Deep Clustering via Probabilistic Ratio-Cut Optimization

Ayoub Ghriss[1]
ayoub.ghriss@colorado.edu

Claire Monteleoni[1,2]
cmontel@colorado.edu

[1]Department of Computer Science, University of Colorado Boulder
[2] INRIA Paris

Abstract

We propose a novel approach for optimizing the graph ratio-cut by modeling the binary assignments as random variables. We provide an upper bound on the expected ratio-cut, as well as an unbiased estimate of its gradient, to learn the parameters of the assignment variables in an online setting. The clustering resulting from our probabilistic approach (PRCut) outperforms the Rayleigh quotient relaxation of the combinatorial problem, its online learning extensions, and several widely used methods. We demonstrate that the PRCut clustering closely aligns with the similarity measure and can perform as well as a supervised classifier when label-based similarities are provided. This novel approach can leverage out-of-the-box self-supervised representations to achieve competitive performance and serve as an evaluation method for the quality of these representations.

1 INTRODUCTION

Unsupervised learning is based on the premise that labels are not necessary for the training process, particularly for clustering tasks where samples that are highly similar are grouped together. Various clustering algorithms (Ezugwu et al., 2022) have been proposed in the context of machine learning and data mining. The K-Means algorithm (Lloyd, 1957) and ratio-cut partitioning (Hagen & Kahng, 1991) were among the first approaches to address the clustering problem. These methods were further refined and extended beyond binary partitioning and Euclidean distances, and they were even shown to be fundamentally equivalent in specific settings (Dhillon et al., 2004).

The recent advances in generative models and self-supervised representation learning have produced powerful embeddings that capture the similarity between the original data samples. This makes leveraging these similarity measures to achieve effective clustering more relevant than ever, as it can serve as pseudo-labels for pre-training classifiers or eliminate the need for a decoder (Ji et al., 2021) in the training of these generative models. Therefore, it is highly advantageous to develop a method that can transform the similarity information into clustering of equal quality. An efficient extension of spectral clustering to stochastic gradient descent would facilitate the conversion of learned embeddings to cluster assignments without relying on the full dataset or large batches to accurately approximate the graph structure of the embedding space.

Several clustering methods have been applied to data streams to provide weak signals for the pre-training of neural networks. Meanwhile, other approaches have been proposed to utilize deep learning specifically for clustering, particularly with autoencoders and generative neural networks (Jiang et al., 2017). Generally, these methods can be categorized into contrastive learning (Li et al., 2020), where samples are grouped based on pairwise distances in a large batch, or generative models such as variational autoencoders that use a prior with an auxiliary variable as the cluster assignment. The former methods do not fully capture the global structure of the clusters, while the latter may suffer from overfitting the prior due to the complexity of the neural network, with limited options to prevent this without relying on labels. Spectral clustering avoids these issues by clustering the data based on global similarities between clusters rather than just pairs of samples.

Spectral clustering has long resisted attempts to extend it to parametric learning, primarily due to the challenge of handling the spectral decomposition of

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. Copyright 2025 by the author(s).

large matrices. Since it is based on the spectral decomposition of the relaxed ratio-cut rather than the combinatorial version, it requires the projection of the data into a principal eigenspace. A clustering algorithm is then applied to these projections, further making the final performance dependent on the chosen clustering algorithm.

In this paper, we develop a novel method to optimize the ratio-cut without relying on the spectral decomposition of the Laplacian matrix by employing a probabilistic approach that treats the cluster assignment as random variables. Furthermore, we utilize neural networks to parameterize the assignment probabilities and address the clustering problem in an online manner using stochastic gradient descent. The result is an online learning algorithm that achieves a better ratio-cut objective than the memory-intensive spectral method applied to the full graph Laplacian. We also demonstrate that our approach achieves comparable performance to a supervised classifier when utilizing supervised similarity (i.e., two samples are considered similar if they share the same label). This showcases a high fidelity between the resulting clustering and the given similarity measure. The proposed algorithm can also serve as a drop-in replacement for any other clustering method used in the pre-training of text or speech transformers, which opens up several opportunities for enhancing the effectiveness of pre-training for downstream tasks.

2 BACKGROUND

In this section, we succinctly present the relevant elements related to the notion of graph ratio-cut. The curious reader may refer to von Luxburg (2007) for a more detailed account of spectral clustering and other types of graph cuts.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{K})$ be an undirected weighted graph where $\mathcal{V} \overset{\text{def}}{=} \{v_i \mid 1 \leq i \leq n\} \subset \mathbb{R}^p$ is the set of vertices, $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the set of edges e_{ij} linking vertices v_i and v_j, and $\mathcal{K} : \mathcal{V} \times \mathcal{V} \to \mathbb{R}^+$ is a symmetric non-negative kernel. Let $\boldsymbol{W}$ be the symmetric $n \times n$ adjacency matrix where $W_{ij} = \mathcal{K}(v_i, v_j)$. The degree of the vertex v_i is $d_i = \sum_j W_{ij}$, and the degree matrix $\boldsymbol{D} \overset{\text{def}}{=} \operatorname{diag}(d_1, \ldots, d_n)$.

Let $k \geq 2$ and $\mathcal{C}_k = \{\mathbb{C}_\ell | 1 \leq \ell \leq k\}$ a partitioning of the graph $\mathcal{G}$ into k disjoint clusters. We shall represent the subset $\mathbb{C}_\ell \subset \mathcal{V}$ using the binary assignment vector $\mathbf{1}_{\mathbb{C}_\ell} \in \{0,1\}^n$ where $\mathbf{1}_{\mathbb{C}_\ell}(i) = 1$ if and only if $v_i \in \mathbb{C}_\ell$. The size of $\mathbb{C}_\ell$ is measured using its cardinality $|\mathbb{C}_\ell| = \sum_{i=1}^n \mathbf{1}_{\mathbb{C}_\ell}(i)$, and we denote by $\boldsymbol{f}^{(\ell)} \overset{\text{def}}{=} \frac{1}{\sqrt{|\mathbb{C}_\ell|}} \mathbf{1}_{\mathbb{C}_\ell}$ its *ratio assignment* when $|\mathbb{C}_\ell| > 0$.

The *ratio-cut* for $\mathcal{C}_k$ is defined as:

$$\operatorname{RatioCut}(\mathcal{C}_k) \overset{\text{def}}{=} \frac{1}{2} \sum_{\ell=1}^{k} \frac{1}{|\mathbb{C}_\ell|} \sum_{i,j \in \mathbb{C}_\ell \times \overline{\mathbb{C}_\ell}} W_{ij} \tag{1}$$

$$= \frac{1}{2} \sum_{\ell=1}^{k} \frac{1}{|\mathbb{C}_\ell|} \mathbf{1}_{\mathbb{C}_\ell}^\top \boldsymbol{W} \left(\mathbf{1}_n - \mathbf{1}_{\mathbb{C}_\ell}\right), \tag{2}$$

where $\overline{\mathbb{A}}$ denotes $\mathcal{V} \backslash \mathbb{A}$, the complement of $\mathbb{A}$ in $\mathcal{V}$.

We define the unnormalized Laplacian matrix as $\boldsymbol{L}_{un} \overset{\text{def}}{=} \boldsymbol{D} - \boldsymbol{W}$, which can be used to express the ratio-cut of $\mathcal{C}_k$ as:

$$\operatorname{RatioCut}(\mathcal{C}_k) = \frac{1}{2} \operatorname{Tr}\left[\boldsymbol{F}_{\mathcal{C}_k}^\top \boldsymbol{L}_{un} \boldsymbol{F}_{\mathcal{C}_k}\right], \tag{3}$$

where $\boldsymbol{F}_{\mathcal{C}_k} \overset{\text{def}}{=} \left[\boldsymbol{f}^{(1)}, \ldots \boldsymbol{f}^{(k)}\right] \in \mathbb{R}^{n \times k}$ is the ratio assignments matrix.

Since the clusters should be disjoint and different from the naive partitioning $\{\mathcal{V}, \emptyset, \ldots, \emptyset\}$, we can express these constraints as $\boldsymbol{F}_{:,\ell}^\top \mathbf{1}_V \neq 1$ for all $1 \leq \ell \leq k$.

The optimization of the ratio-cut is then equivalent to minimizing a Rayleigh quotient on $\{0,1\}^{n \times k}$. Solving eq. (4) on the set $\{0,1\}^n$ is generally an NP-hard problem. The optimization is hence relaxed so that the unknown vectors are in the unit sphere $S^{(n)} = \left\{x \in \mathbb{R}^n, \|x\|^2 = 1\right\}$. The objective is then:

$$\begin{aligned} &\underset{\boldsymbol{F}}{\text{minimize}} && \operatorname{Tr}(\boldsymbol{F}^\top \boldsymbol{L} \boldsymbol{F}) \\ &\text{subject to} && \boldsymbol{F}^\top \boldsymbol{F} = \boldsymbol{I}_k \text{ and } \boldsymbol{F}_{:,j}^\top \mathbf{1}_V \neq 1 \end{aligned} \tag{4}$$

The minimization of the Rayleigh quotient under the outlined constraints yields the k smoothest eigenvectors of the Laplacian matrix (excluding the trivial first eigenvector $\mathbf{1}_V$). Vertices within the same cluster are anticipated to have similar projections onto the solutions of the relaxed problem. As we ascend the Laplacian spectrum, the projections onto the eigenvectors encapsulate more specific (higher frequency) features. Subsequently, the binary assignments are determined through k-means clustering (Ng et al., 2001) of the relaxed problem's solution.

In this paper, we introduce a novel approach to optimizing eq. (4) that circumvents the necessity for spectral decomposition of extensive matrices or kernels, thereby sidestepping the relaxation of the problem into Euclidean space. Rather, we propose to relax the problem into an optimization over a simplex through the parameterization of the cluster assignment distribution.

3 RELATED WORK

There have been many attempts to extend Spectral Clustering to streaming settings (Guha et al., 2000), infinitely countable datasets, or continuous input spaces. For instance, Yoo et al. (2016) proposed an extension to non-parametric Spectral Clustering for data streams, particularly when the kernel similarity is bilinear or can be approximated linearly. With the arrival of each new stream batch, a batch-based Singular Value Decomposition (SVD) is used to embed the stream and realign a facility set consisting of anchor points. These anchor points are then utilized to run a batch K-means algorithm (Shindler & Wong, 2011) to cluster the new points.

However, research on extending Spectral Clustering to the domain of parametric learning, where the optimization variable is a mapping of the input space to the eigenspace, is limited. This is primarily due to the sensitivity of the eigendecomposition (Alam & Bora, 2005) and the proven difficulty of eigendecomposition in non-parametric settings when the similarity kernel takes specific forms (Mohan & Monteleoni, 2017).

One of the early attempts in this direction was made in the context of Deep Reinforcement Learning (Kulkarni et al., 2016), where the Markov Decision Process (MDP) is decomposed into sub-tasks that navigate the representation space (Machado et al., 2017a). These sub-tasks, also called *options* (Machado et al., 2017b), correspond to eigenvectors associated with the largest eigenvalues of the graph Laplacian, where the vertices represent elements of the state space and the similarity is the likelihood of the agent moving from one state to another. Extending such an approach to a clustering setting would require constructing an MDP for which the correspondence between eigenvectors and options holds.

SpectralNets (Shaham et al., 2018) were perhaps the first attempt at explicitly training a neural network to approximate eigenvectors of the Laplacian for large-scale clustering. The learned mapping is a neural network $N_\theta : \mathbb{R}^p \to \mathbb{R}^k$, parameterized by θ, that maps the input vertices to their projection on the subspace spanned by the k-smoothest eigenvectors of the Laplacian. The network is constructed to ensure that the output features are approximately orthonormal:

$$\frac{1}{n}\sum_{i=1}^{n} N_\theta(v_i)^\top N_\theta(v_i) \approx I_k$$

We denote by $\hat{N}_\theta$ the neural network without the last orthogonalization layer, which is based on the Cholesky decomposition of the estimated covariance matrix $\mathbf{\Sigma}_\theta \in \mathbb{R}^{k\times k}$:

$$\mathbf{\Sigma}_\theta = \frac{1}{n}\sum_{i=1}^{n} \hat{N}_\theta(v_i)^\top \hat{N}_\theta(v_i).$$

SpectralNets approach is then based on the minimization of:

$$\mathcal{L}_{spectral}(\theta) \overset{\text{def}}{=} \sum_{i,j=1}^{n} W_{ij} \left\| N_\theta(v_i) - N_\theta(v_j) \right\|^2 \qquad (5)$$

The matrix $\mathbf{\Pi}_\theta \in \mathbb{R}^{k\times k}$ serves as the parametric counterpart of $\boldsymbol{F}^\top \boldsymbol{L} \boldsymbol{F}$ derived from eq. (4). Training the neural network equates to an alternating optimization scheme, where one batch is utilized to estimate $\mathbf{\Sigma}_\theta$, while the other is employed to minimize the loss $\mathcal{L}_{spectral}(\theta)$.

During training, should the input exhibit noise, the SpectralNet's output may converge to a constant mapping, resulting in the spectral loss approaching zero. Consequently, this approach encounters numerical instability in high-dimensional data scenarios. One common strategy to circumvent this involves offsetting the matrix $\mathbf{\Sigma}_\theta$ by a scaled identity and establishing an appropriate stopping criterion. However, the resulting algorithm remains highly susceptible to input and similarity noise. This sensitivity becomes particularly evident when benchmarked against non-parametric Spectral Clustering. SpectralNets achieve competitive performance only when the input consists of the code space from another well-performing pre-trained variational encoder (Jiang et al., 2017).

Spectral Inference Networks (Spin) (Pfau et al., 2020) adopt a distinct strategy for parametric optimization of the Rayleigh quotient. As defined previously, the loss function takes the form:

$$\mathcal{L}_{spin}(\theta) = \operatorname{Tr}\left(\mathbf{\Sigma}_\theta{}^{-1} \mathbf{\Pi}_\theta\right)$$

While primarily utilized within the domain of Deep Reinforcement Learning, targeting the eigenfunctions associated with the largest eigenvalues, this approach can be adapted for loss minimization rather than maximization.

Upon differentiation (Petersen & Pedersen, 2012), the full gradient can be expressed as:

$$\operatorname{Tr}\left(\mathbf{\Sigma}_\theta{}^{-1} \nabla_\theta \mathbf{\Pi}_\theta\right) - \operatorname{Tr}\left(\mathbf{\Sigma}_\theta{}^{-1} (\nabla_\theta \mathbf{\Sigma}_\theta) \mathbf{\Sigma}_\theta{}^{-1} \mathbf{\Pi}_\theta\right)$$

The Spin algorithm estimates $\mathbf{\Sigma}_\theta$ and $\nabla_\theta \mathbf{\Sigma}_\theta$ utilizing moving averages. It addresses the issue of overlapping eigenvectors by modifying the gradient to ensure sequential independence in updates: the gradients of the first l eigenfunctions are made independent of gradients from other components $l+1,\dots,k$. This alteration significantly slows down training, in addition to the computational burden of computing $\nabla_\theta \boldsymbol{\Sigma}_\theta$, which necessitates calculating k^2 backward passes and storing $k^2 \times \text{size}(\theta)$, thereby increasing memory and computational costs.

We assert that the performance of these attempts is ultimately hindered by the difficulty of the eigendecomposition of large kernels. Therefore, we take a different approach to optimizing the graph ratio-cut that circumvents the spectral decomposition.

4 PROPOSED METHOD

In our probabilistic approach, the minimization of the ratio-cut is addressed differently. Instead of the deterministic assignments $\mathbf{1}_{\mathbb{C}_\ell}$, we use the random assignment vector $\mathbf{a}^{(\ell)} \in \{0,1\}^n$ under the assumption that $(\mathbf{a}_i^{(\ell)})_i$ are independent random variables such that:

$$\Pr(v_i \in \mathbb{C}_\ell) = \Pr\left(\mathbf{a}_i^{(\ell)} = 1\right) \overset{\text{def}}{=} P_{i,\ell},$$

where the rows of the matrix $\boldsymbol{P} \in [0,1]^{n\times k}$ sum to 1: $\sum_{\ell=1}^k P_{i,\ell} = 1$ for all $i \in \{1,\dots,n\}$.

The random assignment for each vertex i follows a categorical distribution of parameter $\boldsymbol{P}_{i,:}$ and the random clustering $\mathcal{C}_k$ is thus parameterized by $\boldsymbol{P} \in [0,1]^{n\times k}$. We define the random ratio-assignment vector $\mathbf{f}^{(\ell)}$ for cluster $\mathbb{C}_\ell$ as:

$$\mathbf{f}^{(\ell)} = \frac{1}{\sqrt{|\widehat{\mathbb{C}_\ell}|}}\mathbf{a}^{(\ell)},$$

where $|\widehat{\mathbb{C}_\ell}| = \sum_{i=1}^n \mathbf{a}_i^{(\ell)}$, and all the elements of $\mathbf{f}^{(\ell)}$ are in the interval $[0,1]$ with the convention $\frac{0}{0} = 1$.

The stochastic counterpart of the quantity introduced earlier in Equation (1) is:

$$\widehat{\text{RatioCut}}(\mathcal{C}_k) = \frac{1}{2}\sum_{\ell=1}^{k}\sum_{1\le i,j\le n} W_{ij}(\mathbf{f}_i^{(\ell)} - \mathbf{f}_j^{(\ell)})^2. \quad (6)$$

Equation (3) can also be extended to the stochastic setting similarly.

The next step is to compute the expected ratio-cut for a clustering $\mathcal{C}_k$ parameterized by $\boldsymbol{P}$. Without loss of generality, we only have to compute $\mathbb{E}\left[(\mathbf{f}_1^{(\ell)} - \mathbf{f}_2^{(\ell)})^2\right]$ as a function of $\boldsymbol{P}$ due to the linearity of the expectation. We prove the following result in Appendix A.1:

Lemma 4.1 (Difference expectation). *Let $\mathbb{C}$ be a subset of $\mathcal{V}$ and $\mathbf{a}$ its random assignment vector parameterized by $\boldsymbol{p} \in [0,1]^n$. Let $\mathbf{f}$ be its random ratio-assignment vector. Then we have the following:*

$$\mathbb{E}\left[(\mathbf{f}_1 - \mathbf{f}_2)^2\right] = (p_1 + p_2 - 2p_1p_2)\mathbb{E}\left[\frac{1}{1 + |\widehat{\mathbb{C}^{\perp(1,2)}}|}\right],$$

where $\mathbb{C}^{\perp(i,j)} = \mathbb{C}\setminus\{v_i, v_j\}$

To avoid excluding pairs $(i,i)_{1\le i\le n}$ from the summation in Equation (6) each time, we will henceforth assume that $W_{ii} = 0$ for all $i \in \{1,\dots,n\}$.

The random variable $|\widehat{\mathbb{C}^{\perp(1,2)}}| = \sum_{i=3}^n \mathbf{a}_i$ is the sum of (n-2) independent (but not identical) Bernoulli random variables, also known as a Poisson binomial. It coincides with the binomial distribution when the Bernoulli random variables share the same parameter. See Appendix A.3 for a compilation of properties of this distribution including the proof of Lemma 4.2.

Lemma 4.2 (Poisson binomial expectation). *Let Z be a Poisson binomial random variable of parameter $\boldsymbol{\alpha} = (\alpha_1,\dots,\alpha_m) \in [0,1]^m$. Then we have:*

$$\mathbb{E}\left[\frac{1}{1+\text{Z}}\right] = \int_0^1 \prod_{i=1}^m (1 - \alpha_i t)dt$$

We can now express the expected ratio-cut by combining the results from Lemma 4.1 and Lemma 4.2.

Theorem 4.1 (Ratio-cut expectation). *The expected ratio-cut of the random clustering $\mathcal{C}_k$ parameterized by $\boldsymbol{P} \in [0,1]^{n\times k}$ can be computed as $\mathbb{E}\left[\widehat{\text{RatioCut}}(\mathcal{C}_k)\right] = \sum_{\ell=1}^k \text{RC}(\boldsymbol{P}_{:,\ell})$ where:*

$$\text{RC}(\boldsymbol{p}) = \frac{1}{2}\sum_{i,j=1}^n W_{ij}\,(p_i + p_j - 2p_ip_j)\,\mathbb{I}(p,i,j)$$

$$\mathbb{I}(\boldsymbol{p},i,j) \overset{def}{=} \int_0^1 \prod_{m\ne i,j}^{n} (1 - p_m t)dt$$

The expression for $\text{RC}(\boldsymbol{p})$ can be further simplified by consolidating the terms linear in $\boldsymbol{p}$. However, we retain the current formulation as it will prove useful when sampling random pairs (i,j) in the implementation.

We will drop the cluster index ℓ in the next sections and use $\boldsymbol{p} \in [0,1]^n$ as the parameter of the random assignment. The goal now is to optimize $\text{RC}(\boldsymbol{p})$.

4.1 Computation of the expected ratio-cut

A straightforward method to estimate the expected ratio-cut from Theorem 4.1 involves discretizing the interval $[0, 1]$ to approximate the integral. By using a uniform discretization with a step size of $\frac{1}{T}$ for $T > 0$, we can apply the formula:

$$\mathbb{I}(\boldsymbol{p}, i, j) = \frac{1}{T} \sum_{t=1}^{T} \prod_{m \neq i,j} (1 - p_i \frac{t}{T})$$

The quality of the approximation depends on T, but since the integrated function is polynomial in t, we have the luxury of using a weighted discretization scheme that ensures the exact computation of the integral using the following quadrature method proven in Appendix A.4.

Lemma 4.3 (Integral computation via quadrature)**.** *Let $m > 0$ be an integer and $c_m \overset{def}{=} \lfloor \frac{m}{2} \rfloor + 1$ (the smallest integer such that $2c_m \geq m + 1$), then there exist c_m tuples $(s_q, t_q)_{1 \leq q \leq c_m}$ such that:*

$$\int_0^1 \prod_{i=1}^{m} (1 - p_i t) dt = \sum_{q=1}^{c_m} s_q \prod_{i=1}^{m} (1 - p_i t_q),$$

for all $\boldsymbol{p} \in [0, 1]^m$.

The computation of $\mathbb{I}(\boldsymbol{p}, i, j)$ for a given pair (i, j) costs $O(n^2)$. We can still obtain the total $\mathrm{RC}(\boldsymbol{p})$ in $O(c_n n^2)$ instead of $O(n^4)$ by computing the c_n quantities $\left(\sum_{m=1}^{n} \log(1 - p_m t_q)\right)_{1 \leq q \leq c_n}$ in advance. Another challenge encountered when using the quadrature is the instability of the computations due to the potentially large number of elements in the product. Even with a tight approximation using batch-based estimates, it still costs $O(b^3)$, where b is the batch size. Furthermore, the higher the number of clusters, the larger the batch size should be to obtain a good estimate of the integral for different clusters. These challenges are to be expected since the original combinatorial problem is generally NP-hard. See Appendix A.5 for a detailed analysis of the batch-based estimate. Instead, we prove in Appendix A.6 that the expected ratio cut can be upper-bounded by a much more accessible quantity, as shown in Lemma 4.4.

Lemma 4.4 (Integral upper-bound)**.** *We adopt the same notation of Lemma 4.2 where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m) \in [0, 1]^m$, and we assume that $\overline{\boldsymbol{\alpha}} \overset{def}{=} \frac{1}{m} \sum_{i=1}^{m} \alpha_i > 0$. Then we have:*

$$\int_0^1 \prod_{i=1}^{m} (1 - \alpha_i t) dt \leq \frac{1}{(m + 1)\overline{\boldsymbol{\alpha}}}.$$

The assumption $\overline{\boldsymbol{\alpha}} > 0$ is not necessary if we extend the property to $\mathbb{R} \cup \{+\infty\}$. We can now use Lemma 4.4 to upper-bound $\mathrm{RC}(\boldsymbol{p})$.

Lemma 4.5 (RC upper-bound)**.** *Using the notations of Theorem 4.1 and Lemma 4.4, we have:*

$$\mathrm{RC}(\boldsymbol{p}) \leq \frac{e^2}{2n} \frac{1}{\overline{\boldsymbol{p}}} \sum_{1 \leq i,j \leq n} w_{ij}(p_i + p_j - 2p_i p_j)$$

Interpretation Before we proceed with determining an unbiased gradient estimator for the bound in Lemma 4.5, we seek to understand the significance of the various quantities introduced previously. In contrastive learning, the objective can be written as $-2 \sum_{ij} W_{ij} p_i p_j$, whose minimization aims to assign highly similar samples to the same cluster. The objective in ratio-cut can be expressed as $\sum_{ij} W_{ij} \left[p_i(1 - p_j) + p_j(1 - p_i)\right]$, which is minimized when dissimilar samples are separated into different clusters, in alignment with the initial goal of minimizing the cut. It is worth noting that the minimum of the term $W_{ij} \left[p_i(1 - p_j) + p_j(1 - p_i)\right]$ is W_{ij}, and it is achieved when either v_i and v_j are in different clusters.

Lemma 4.2 generalizes the scaling by the inverse of the size of a cluster $\mathbb{C}$ to the probabilistic setting. Let's assume that $\boldsymbol{p} \in \{0, 1\}^n$ (deterministic assignment), then $(p_i + p_j - 2p_i p_j) = 1$ if and only if v_i and v_j are in different clusters. In such case, we can easily prove the following:

$$\mathbb{E}\left[\frac{1}{1 + |\widehat{\mathbb{C}}|}\right] = \int_0^1 \prod_{i=1, p_i=1}^{n} (1 - t) dt = \frac{1}{1 + n\overline{\boldsymbol{p}}}.$$

As n becomes sufficiently large, the quantity $\frac{1}{n} \frac{1}{\overline{\boldsymbol{p}}}$ from Lemma 4.5 approximates $\frac{1}{1 + n\overline{\boldsymbol{p}}}$ and, therefore, also approximates $\mathbb{E}\left[\frac{1}{1 + |\widehat{\mathbb{C}}|}\right]$. Consequently, the bounds we've seen so far are tight in the deterministic case.

4.2 Stochastic gradient of the objective

We will drop the scaling factor $\frac{1}{2n}$ when upper bounding $\mathrm{RC}(\mathcal{C}_k)$, we can thus succinctly write the following upper bound for the expected ratio-cut with $\overline{\boldsymbol{P}} = \mathrm{diag}\left(\overline{\boldsymbol{P}_{:,1}}, \ldots, \overline{\boldsymbol{P}_{:,k}}\right)$ as $\mathrm{RC}(\mathcal{C}_k) \leq \mathcal{L}_{rc}(\boldsymbol{W}, \boldsymbol{P})$ such that:

$$\begin{aligned} \mathcal{L}_{rc}(\boldsymbol{W}, \boldsymbol{P}) &= \sum_{\ell=1}^{k} \sum_{i,j=1}^{n} \frac{1}{\overline{\boldsymbol{P}_{:,\ell}}} W_{ij} \left(P_{i,\ell} + P_{j,\ell} - 2P_{i,\ell} P_{j,\ell}\right) \\ &= \mathrm{Tr}(\overline{\boldsymbol{P}}^{-1} (\mathbf{1}_{n,k} - \boldsymbol{P})^\top \boldsymbol{W} \boldsymbol{P}), \end{aligned} \tag{7}$$

Let us examine the derivative of $\mathcal{L}_{rc}$ of a single cluster with respect to p_i,$\dot{p}_i = \frac{d\mathcal{L}_{prcut}}{dp_i}$:

$$\dot{p}_i \propto \frac{1}{\overline{p}^2} \sum_{i,j=1}^{n} W_{ij} \left[(1 - 2p_j)\overline{p} - \frac{1}{n}(p_i + p_j - 2p_i p_j)\right], \tag{8}$$

Algorithm 1 Probabilistic Ratio-Cut (PRCut) Algorithm

```
Require: Dataset $(v_i)$, Similarity kernel $\mathcal{K}$, batch size $b$, encoder $N_\theta$ parameterized by $\theta$, number of clusters $k$,$\overline{\boldsymbol{P}}_0 = \frac{1}{k}\mathbf{1}_k$,$\beta > 0$, polytope regularization weight $\gamma$, $t = 0$.
1: while not terminated do
2:     $t \leftarrow t + 1$
3:     Sample left batch $S_l$ of size $b$
4:     Sample right batch $S_r$ of size $b$
5:     Compute $\boldsymbol{W} = \mathcal{K}(S_l, S_r)$
6:     compute $\boldsymbol{P}^l_\theta, \boldsymbol{P}^r_\theta \leftarrow N_\theta(S_l), N_\theta(S_r) \in \mathbb{R}^{b\times k}$
7:     Update $\overline{\boldsymbol{P}}_t \leftarrow (1-\beta_t)\overline{\boldsymbol{P}}_{t-1} + \frac{\beta_t}{2}\left(\overline{\boldsymbol{P}^l_\theta} + \overline{\boldsymbol{P}^r_\theta}\right)$
8:     Compute $\dot{\boldsymbol{P}}^l_\theta$ and $\dot{\boldsymbol{P}}^r_\theta$ using Equation (8)
9:     Back-propagate Tr $\left[\boldsymbol{P}^l_\theta \,\mathrm{sg}(\dot{\boldsymbol{P}}^l_\theta)^\top + \boldsymbol{P}^r_\theta \,\mathrm{sg}(\dot{\boldsymbol{P}}^r_\theta)^\top\right]$
10:    Back-propagate $\gamma D_{\mathrm{KL}}(\overline{[\boldsymbol{P}^l_\theta, \boldsymbol{P}^r_\theta]} \parallel \frac{1}{k}\boldsymbol{I}_k)$
11:    Use the accumulated gradients $g_t$ to update $\theta$
12: end while
```

In order to obtain an unbiased estimate for the gradient, we need to acquire an accurate estimate of $\overline{\boldsymbol{p}}$. We do this in the online setting by computing the moving average:

$$\overline{\boldsymbol{p}}_t^{\,ma} = (1-\beta_t)\overline{\boldsymbol{p}}_{t-1}^{\,ma} + \beta_t \overline{\boldsymbol{p}}_t,$$

such that $\beta_t = \frac{\beta}{t}$ and $\overline{\boldsymbol{p}}_t$ is a batch-based estimate of $\overline{\boldsymbol{p}}$ at time t. Then, the unbiased gradient can be obtained by back-propagating $[\mathrm{sg}(\dot{p}_i)p_i]$, with sg being the gradient-stopping operator.

4.3 Regularization

As we aim to optimize $\mathcal{L}_{rc}(\boldsymbol{W}, \boldsymbol{P})$ using gradient descent, the main challenge our method faces is that the assignments may collapse to a single cluster $\mathbb{C}_m$ with a probability close to 1. This translates to $\left(\overline{\boldsymbol{P}_{:,\ell}}\right)_{\ell \neq m} \approx 0$, which renders the gradient updates highly unstable. Benamou et al. (2014) have adopted the constraint that $\boldsymbol{P}$ belongs to the polytope of distributions $\mathcal{U}_k$, defined as:

$$\mathcal{U}_k = \left\{ P \in \mathbb{R}_+^{n\times k} | P^\top \mathbf{1}_n = \frac{1}{k}\mathbf{1}_k \right\},$$

Instead of restricting the clusters to be equally likely through the Bregman projection into $\mathcal{U}_k$, we only encourage such behavior using Kullback-Leibler divergence regularization. By selecting the appropriate regularization weight γ, the additional term will ensure that the likelihood of each of the k clusters, $\overline{\boldsymbol{P}}$, exceeds a certain threshold $\delta(\gamma) > 0$ for the optimal $\boldsymbol{P}$. This approach does not necessarily imply that all the clusters will be utilized, as the assignment probability for cluster $\mathbb{C}_\ell$ could be significantly above δ without any sample being more likely assigned to cluster $\mathbb{C}_\ell$. The final objective that we aim to optimize is:

$$\mathcal{L}_{prcut}(\boldsymbol{W}, \boldsymbol{P}) = \mathcal{L}_{rc}(\boldsymbol{W}, \boldsymbol{P}) + \gamma D_{\mathrm{KL}}(\overline{P} \parallel \frac{1}{k}\mathbf{1}_k)$$

4.4 Similarity measure

We have assumed thus far that the kernel $\mathcal{K}$ is provided as input. We also assert that the performance of any similarity-based clustering ultimately depends on the quality of the similarity function used. The simplest kernel we can use is the adjacency matrix for a k-nearest neighbors graph, where $\mathcal{K}(v_i, v_j) = 1$ when either v_i or v_j is among the k nearest neighbors of each other (to ensure that the kernel is symmetric). We can also use the *Simple* (SimCLR) or the *Symbiotic* (All4One) contrastive learning methods (Chen et al., 2020; Estepa et al., 2023) to train a neural network to learn the pairwise similarities. Once we train our similarity network, we use the cosine of the representations of the samples as our similarity function to either compute $W_{ij} = \exp\left(\frac{cosine(z_i, z_j)}{\tau}\right)$ for some temperature $\tau > 0$ or to build the k-nearest neighbors graph based on these representations.

4.5 Computational and Memory Footprint

If the similarity matrix $\boldsymbol{W}$ is dense, the time complexity of computing the PRCut objective is $O(n^3)$, identical to a vanilla spectral clustering approach. In contrast, for a sparse graph where similarity is determined based on the m-nearest neighbors, the time complexity of conventional spectral clustering is $O(nmk)$, not accounting for the additional cost of $O(nk^2)$ incurred by running k-means on the computed eigenvectors.

In the case of batch PRCut, the expected computational cost for evaluating the batch loss with a batch size of b is $O\left(\frac{m}{n}kb^2\right)$, while the memory requirement remains constant at $O(b^2)$. Consequently, when executed for T steps, the overall time complexity of stochastic PRCut becomes $O\left(T\frac{m}{n}kb^2\right)$.

5 EXPERIMENTS

We train a neural network $N_\theta : \mathbb{R}^p \mapsto \Delta^{k-1}$ that maps the vertex v_i to its cluster assignment probabilities $\boldsymbol{P}_i^\theta$, where $\theta \in \mathbb{R}^q$ is the parameter of the network. The last layer of the neural network is a Softmax layer, ensuring that the constraint $\sum_{\ell=1}^k P_{i\ell} = 1$ is always satisfied. We assume that the number of class labels k is provided. When computing the batch-based gradient using Equation (8), we use the factor $\frac{1}{b}$ instead of

$\frac{1}{n}$, where b is the batch size[1].

Since $\mathcal{L}_{rc}(\boldsymbol{W}, \boldsymbol{P}^{\theta})$ is linear in $\boldsymbol{W}$, we scale it by $\frac{1}{\|\boldsymbol{W}\|_1}$ to ensure consistency in the gradient descent method across different datasets and similarity measures.

Metrics To benchmark the performance of our algorithm, we assume that we have access to the true labeling $\boldsymbol{y} = (y_i)_i$ and the algorithm's clustering $\boldsymbol{c}$. We use three different metrics defined as follows:

- **Unsupervised Accuracy (ACC)**: We use the *Kuhn-Munkres* algorithm (Munkres, 1957) to find the optimal permutationσ_k of $\{1, \ldots, k\}$ such that $\frac{1}{n} \max_{\sigma \in \sigma_k} \sum_{i=1}^{n} 1_{y_i = \sigma(c_i)}$ is maximized between the true labeling $(y_i)_i$ and the clustering $(c_i)_i$.
- **Normalized Mutual Information (NMI)** is defined as: $\text{NMI}(\boldsymbol{y}, \boldsymbol{c}) = \frac{\mathcal{I}(\boldsymbol{y}, \boldsymbol{c})}{\max\{\mathcal{H}(\boldsymbol{l}), \mathcal{H}(\boldsymbol{c})\}}$ Where $\mathcal{I}(\boldsymbol{y}, \boldsymbol{c})$ is the mutual information between $\boldsymbol{y}$ and $\boldsymbol{c}$ and $\mathcal{H}$ is the entropy measure.
- **Adjusted Rand Index (ARI)** to evaluates the agreement between the true class labels and the learned clustering.
- **Ratio Cut (RC)** is computed using the formula in Equation (3) using the raw Laplacian matrix.

Before we compare our method to spectral clustering and other clustering approaches, we first assess the quality of the clustering when using the perfect similarity measure, where $\mathcal{K}(v_i, v_j) = 1$ if v_i and v_j share the same label, and $\mathcal{K}(v_i, v_j) = 0$ otherwise. For this experiment, we employ a simple Multi-Layer Perceptron (MLP) consisting of 3 layers with 512 hidden units, followed by Gaussian Error Linear Units (GeLU) activations. In the PRCut network, the last layer utilizes the Softmax activation. We compare the two methods across three datasets: MNIST (Lecun et al., 1998), Fashion-MNIST (F-MNIST) (Xiao et al., 2017), and CIFAR10 (Krizhevsky, 2009). The classifier is trained by optimizing the cross-entropy loss (CE).

The MLP classifier trained directly on the labeled data and the PRCut clustering using the labeled similarity demonstrate similar performance. This shows that our approach is more versatile and can be used in various learning methods while perfectly reflecting the quality of the similarity in the resulting clustering.

In Table 2 we compare our approach to vanilla Spectral Clustering (SC) using k-neighbor graph adjacency as a similarity measure for $k = 150$ using the entire training set. Since the final performance depends on the k-means initialization, we report the best run for SC.

[1]The code to reproduce PRCut is available at `https://github.com/ayghri/prcut`

Table 1: Benchmarking PRCut when using label-based similarity

Dataset	Method	ACC	NMI
MNIST	CE	0.980	**0.943**
	PRCut	**0.987**	0.938
F-MNIST	CE	0.885	**0.803**
	PRCut	**0.887**	0.789
CIFAR10	CE	**0.582**	**0.369**
	PRCut	0.571	0.359

Table 2: Comparison between PRCut and Spectral Clustering (SC)

Dataset	Method	ACC	NMI	RC
MNIST	SC (Best)	0.70	0.744	170.1
	PRCut	**0.821**	**0.778**	**150.2**
F-MNIST	SC (Best)	0.596	0.593	110.2
	PRCut	**0.658**	**0.620**	**101.5**
CIFAR10	SC (Best)	0.217	0.086	479.3
	PRCut	**0.243**	**0.121**	**440.3**

Since the final objective is to minimize the ratio cut, our approach achieves a better ratio cut value compared to vanilla spectral clustering. Whether such improvement translates to an enhancement in the clustering metrics depends on the similarity function. In all the 3 datasets, our approach outperforms the spectral relaxation of the ratio-cut.

For the third set of experiments reported in Table 3, we compare our method to VaDE (Jiang et al., 2017), VMM (Stirn & Knowles, 2024), and Turtle (Gadetsky et al., 2024). We have observed that VaDE does not generalize well across datasets, as its performance drastically degrades when applied to the Fashion-MNIST dataset. The VMM approach is a more consistent variational autoencoder (VAE) with a mixture model prior, achieving the best reported performance on the Fashion-MNIST dataset in the literature. We also report the results for methods with the suffix "-D," which use the pre-trained representation model DINOv2 (Oquab et al., 2024), while the suffix "-V" denotes methods that utilize the representation from the CLIP (Radford et al., 2021) vision transformer. In both of these approaches, the k-neighbors graph and the trained neural networks take the samples in the representation spaces as input.

The neural network consists of a single linear layer followed by the softmax operator. The learning rate was set to a constant value of 10^{-4} and using a 10^{-7} weight decay, with a bach size of $b = 2048$. We set the entropy regularization to $\gamma = 100.0$ and the moving average parameter to $\beta = 0.8$.

Table 3: Comparison of PRCut to the best-performing clustering methods.

Dataset	Method	ACC	NMI
MNIST	SC	0.701	0.744
	VaDE	0.857	0.838
	VMM	**0.960**	**0.907**
	Turtle-D	0.573	0.544
	PRCut-V	0.771	0.734
F-MNIST	SC	0.596	0.593
	VaDE	0.352	0.496
	VMM	0.712	0.688
	Turtle-D	0.764	0.723
	PRCut-D	**0.791**	**0.758**
CIFAR10	SC	0.217	0.121
	Turtle-V	0.972	0.929
	PRCut-V	**0.975**	**0.934**
CIFAR100	Turtle-D	**0.806**	**0.870**
	PRCut-D	0.789	0.856

Our method remains competitive compared to Turtle and achieves the best reported performance on Fashion-MNIST, which was designed to be a more challenging dataset than MNIST. Furthermore, our approach does not rely on any assumptions about the structure of the representation spaces, in contrast to Turtle, which is based on the premise that the best clusters are linearly separable—a property that is inherently valid for DINOv2 and CLIP. This may explain why it does not perform as well on grayscale images such as MNIST and Fashion-MNIST, where linear separability is not as prominent. We note that the Turtle results in our benchmark are based on a single representation space for a fair comparison, as the original paper performs best when it combines two representation spaces.

Table 4: Comparison representations using PRCut

Dataset	Rep	ACC	NMI
CIFAR10	Raw	0.243	0.121
	SimCLR	0.721	0.652
	All4One	0.710	0.635
	VitL-14	**0.975**	**0.934**
	DinoV2	0.774	0.797
CIFAR100	Raw	0.054	0.022
	SimCLR	0.362	0.483
	All4One	0.382	0.511
	VitL-14	0.720	0.755
	DinoV2	**0.789**	**0.856**

In Table 4, we compare the performance of our method using various pre-trained self-supervised representation learning models. In particular, we rely on the *solo-learn* library (da Costa et al., 2022) to retrieve or fine-tune the SimCLR and All4One models. While these two approaches perform well when evaluated using linear probes or a k-nearest neighbor classifier, the similarities in the embedding space for datasets with a higher number of class labels (CIFAR100) are too sparse to capture useful clusters. In that regard, DinoV2 embedding performs better than CLIP vision transformer.

6 Conclusion and future work

We have introduced a novel method that approaches the graph ratio-cut optimization from a probabilistic perspective. Compared to the classical Spectral Clustering based on the raw Laplacian, PRCut achieves better clustering performance and more optimal ratio-cut values. However, the performance strongly depends on the sparsity of the global similarity. With the recently developed self-supervised representation models that have proven to be powerful, we have demonstrated that PRCut translates the similarities between samples in the embedding space into high quality clustering and achieve new best clusterings for Fashion-MNIST while remaining competitive with state-of-the-art clustering methods.

While the current work serves as an introduction to this novel approach, the potential extensions of our method appear boundless. For example, PRCut could be leveraged as a dimensionality reduction technique by incorporating a bottleneck layer within the clustering neural network. Furthermore, the methodology could be adapted under the premise of linear separability of the true class labels, similar to the approach adopted by Gadetsky et al. (2024). The issue of dynamically determining the optimal number of clusters remains unresolved, given our current assumption that this information is supplied as an input to our algorithm.

The algorithm can also be extended to offline learning via Equation (7). Notably, when considering equally likely clusters, the problem simplifies to a quadratic form, with the Hessian being proportional to $-\boldsymbol{W}$. By leveraging the structure of the similarity matrix, we can derive additional guarantees about the optimal solution. This enables the definition of an iterative approach similar to the Sinkhorn-Knopp algorithm (Benamou et al., 2014), offering promising avenues for further exploration.

References

R. Alam and S. Bora. On sensitivity of eigenvalues and eigendecompositions of matrices. *Linear Algebra and its Applications*, 396:273–301, 2005. ISSN 0024-3795.

Jean-David Benamou, Guillaume Carlier, Marco Cuturi, Luca Nenna, and Gabriel Peyré. Iterative bregman projections for regularized transportation problems, 2014.

Sean X. Chen and Jun S. Liu. Statistical applications of the poisson-binomial and conditional bernoulli distributions. *Statistica Sinica*, 7(4):875–892, 1997. ISSN 10170405, 19968507.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.

Victor Guilherme Turrisi da Costa, Enrico Fini, Moin Nabi, Nicu Sebe, and Elisa Ricci. solo-learn: A library of self-supervised methods for visual representation learning. *Journal of Machine Learning Research*, 23(56):1–6, 2022. URL `http://jmlr.org/papers/v23/21-1155.html`.

Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: Spectral clustering and normalized cuts. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pp. 551–556, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138881.

Imanol G. Estepa, Ignacio Sarasua, Bhalaji Nagarajan, and Petia Radeva. All4one: Symbiotic neighbour contrastive learning via self-attention and redundancy reduction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16243–16253, October 2023.

Absalom E. Ezugwu, Abiodun M. Ikotun, Olaide O. Oyelade, Laith Abualigah, Jeffery O. Agushaka, Christopher I. Eke, and Andronicus A. Akinyelu. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence*, 110:104743, 2022. ISSN 0952-1976.

Artyom Gadetsky, Yulun Jiang, and Maria Brbic. Let go of your labels with unsupervised transfer, 2024.

S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pp. 359–366, 2000.

Lars W. Hagen and Andrew B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 11:1074–1085, 1991.

R. Bulirsch J. Stoer. *Introduction to Numerical Analysis*, volume 1. Springer New York, NY, 2013.

Qiang Ji, Yanfeng Sun, Junbin Gao, Yongli Hu, and Baocai Yin. A decoder-free variational deep embedding for unsupervised clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 33:5681–5693, 2021.

Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Yunfan Li, Peng Hu, Zitao Liu, Dezhong Peng, Joey Tianyi Zhou, and Xi Peng. Contrastive clustering. *CoRR*, abs/2009.09687, 2020. URL `https://arxiv.org/abs/2009.09687`.

S. P. Lloyd. Least square quantization in pcm. *Bell Telephone Laboratories Paper*, 28(2):129–137, March 1957. ISSN 0018-9448. doi: 10.1109/TIT.1982.1056489.

Marlos C. Machado, Marc G. Bellemare, and Michael Bowling. A Laplacian framework for option discovery in reinforcement learning. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2295–2304. PMLR, 06–11 Aug 2017a.

Marlos C. Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption discovery through the deep successor representation. *CoRR*, abs/1710.11089, 2017b.

Mahesh Mohan and Claire Monteleoni. Exploiting sparsity to improve the accuracy of nyström-based large-scale spectral clustering. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 9–16, 2017.

James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957. ISSN 03684245. URL `http://www.jstor.org/stable/2098689`.

Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In T. Dietterich, S. Becker, and Z. Ghahramani (eds.), *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. URL `https://arxiv.org/abs/2304.07193`.

K. B. Petersen and M. S. Pedersen. The matrix cookbook, nov 2012. URL `http://www2.compute.dtu.dk/pubdb/pubs/3274-full.html`. Version 20121115.

David Pfau, Stig Petersen, Ashish Agarwal, David G. T. Barrett, and Kimberly L. Stachenfeld. Spectral inference networks: Unifying deep and spectral learning, 2020.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL `https://arxiv.org/abs/2103.00020`.

Uri Shaham, Kelly Stanton, Henry Li, Ronen Basri, Boaz Nadler, and Yuval Kluger. Spectralnet: Spectral clustering using deep neural networks. In *International Conference on Learning Representations*, 2018.

Michael Shindler and Alex Wong. Fast and accurate k-means for large datasets. *NIPS*, 24, 01 2011.

Andrew Stirn and David A. Knowles. The vampprior mixture model, 2024. URL `https://arxiv.org/abs/2402.04412`.

Ulrike von Luxburg. A tutorial on spectral clustering. *CoRR*, abs/0711.0189, 2007.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017. URL `http://arxiv.org/abs/1708.07747`.

Shinjae Yoo, Hao Huang, and Shiva Prasad Kasiviswanathan. Streaming spectral clustering. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pp. 637–648, 2016.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [**Yes**]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [**Yes**]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [**Yes**]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results.[**Yes**]
   (b) Complete proofs of all theoretical results.[**Yes**]
   (c) Clear explanations of any assumptions.[**Yes**]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL).[**Yes**]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen).[**Yes**]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times).[**Yes**]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [**No**]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [**Yes**]
   (b) The license information of the assets, if applicable. [**Yes**]
   (c) New assets either in the supplemental material or as a URL, if applicable.[**Not Applicable**]
   (d) Information about consent from data providers/curators. [**Not Applicable**]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [**Not Applicable**]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [**Not Applicable**]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [**Not Applicable**]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [**Not Applicable**]

A Proofs for the upper bound

A.1 Ratio-cut expectation proof

Lemma 4.1 (Difference expectation)**.** *Let $\mathbb{C}$ be a subset of $\mathcal{V}$ and* $\mathbf{a}$ *its random assignment vector parameterized by $\boldsymbol{p} \in [0,1]^n$. Let* $\mathbf{f}$ *be its random ratio-assignment vector. Then we have the following:*

$$\mathbb{E}\left[(\mathbf{f}_1 - \mathbf{f}_2)^2\right] = (p_1 + p_2 - 2p_1p_2)\mathbb{E}\left[\frac{1}{1 + |\widehat{\mathbb{C}^{\perp(1,2)}}|}\right],$$

where $\mathbb{C}^{\perp(i,j)} = \mathbb{C}\backslash \{v_i, v_j\}$

Proof. Using the probabilistic formulations. We have:

$$|\widehat{C}| = \sum_{i=1}^{n} \mathbf{a}_i = \mathbf{a}_1 + \mathbf{a}_2 + |\widehat{C^{\perp(1,2)}}|$$

We can then expand the expectation:

$$\begin{aligned}
\mathbb{E}\left[(\widehat{\mathbf{f}}_1 - \widehat{\mathbf{f}}_2)^2\right] &= \mathbb{E}\left[\frac{1}{\mathbf{a}_1 + \mathbf{a}_2 + |\widehat{C^{\perp(i,j)}}|}\left(\mathbf{a}_1 - \mathbf{a}_2\right)^2\right] \\
&= \mathbb{E}\left[\frac{1}{1 + |\widehat{C^{\perp(1,2)}}|}\right]\Pr\left(\mathbf{a}_1, \mathbf{a}_2 = 1, 0\right) + \mathbb{E}\left[\frac{1}{1 + |\widehat{C^{\perp(1,2)}}|}\right]\Pr\left(\mathbf{a}_1, \mathbf{a}_2 = 0, 1\right) \\
&= \mathbb{E}\left[\frac{1}{1 + |\widehat{C^{\perp(1,2)}}|}\right]\left(p_1(1 - p_2) + (1 - p_1)p_2\right) \\
&= \mathbb{E}\left[\frac{1}{1 + |\widehat{C^{\perp(1,2)}}|}\right]\left(p_1 + p_2 - 2p_1p_2\right)
\end{aligned}$$

□

A.2 Binomial Poisson distribution properties

Let Z be a Poisson binomial random variable of parameter $\boldsymbol{p} = (\alpha_1, \ldots, \alpha_n)$. We can write Z as:

$$\mathrm{Z} = \sum_{i=1}^{m} \mathbf{r}_i,$$

where $(\mathbf{r}_i)_i$ are independent random variable such that:

$$\mathbf{r}_i \sim \text{Bernoulli}(\alpha_i).$$

Lemma A.1. *We denote by G_Z the Probability-Generating Function (PGF) for the Poisson binomial* Z *of parameter $\boldsymbol{\alpha} \in [0,1]^m$. We have the following:*

$$\mu \overset{def}{=} \mathbb{E}\left[\mathrm{Z}\right] = \sum_{i=1}^{n} \alpha_i$$

$$\sigma^2 \overset{def}{=} \mathbb{E}\left[(\mathrm{Z} - \mu)^2\right] = \mu - \sum_{i=1}^{m} \alpha_i^2$$

$$G_Z(t) \overset{def}{=} \mathbb{E}\left[t^{\mathrm{Z}}\right] = \prod_{i=1}^{m}(1 - \alpha_i + \alpha_i t)$$

Proof.

$$\mu \overset{\text{def}}{=} \mathbb{E}\left[\mathrm{Z}\right] = \sum_{i=1}^{m} \mathbb{E}\left[\mathbf{r}_i\right] = \sum_{i=1}^{n} \alpha_i$$

Since $(\mathbf{r}_i)$ are independent random variable, the variance of their sum is the sum of the variance:

$$\sigma^2 \overset{\text{def}}{=} \mathrm{Var}[Z] = \sum_{i=1}^{n} \mathrm{Var}[\mathbf{r}_i] = \sum_{i=1}^{n} \alpha_i(1-\alpha_i) = \mu - \sum_{i=1}^{n} \alpha_i^2$$

For the PGF:

$$\begin{aligned} G_Z(t) \overset{\text{def}}{=} \mathbb{E}\left[t^Z\right] &= \mathbb{E}\left[t^{\sum_{i=1}^{n} \mathbf{r}_i}\right] \\ &= \prod_{i=1}^{n} \mathbb{E}\left[t^{\mathbf{r}_i}\right] \prod_{i=1}^{n} (1-\alpha_i) * t^0 + \alpha_i t^1 \\ &= \prod_{i=1}^{n} (1-\alpha_i+\alpha_i t) \end{aligned}$$

□

The first approach for computing $\mathbb{E}\left[\frac{1}{1+Z}\right]$ is to use the definition of the expectation, which requires the knowledge of the probabilities $\left(\Pr\left(Z=i\right)\right)_i$:

$$\mathbb{E}\left[\frac{1}{1+Z}\right] = \sum_{i=0}^{n} \frac{1}{1+i} \Pr\left(Z=i\right).$$

Lemma A.2. *Let $\mathbb{P}\left[n\right]$ denote the power set of $\{1,\ldots,n\}$, and $\mathcal{I}_i$ be the set of elements of $\mathbb{P}\left[n\right]$ that contain exactly i unique integers.* $\Pr\left(Z=i\right)$ *can be computed as:*

$$\Pr\left(Z=i\right) = \sum_{I\in\mathcal{I}_i} \prod_{j\in I} p_j \prod_{m\in\overline{I}} (1-p_m) \tag{9}$$

Proof. Refer to Chen & Liu (1997) □

A.3 Proof of the integral formula

We will use the properties introduced in Appendix A.2 to prove the following lemma:

Lemma 4.2 (Poisson binomial expectation)**.** *Let* Z *be a Poisson binomial random variable of parameter* $\boldsymbol{\alpha} = (\alpha_1,\ldots,\alpha_m) \in [0,1]^m$*. Then we have:*

$$\mathbb{E}\left[\frac{1}{1+\mathrm{Z}}\right] = \int_0^1 \prod_{i=1}^{m} (1-\alpha_i t) dt$$

Proof. The PGF for Z is

$$G_{\mathrm{Z}}(t) = \mathbb{E}\left[t^{\mathrm{Z}}\right] = \prod_{i=1}^{m} (1-\alpha_i+\alpha_i t)$$

Using the Probability-Generating Function (PGF), we can write:

$$\begin{aligned}
\mathbb{E}\left[\frac{1}{1+Z}\right] &= \mathbb{E}\left[\int_0^1 t^Z dt\right] = \int_0^1 \mathbb{E}\left[t^Z\right] dt \\
&= \int_0^1 \prod_i (1-\alpha_i+\alpha_i t) dt \\
&= \int_0^1 \prod_i (1-\alpha_i t) dt && \text{change of variable: } t \leftarrow 1-t
\end{aligned}$$

Exchanging the expectation and the integral is based on the fact that the function we are integrating is measurable, non-negative, and bounded by the constant 1. We can then use Fubini's theorem to change the order of integration. □

Corollary 1. *Using the notations from lemma A.2 we can show that:*

$$\mathbb{E}\left[\frac{1}{1+Z}\right] = \sum_{i=1}^{n} \frac{(-1)^i}{1+i} \sum_{I\in\mathcal{I}_i} \prod_{j\in I} p_j,$$

where $\mathcal{I}_i$ is the set of subsets of $\{1,\ldots,n\}$ that contain exactly i elements.

This offers a straightforward approach for approximating the expectation, which is notably simpler to compute iteratively. However, it's not practical as the number of terms to sum is $n!$ and can only be implemented for small graphs.

A.4 Integral computation

Lemma 4.3 (Integral computation via quadrature)**.** *Let $m > 0$ be an integer and $c_m \overset{def}{=} \lfloor \frac{m}{2} \rfloor + 1$ (the smallest integer such that $2c_m \geq m+1$), then there exist c_m tuples $(s_q, t_q)_{1\leq q\leq c_m}$ such that:*

$$\int_0^1 \prod_{i=1}^{m} (1-p_i t) dt = \sum_{q=1}^{c_m} s_q \prod_{i=1}^{m} (1-p_i t_q),$$

for all $\boldsymbol{p} \in [0,1]^m$.

Proof. This theorem is a direct application of Gauss-Legendre quadrature (See J. Stoer (2013)). It states that any integral of the form:

$$\int_{-1}^{1} f_m(t) dt$$

where f_m is polynomial of degree at most $2m-1$ can be exactly computed using the sum:

$$\sum_{i=1}^{m} w_i f(r_i)$$

where (r_i) are the roots of P_m, Legendre polynomial of degree m, and the weights are computed as:

$$w_i = \frac{2}{(1-r_i)^2 [P_n'(r_i)]^2} \geq 0$$

For an integral on the $[0,1]$ interval, we use change of variable to find the corresponding weights and nodes are:

$$\begin{aligned}
s_j &= \frac{w_j}{2} \\
t_j &= \frac{r_j}{2} + \frac{1}{2}
\end{aligned}$$

□

A.5 Batch estimation of the expected ratio-cut

Returning to the probabilistic ratio-cut, we can combine corollary theorem 4.1 and lemma 4.2 to express the expected ratio-cut as:

$$\sum_{i=1}^{n}\sum_{j>i}^{n} w_{i,j}(p_i + p_j - 2p_ip_j)\int_0^1 \prod_{k\neq i,j}(1-p_kt)dt$$

Practically, with large datasets, it is expensive to compute the quantity $\int_0^1 \prod_{k\neq i,j}(1-p_kt)dt$ on the entire dataset: the quadrature calculation costs $\mathcal{O}(\frac{n^2}{2})$ and the total cost of computing the objective is $\mathcal{O}(kn^3)$ for all the clusters.

We can also show that we can get a tighter bound if S is a random subset such that

$$\forall i \in [1,n] P(i\in S) = \gamma$$

Theorem A.1 (Batch estimation)**.** *Let S is a random subset such that*

$$\forall i \in [1,n], P(i\in S) = \gamma < 1.$$

Then we can show that:

$$\int_0^1 \prod_{i=1}^{n}(1-p_kt)dt \leq \mathbb{E}_S\left[\int_0^1 \prod_{s\in S}(1-pst)^{\frac{1}{\gamma}}\,dt\right] \tag{10}$$

Proof.

$$\begin{aligned}
\prod_{i=1}^{n}(1-p_it) &= \exp\left[\sum_{i=1}^{n}\log(1-p_it)\right]\\
&= \exp\left[\frac{1}{\gamma}\mathbb{E}_S\left[\sum_{i\in S}\log(1-p_it)\right]\right]\\
&= \exp\left[\mathbb{E}_S\left[\sum_{i\in S}\frac{1}{\gamma}\log(1-p_it)\right]\right]\\
&\leq \mathbb{E}_S\left[\exp\left[\sum_{i\in S}\frac{1}{\gamma}\log(1-p_it)\right]\right]\\
&= \mathbb{E}_S\left[\prod_{s\in S}(1-p_st)^{\frac{1}{\gamma}}\right].
\end{aligned}$$

Theorem A.1 provides a bound on the integral based on the expectation of a random batch of samples. We can show that if $p_i \in \{0,1\}$ that the expectation over the batch equal to the estimate over the whole dataset. If we sample a random batch of size b then $\gamma = \frac{b}{n}$, the fraction of the entire data that we're using. □

A.6 Objective upper bound

Lemma 4.4 (Integral upper-bound)**.** *We adopt the same notation of Lemma 4.2 where $\boldsymbol{\alpha} = (\alpha_1,\ldots,\alpha_m) \in [0,1]^m$, and we assume that $\overline{\boldsymbol{\alpha}} \overset{def}{=} \frac{1}{m}\sum_{i=1}^{m}\alpha_i > 0$. Then we have:*

$$\int_0^1 \prod_{i=1}^{m}(1-\alpha_it)dt \leq \frac{1}{(m+1)\overline{\boldsymbol{\alpha}}}.$$

$$
\begin{aligned}
\int_0^1 \prod_{i=1}^{m}(1-\alpha_i t)dt &= \int_0^1 \exp\left[\sum_{i=1}^{m}\log(1-\alpha_i t)\right]dt \\
&= \int_0^1 \exp\left[\frac{1}{m}\sum_{i=1}^{m}\log(1-\alpha_i t)\right]^m dt \\
&\leq \int_0^1 \left[\frac{1}{m}\sum_{i=1}^{m}\exp\log(1-\alpha_i t)\right]^m dt && \text{exponential is convex} \\
&= \int_0^1 \left[\frac{1}{m}\sum_{i=1}^{m}(1-\alpha_i t)\right]^m dt \\
&= \int_0^1 (1-\overline{p}t)^m\, dt \\
&= -\frac{1}{\overline{\boldsymbol{\alpha}}(m+1)}\left[(1-\overline{\alpha}t)^{m+1}\right]_0^1 \\
&= \frac{1}{\overline{\boldsymbol{\alpha}}(m+1)}\left[1-(1-\overline{\alpha})^{m+1}\right] \\
&\leq \frac{1}{\overline{\boldsymbol{\alpha}}(m+1)}.
\end{aligned}
$$

Moving from the proof of Lemma 4.4 to Lemma 4.5 requires a modification in the previous proof. To get an upper bound without making any assumptions about $\overline{\boldsymbol{p}}$, we can prove the following:

$$
\begin{aligned}
\mathbb{I}(\boldsymbol{p},1,2) &= \int_0^1 \prod_{i\geq 3}^{n}(1-p_i t)dt \\
&\leq \int_0^1 \left[\frac{1}{n-2}\sum_{i=3}^{n}(1-p_i t)\right]^{(n-2)} dt \\
&\leq \int_0^1 \left[\frac{1}{n-2}\sum_{i=1}^{n}(1-p_i t)\right]^{(n-2)} dt \\
&= \int_0^1 \left[\frac{n}{n-2}(1-\overline{\boldsymbol{p}}t)\right]^{(n-2)} dt \\
&= \left(\frac{n}{n-2}\right)^{n-2}\frac{1}{(n-1)\overline{\boldsymbol{p}}} \\
&= \left(\frac{n}{n-2}\right)^{n-2}\frac{n}{n-1}\frac{1}{n\overline{\boldsymbol{p}}} \\
&= (e^2-o(n))\frac{1}{n\overline{\boldsymbol{p}}} \\
&< \frac{e^2}{n\overline{\boldsymbol{p}}}
\end{aligned}
$$

But by assuming that $\frac{1}{n-2}\sum_{i=3}^{n}(1-p_i t) \approx \frac{1}{n}\sum_{i=1}^{n}(1-p_i t) = \overline{\boldsymbol{p}}$, we get the bound in Lemma 4.5:

$$\begin{aligned}\frac{1}{n-1}\frac{1}{\frac{1}{n-2}\sum_{i=3}^{n} p_i} &= \frac{n-2}{n-1}\frac{1}{\sum_{i=3}^{n} p_i} \\ &\approx \frac{n-2}{n-1}\frac{1}{\sum_{i=1}^{n} p_i} \\ &\leq \frac{1}{n}\frac{1}{\overline{\boldsymbol{p}}}\end{aligned}$$

B Proofs for the stochastic gradient

B.1 Offline Gradient

The offline gradient of the objective can be obtained directly by differentiating the matrix version of the expression

$$\frac{d\mathcal{L}_{rc}}{d\boldsymbol{P}} = \overline{\boldsymbol{P}}^{-1}\left[(\mathbf{1}_{n,k}-\boldsymbol{P})^\top \boldsymbol{W}\mathbf{1}_{n,k} - \mathbf{1}_{n,k}^\top \boldsymbol{W}\boldsymbol{P}\right] - \frac{1}{n}\mathbf{1}_{n,k}\left[\overline{\boldsymbol{P}}^{-1}(\mathbf{1}_{n,k}-\boldsymbol{P})^\top \boldsymbol{W}\boldsymbol{P}\overline{\boldsymbol{P}}^{-\top}\right]$$

This expression is particularly useful in the offline learning setting, especially when $\boldsymbol{W}$ is sparse, as it makes the computation graph more efficient due to the limited support of sparse gradients in major deep learning libraries.

B.2 Online Gradient

To compute the derivative of with respect to p_i, we simply calculate the derivatives of $\frac{1}{\overline{\boldsymbol{p}}}$ and $(p_i + p_j - 2p_i p_j)$ w.r.t p_i which are:

$$\begin{aligned}\frac{d}{dp_i}\frac{1}{\overline{\boldsymbol{p}}} &= -\frac{1}{n\overline{\boldsymbol{p}}^2} \\ \frac{d}{dp_i}(p_i + p_j - 2p_i p_j) &= (1-2p_j),\end{aligned}$$

respectively.

By plugging these derivatives and considering the rule for the derivative of the product $(fg)' = f'g + fg'$ of two functions f and g, the result in Equation (8) follows.

C Reproducibility method

C.1 Experiments setup

In our sets of experiments, there are 2 variants:

- **Original representation space**, where the PRCut algorithm is used to train a neural network that takes the original as input
- **SSL representation space**, where the neural network takes as input the transformed dataset via a pre-trained self-supervised model

C.1.1 Datasets

We rely on the following datasets to benchmark our method:

MNIST MNIST (Lecun et al., 1998) is a dataset consisting of 70,000 28x28 grayscale images of handwritten digits, which are split into training (60,000) and test (10,000) samples.

Fashion MNIST Fashion-MNIST (Xiao et al., 2017) is a dataset of Zalando's article images, comprising a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image linked to a label from 10 classes. Zalando designed Fashion-MNIST to be a direct substitute for the original MNIST dataset in benchmarking machine learning algorithms. However, it introduces more complexity to machine learning tasks due to the increased difficulty in classifying the samples.

CIFAR The CIFAR-10 and CIFAR-100 datasets(Krizhevsky, 2009) is a widely used benchmark for image classification tasks in computer vision. It consists of 60,000 32x32 color images, divided into 100 classes, with 600 images per class. The images are grouped into 20 superclasses.

Key Characteristics: 60,000 images (50,000 for training and 10,000 for testing) 32x32 pixel resolution 100 classes, with 600 images per class 20 superclasses, grouping the 100 classes

C.1.2 Encoder Architecture

The encoder is a simple stack of linear layers followed each by a GeLU activation. The first layer and the last layer are always weight-normalized. The last activation is a softmax layer. The hidden unit used is constant across layers.

C.1.3 Hyperparameters

Table 5: Hyperparameters

Hyperparameter	Value
β (moving average)	0.8
γ (regularization weight)	100.0
Optimizer original space	RMSProp
Optimizer representation space	Adam
Learning rate	10^{-4}
Weight decay	10^{-7}
k (k neighbors graph)	100