysis pipeline below. The key takeaway is that we only have to focus on the population distribution case (step 3).

0. Assume expressivity, i.e., assuming $\exists \theta^*$ such that $L(\theta^*)$ is sufficiently small (the details will be quantified later).

1. For large enough n (e.g., $n > \text{Comp}(\mathcal{F})/\epsilon^2$ where $\mathcal{F} = \{f_\theta\}$ is the function class, $\text{Comp}(\cdot)$ is some measure of complexity, ϵ is the target error), show that $\hat{L}(\theta) = L(\theta) \pm \epsilon$.

2. Let $\hat{\theta}$ be the parameter learned on empirical data. Since $\hat{L}(\hat{\theta}) = \min_\theta \hat{L}(\theta) \le \hat{L}(\theta^*) \le L(\theta^*) + \epsilon$, we have

$$\hat{L}(\hat{\theta}) \le \epsilon \Rightarrow L(\hat{\theta}) \le 2\epsilon \tag{10.105}$$

3. **Key step:** (infinite data case) We will prove a theorem (Theorem 10.12 below as a simplified version, or Theorem 3.8 of HaoChen et al. 2021) that shows if $L(\hat{\theta}) \le 2\epsilon$, then there exists w such that $L_{\text{task}}(\theta, w) \le \delta$, where δ is a function of ϵ and data distribution $\bar{P}$.

4. When we have enough downstream data $n_{\text{task}} \ge \text{poly}(k, \frac{1}{\epsilon'})$, for any θ, with high probability we have (via uniform convergence) that for any w, $\hat{L}_{\text{task}}(w, \theta) \approx L_{\text{task}}(w, \theta) \pm \epsilon'$.

5. Using the results in step 3 and step 4, we have $\hat{L}_{\text{task}}(\hat{w}, \hat{\theta}) = \min_w \hat{L}_{\text{task}}(w, \hat{\theta}) \le \min_w L_{\text{task}}(w, \hat{\theta}) + \epsilon' \le \delta + \epsilon'$. Thus, the final evaluation loss on the downstream task is $L_{\text{task}}(\hat{w}, \hat{\theta}) \le \hat{L}_{\text{task}}(\hat{w}, \hat{\theta}) + \epsilon' \le \delta + 2\epsilon'$.

Key step: the case with population pretraining and downstream data. We will now dive into the analysis of step 3, as all the other steps are from standard concentration inequalities. Recall that

$$L(\theta) = -2 \mathop{\mathbb{E}}_{(x,x^+)} f_\theta(x)^\top f_\theta(x^+) + \mathop{\mathbb{E}}_{(x,x')} \left(f_\theta(x)^\top f_\theta(x')\right)^2. \tag{10.106}$$

As expected, the analysis requires structural assumptions on the data. In particular, we will use the graph-theoretic language to describe the assumptions on population data. Let X be the set of all augmented data, P be the distribution of augmented data $x \sim \mathcal{A}(\cdot|\bar{x})$ where $\bar{x} \sim \bar{P}_{\bar{X}}$. Let $p(x, x^+)$ be the probability density of positive pair (x, x^+). We define a graph $G(V, w)$ where vertex set $V = X$ and edge weights $w(x, z) = p(x, z)$ for any $(x, z) \in X \times X$. In general, this graph may be infinitely large. To simplify math and avoid integrals, we assume $|X| = N$ where N is the number of all possible augmented images (which can be infinite or exponential in dimension).

The degree of node x is $p(x) = \sum_{z \in X} p(x, z)$. Let $A \in \mathbb{R}^{N \times N}$ be the adjacency matrix of this graph defined as $A_{x,z} = p(x, z)$, and let $\bar{A}$ ber the normalized adjacency matrix such that $\bar{A}_{x,z} = \frac{p(x,z)}{\sqrt{p(x)p(z)}}$.

The following lemma shows that contrastive learning is closely related to the eigendecomposition of $\bar{A}$.

Lemma 10.9. *Let $L(f) = -2\,\mathbb{E}_{(x,x^+)} f(x)^\top f(x^+) + \mathbb{E}_{(x,x')} \left(f(x)^\top f(x')\right)^2$. Suppose $X = \{x_1, \cdots, x_N\}$, let matrix*

$$F = \begin{bmatrix} p(x_1)^{\frac{1}{2}} f(x_1)^\top \\ \vdots \\ p(x_N)^{\frac{1}{2}} f(x_N)^\top \end{bmatrix}. \tag{10.107}$$

Then,

$$L(f) = \|\bar{A} - FF^\top\|_F^2 + \textit{const.} \tag{10.108}$$

Hence, minimizing $L(f)$ w.r.t the variable f is equivalent to eigendecomposition of $\bar{A}$.

Proof. Directly expanding the Frobenius norm $\|\bar{A} - FF^\top\|_F^2$ as a sum over entries, we have

$$\|\bar{A} - FF^\top\|_F^2 = \sum_{x,z \in X} \left(\frac{p(x,z)}{\sqrt{p(x)}\sqrt{p(z)}} - f(x)^\top f(z) \sqrt{p(x)}\sqrt{p(z)} \right)^2 \tag{10.109}$$

$$= \text{const} - 2 \sum_{x,z \in X} p(x,z) f(x)^\top f(z) + \sum_{x,z \in X} p(x)p(z) \left(f(x)^\top f(z)\right)^2 \tag{10.110}$$

$$= \text{const} - 2 \mathop{\mathbb{E}}_{(x,x^+) \sim \text{positive}} f(x)^\top f(x^+) + \mathop{\mathbb{E}}_{(x,x') \sim \text{random}} \left(f(x)^\top f(x')\right)^2, \tag{10.111}$$

where the last equation uses the fact that $p(x,z)$ and $p(x)p(z)$ are the probability densities of (x,z) being a positive pair and a random pair, respectively. □

Standard matrix decomposition results tell us that the minimizer of $\|\bar{A}-FF^\top\|_F^2$ satisfies $F = U\cdot\mathrm{diag}(\gamma_i^{\frac{1}{2}})$, where γ_i's are the eigenvalues of $\bar{A}$ and $U \in \mathbb{R}^{N\times k}$ contains the top k eigenvectors of $\bar{A}$ as its columns. Suppose we use $v_1, \cdots, v_N$ to represent the rows of U, i.e.,

$$U = \begin{bmatrix} v_1^\top \\ \vdots \\ v_N^\top \end{bmatrix}. \tag{10.112}$$

Then we know $f(x_j) = p(x_j)^{-\frac{1}{2}} \cdot \mathrm{diag}(\gamma_i^{\frac{1}{2}}) \cdot v_j$ is the minimizer of the contrastive loss.

One interesting thing is that $f(x_i)$ has the same separability as v_i. This is because for any vector $b \in \mathbb{R}^k$, we have $\mathbb{1}\left[b^\top v_i > 0\right] = \mathbb{1}\left[b^\top \mathrm{diag}(\gamma_i^{-\frac{1}{2}}) f(x) > 0\right]$, suggesting linear head $\mathrm{diag}(\gamma_i^{-\frac{1}{2}})b$ applied on feature $f(x_i)$ would achieve the same classification accuracy as b applied on v_i. Thus, it suffices to analyze v_i's downstream accuracy under linear head.

Since v_i is exactly the feature used by the classic spectral clustering algorithm, we may ask when spectral clustering produces good features. As discussed in Section 10.2, spectral clustering is good at graph partitioning in stochastic block models. In this section, we aim to find more general settings where spectral clustering produces good features. For simplicity, let's consider a regular graph where $w(x) = \sum_{x'\in V} w(x,x') = \kappa$.[3]

The following lemma shows that suppose the graph roughly contains two clusters, then the spectral clustering features can be used to accurately predict which cluster a node belongs to.

Lemma 10.10. *Suppose the graph G can be partitioned into 2 clusters S_1, S_2 with size $|S_1| = |S_2| = \frac{N}{2}$, such that $E(S_1,S_2) = \sum_{x\in S_1, z\in S_2} w(x,z) \le \alpha\kappa N$. Furthermore, suppose G cannot be partitioned well into 3 clusters in the sense that for all partition T_1, T_2, T_3, we have $\max\{\phi(T_1), \phi(T_2), \phi(T_3)\} \ge \rho$. (Figure 10.5 gives a demonstration of these assumptions.) Then, let $g = \mathbb{1}(S_1) \in \mathbb{R}^N$ (i.e., $g_i = 1$ if $i \in S_1$), and $k \ge 6$,*

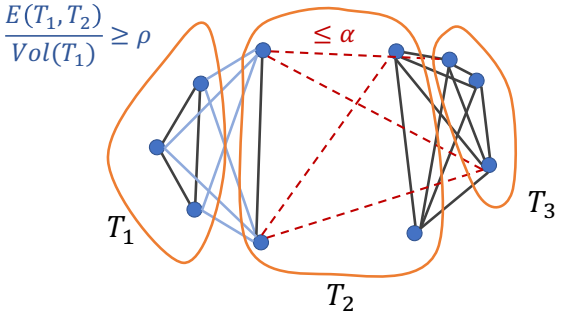

Figure 10.5: A demonstration of the assumptions in Lemma 10.10. The left half and right half of the graph can be chosen as S_1 and S_2, since there's at most α proportion of edges between them. Sets T_1, T_2, T_3 form a 3-way partition where $\phi(T_1) \ge \rho$.

there exists linear classifier b such that

$$\|Ub - g\|_2^2 \lesssim \frac{N\alpha}{\rho^2}, \tag{10.113}$$

where U contains the top k eigenvectors of $\bar{A}$ as its columns.

[3] It turns out that most, if not all, spectral graph theory tools on regular graph can extend to general graph settings. Therefore, it oftentimes suffices to consider a regular graph.

The above lemma essentially says that $\langle v_x, b\rangle \approx g_x$ for all data $x \in X$, where v_x is the x-th row of U.

Before proving the above lemma, we first introduce the following higher-order Cheeger inequality, which shows that when the graph cannot be partitioned well into 3 clusters, the 6-th smalled eigenvalue of the Laplacian cannot be too small.

Lemma 10.11 (Proposition 1.2 in [Louis and Makarychev, 2014]). *Let $G = (V, w)$ be a weight graph. Suppose the graph cannot be partitioned into 3 sets S_1, S_2, S_3 such that $\max\{\phi(S_1), \phi(S_2), \phi(S_3)\} \le \rho$. Then, we have*

$$\lambda_6 \gtrsim \rho^2.$$

Now we give a proof of Lemma 10.10.

Proof of Lemma 10.10. By Lemma 10.8 we know that

$$\frac{2}{N} g^\top L g = \frac{1}{N\kappa} \sum_{x,z} (g_x - g_z)^2 w(x,z) \tag{10.114}$$

↓ from the definition of g

$$= \frac{1}{N\kappa} \left(\sum_{x\in S_1, z\in S_2} w(x,z) + \sum_{x\in S_2, z\in S_1} w(x,z) \right) \tag{10.115}$$

$$= \frac{2}{N\kappa} E(S_1, S_2) \tag{10.116}$$

↓ from the assumption in theorem

$$\le \alpha. \tag{10.117}$$

Thus, g has to be mostly in the smaller eigenspace of L. Suppose L has eigenvalue $0 = \lambda_1 \le \lambda_2 \le \cdots \le \lambda_N$, with corresponding eigenvectors $u_1, u_2, \cdots u_N$. Define matrix $U = [u_1, \cdots, u_k] \in \mathbb{R}^{N\times k}$. Suppose $\sqrt{\frac{2}{N}} g = \sum_{i=1}^N \beta_i u_i$. Since $\|\sqrt{\frac{2}{N}} g\| = 1$, we know $\sum_{i=1}^N \beta_i^2 = 1$.

Since we know $g^\top L g = \sum_{i=1}^N \beta_i^2 \lambda_i \le \frac{N\alpha}{2}$, we can conclude $\sum_{i>k} \beta_i^2 \lambda_i \le \frac{N\alpha}{2}$, which implies that $\sum_{i>k} \beta_i^2 \le \frac{N\alpha}{2\lambda_{k+1}} \lesssim \frac{N\alpha}{\rho^2}$. Here we used the fact $\lambda_6 \gtrsim \rho^2$ by higher-order Cheeger inequality (Lemma 10.11). Thus, we have $\|g - \sum_{i=1}^k \beta_i u_i\|_2^2 = \|\sum_{i>k} \beta_i u_i\|_2^2 \lesssim \frac{N\alpha}{\rho^2}$ which finishes the proof. □

We can combine Lemma 10.9 and Lemma 10.10 to prove the following theorem, which shows that when the graph roughly contains 2 clusters, the feature learned from contrastive learning can be used to predict the cluster membership accurately.

Theorem 10.12. *Let $L(f) = -2\mathbb{E}_{(x,x^+)} f(x)^\top f(x^+) + \mathbb{E}_{(x,x')} \big(f(x)^\top f(x')\big)^2$, and $f^* : X \to \mathbb{R}^k$ is a minimizer of $L(f)$ for $k \ge 6$. Suppose the graph G can be partitioned into 2 clusters S_1, S_2 with size $|S_1| = |S_2| = \frac{N}{2}$, such that $E(S_1, S_2) = \sum_{x\in S_1, z\in S_2} w(x,z) \le \alpha\kappa N$. Furthermore, suppose G cannot be partitioned well into 3 clusters in the sense that for all partition T_1, T_2, T_3, we have $\max\{\phi(T_1), \phi(T_2), \phi(T_3)\} \ge \rho$. Let $y(x_i) = \mathbb{1}(x_i \in S_1)$ (i.e., $y(x_i) = 1$ if $x_i \in S_1$, otherwise $y(x_i) = 0$). Then, there exists linear classifier $b \in \mathbb{R}^k$ such that*

$$\frac{1}{N} \sum_{i\in[N]} \big(f(x_i)^\top b - y(x_i)\big)^2 \lesssim \frac{\alpha}{\rho^2}. \tag{10.118}$$

Proof. Let $U \in \mathbb{R}^{N\times k}$ contains the top k eigenvectors of $\bar{A}$ as its columns. By Lemma 10.10, we know there exists some $\hat{b} \in \mathbb{R}^k$ such that $\|U\hat{b} - g\|_2^2 \lesssim \frac{N\alpha}{\rho^2}$, where $g \in \mathbb{R}^N$ such that $g_i = y(x_i)$. Let $v_1, \cdots, v_N$ be the rows of U. According to Lemma 10.9, we know that $f(x_i) = p(x_i)^{-\frac{1}{2}} \cdot \mathrm{diag}(\gamma_j^{\frac{1}{2}}) \cdot v_i = \kappa^{-\frac{1}{2}} \cdot \mathrm{diag}(\gamma_j^{\frac{1}{2}}) \cdot v_i$, where γ_j is the j-th largest eigenvalue of $\bar{A}$, and $\mathrm{diag}(\gamma_j^{\frac{1}{2}})$ is a diagonal matrix containing $\gamma_1^{\frac{1}{2}}, \gamma_2^{\frac{1}{2}}, \cdots, \gamma_k^{\frac{1}{2}}$ as

its entries. Thus, if we let $b = \sqrt{\kappa} \cdot \mathrm{diag}(\gamma_j^{-\frac{1}{2}}) \cdot \hat{b}$, we would have

$$\sum_{i\in[N]} (f(x_i)^\top b - y(x_i))^2 = \sum_{i\in[N]} (v_i^\top \hat{b} - g_i)^2 = \|U\hat{b} - g\|_2^2 \lesssim \frac{N\alpha}{\rho^2}. \tag{10.119}$$

□