Deep Spectral Clustering Learning

Marc T. Law [1] **Raquel Urtasun** [1] **Richard S. Zemel** [1 2]

Abstract

Clustering is the task of grouping a set of examples so that similar examples are grouped into the same cluster while dissimilar examples are in different clusters. The quality of a clustering depends on two problem-dependent factors which are *i)* the chosen similarity metric and *ii)* the data representation. Supervised clustering approaches, which exploit labeled partitioned datasets have thus been proposed, for instance to learn a metric optimized to perform clustering. However, most of these approaches assume that the representation of the data is fixed and then learn an appropriate linear transformation. Some deep supervised clustering learning approaches have also been proposed. However, they rely on iterative methods to compute gradients resulting in high algorithmic complexity. In this paper, we propose a deep supervised clustering metric learning method that formulates a novel loss function. We derive a closed-form expression for the gradient that is efficient to compute: the complexity to compute the gradient is linear in the size of the training mini-batch and quadratic in the representation dimensionality. We further reveal how our approach can be seen as learning spectral clustering. Experiments on standard real-world datasets confirm state-of-the-art Recall@K performance.

1. Introduction

Clustering is a widely used technique with applications in machine learning, statistics, speech processing, computer vision. It consists in grouping a set of examples so that "similar" examples are in the same cluster while "dissimilar" examples are in different clusters. In most applications, the vector representation of examples is given as input and a key step is then to determine an appropriate similarity metric so that similar and dissimilar objects can be easily identified. In some cases, experts with domain knowledge may help determine an appropriate distance metric. However, in high-dimensional problems, determining an effective metric becomes increasingly difficult even for an expert, and standard metrics such as the Euclidean distance can lead to very poor results.

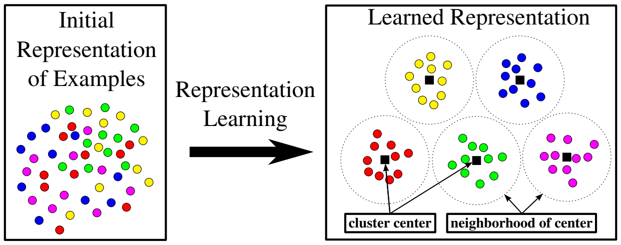

Figure 1. Our approach learns a nonlinear embedding function in a supervised way so that elements in the same category (here with the same color) are organized into the same cluster.

Many approaches that learn an appropriate similarity metric (Xing et al., 2002; Lajugie et al., 2014; Law et al., 2016) or a nonlinear embedding function (Schroff et al., 2015; Sohn, 2016; Song et al., 2017) in a supervised way have thus been proposed. They assume the availability of a training dataset that "shares the same metric" as the test dataset that they want to partition (*e.g.* the datasets represent the same concepts such as birds species or car models). As both datasets share the same metric, the model learned from training examples is expected to correctly compare test examples. The approaches can roughly be divided into four groups based on two criteria: semi-supervised/supervised setting and shallow/deep architecture. In the semi-supervised setting (Xing et al., 2002; Chopra et al., 2005), the training data is given as a small set of example pairs that are expected to be in the same or different clusters. On the other hand, supervised approaches (Lajugie et al., 2014; Law et al., 2016) assume the availability of labeled datasets for which the ground truth partitions are provided during training. A model is then learned so that some clustering algorithm will produce a partition similar to the ground truth partition on the training dataset. The supervised clustering setting can be seen as the spe-

[1] Department of Computer Science, University of Toronto, Toronto, Canada [2] CIFAR Senior Fellow. Correspondence to: Marc T. Law <law@cs.toronto.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

cial case of semi-supervised setting where all the pairwise similarity relations between training examples are given. In particular, supervised clustering is a specific classification problem where the model is learned so that the representations of training examples are closer to the representative vector of their category than to the representative vector of any other category. In this paper, we propose a novel metric learning approach whose rationale is illustrated in Fig. 1. In particular, we leverage deep nonlinear and hierarchical architectures to learn complex representations that better reflect similarity relations among examples.

Many deep metric learning approaches (Schroff et al., 2015; Song et al., 2016) extend the ideas introduced in the shallow metric learning literature (Xing et al., 2002; Weinberger et al., 2006) to learn deep neural networks on large datasets. The main difficulty is how to implement these ideas so that they are scalable and avoid memory bottleneck. For instance, many approaches have proposed hard negative mining strategies to limit the number of training constraints. A deep supervised clustering approach was proposed in (Song et al., 2017): to compute its gradient, the approach uses an iterative greedy algorithm whose algorithmic complexity is high.

Contributions: In this paper, we propose a metric learning framework that optimizes an embedding function so that the learned representations of similar examples are grouped into the same cluster, and dissimilar examples are in different clusters. To this end, we relax the problem of partitioning a dataset with Bregman divergences (Banerjee et al., 2005), and we formulate our problem so that the gradient is efficient to compute. In particular, the gradient can be expressed in closed-form, and the algorithmic complexity to compute it is linear in the size of the training mini-batch and quadratic in the representation dimensionality, which is better than the complexity of existing iterative methods. Our method is also simple to implement and obtains state-of-the-art performance on standard datasets.

2. Preliminaries

In this section, we provide some technical background about clustering, and set up the notation throughout. We reformulate in matrix form the problem of clustering a set of examples with Bregman divergences (Banerjee et al., 2005) and write it as an optimization problem *w.r.t.* one variable in Eq. (2).

Notation: We note $\langle A, B\rangle := \mathrm{tr}(AB^\top)$, the Frobenius inner product where A and B are real-valued matrices; and $\|A\| := \sqrt{\mathrm{tr}(AA^\top)}$, the Frobenius norm of A. $\mathbf{1}$ is the vector of all ones with appropriate dimensionality. $A^\dagger$ is the Moore-Penrose pseudoinverse of A. $\mathrm{ri}(\mathcal{F})$ is the relative interior of the set $\mathcal{F}$.

Data: We consider that we are given a set of n examples $\mathbf{f}_1, \cdots, \mathbf{f}_n \in \mathcal{F}$ which may be represented as a single matrix $F = [\mathbf{f}_1, \cdots, \mathbf{f}_n]^\top \in \mathcal{F}^n$. In the following, we consider that $\mathcal{F} \subseteq \mathbb{R}^d$, and thus $\mathcal{F}^n \subseteq \mathbb{R}^{n\times d}$.

Bregman divergence: Any Bregman divergence $\mathsf{d}_\phi : \mathcal{F} \times \mathrm{ri}(\mathcal{F}) \to [0, +\infty)$ is defined as $\mathsf{d}_\phi(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) - \phi(\mathbf{y}) - \langle \mathbf{x} - \mathbf{y}, \nabla\phi(\mathbf{y})\rangle$ where $\phi : \mathcal{F} \to \mathbb{R}$ is a continuously-differentiable and strictly convex function, $\nabla\phi(\mathbf{y}) \in \mathbb{R}^d$ represents the gradient vector of ϕ at $\mathbf{y}$. The most commonly used Bregman divergences for clustering are the squared Euclidean distance defined with $\phi(\mathbf{x}) = \|\mathbf{x}\|_2^2$, the KL-divergence (Dhillon et al., 2003) or the Itakura-Saito distance (Buzo et al., 1980). We refer the reader to (Banerjee et al., 2005; Nielsen & Nock, 2009) for details.

Clustering: Partitioning the n observations in $F = [\mathbf{f}_1, \cdots, \mathbf{f}_n]^\top \in \mathcal{F}^n$ into k clusters is equivalent to determining an assignment matrix $\hat{Y} \in \{0,1\}^{n\times k}$ such that $\hat{Y}_{ic} = 1$ if $\mathbf{f}_i$ is assigned to cluster c and 0 otherwise. In this paper, we assume that each example is assigned to one and only one cluster (*i.e.* the sum of each row of $\hat{Y}$ is 1) and that there is no empty cluster (which corresponds to adding the constraint $\mathrm{rank}(\hat{Y}) = k$). Therefore, $\hat{Y}$ is in the following set of assignment matrices:

$$\mathcal{Y}^{n\times k} := \{\hat{Y} \in \{0,1\}^{n\times k} : \hat{Y}\mathbf{1} = \mathbf{1}, \mathrm{rank}(\hat{Y}) = k\}$$

We assume as in (Banerjee et al., 2005) that each cluster c is represented by a single representative vector $\mathbf{z}_c \in \mathrm{ri}(\mathcal{F})$ and that an example $\mathbf{f}_i$ is assigned to cluster c only if $\forall b \neq c, \mathsf{d}_\phi(\mathbf{f}_i, \mathbf{z}_c) \leq \mathsf{d}_\phi(\mathbf{f}_i, \mathbf{z}_b)$ where d_ϕ is the chosen Bregman divergence. To simplify the notations, we consider that $\mathbf{z}_c \in \mathcal{F}$, and we concatenate the k representative vectors into a single matrix $Z = [\mathbf{z}_1, \cdots, \mathbf{z}_k]^\top \in \mathcal{F}^k$. The problem of partitioning the n examples in $F \in \mathcal{F}^n$ with d_ϕ can then be formulated as minimizing the energy function:

$$\min_{\hat{Y} \in \mathcal{Y}^{n\times k}, Z \in \mathcal{F}^k} \sum_{i=1}^{n} \sum_{c=1}^{k} \hat{Y}_{ic} \cdot \mathsf{d}_\phi(\mathbf{f}_i, \mathbf{z}_c) \tag{1}$$

$$= \min_{\hat{Y} \in \mathcal{Y}^{n\times k}, Z \in \mathcal{F}^k} \mathbf{1}^\top \Phi(F) - \mathbf{1}^\top \Phi(\hat{Y}Z) - \langle F - \hat{Y}Z, \nabla\Phi(\hat{Y}Z)\rangle$$

where we have used the definition of Bregman divergences and we note: $\forall A = [\mathbf{a}_1, \cdots, \mathbf{a}_n]^\top \in \mathcal{F}^n$, $\Phi(A) = [\phi(\mathbf{a}_1), \cdots, \phi(\mathbf{a}_n)]^\top \in \mathbb{R}^n$ is the concatenation into a single vector of the different $\phi(\mathbf{a}_i) \in \mathbb{R}$, and $\nabla\Phi(A) = [\nabla\phi(\mathbf{a}_1), \cdots, \nabla\phi(\mathbf{a}_n)]^\top \in \mathbb{R}^{n\times d}$ is the concatenation into a single matrix of the different gradients $\nabla\phi(\mathbf{a}_i) \in \mathbb{R}^d$.

Banerjee *et al.* (2005) demonstrated that, for any value of $\hat{Y} \in \mathcal{Y}^{n\times k}$, the minimizer of $\mathbf{z}_c$ in Eq. (1) is unique and is the mean vector of all the examples in F assigned to cluster c if and only if d_ϕ is a Bregman divergence. In matrix form, this means that $Z = (\hat{Y}^\top\hat{Y})^{-1}\hat{Y}^\top F = \hat{Y}^\dagger F$

is the unique global minimizer of Z in Eq. (1). We then define the set $\mathcal{P} = \{\hat{Y}\hat{Y}^\dagger : \hat{Y} \in \mathcal{Y}^{n\times k}\}$ and rewrite Eq. (1) as a minimization problem *w.r.t.* one variable:

$$\min_{\hat{C}\in\mathcal{P}} \mathbf{1}^\top \mathit{\Phi}(F) - \mathbf{1}^\top \mathit{\Phi}(\hat{C}F) - \langle F - \hat{C}F, \nabla\Phi(\hat{C}F)\rangle \quad (2)$$

As an illustration, if d_ϕ is the squared Euclidean distance, then Eq. (2) reduces to: $\|F\|^2 - \|\hat{C}F\|^2 - \langle F - \hat{C}F, 2\hat{C}F\rangle = \|F\|^2 + \|\hat{C}F\|^2 - 2\langle F, \hat{C}F\rangle = \|F - \hat{C}F\|^2$. We then obtain the formulation mentioned in (Lajugie et al., 2014)[Section 2.2] of the usual kmeans algorithm:

$$\min_{\hat{C}\in\mathcal{P}} \|F - \hat{C}F\|^2 \Leftrightarrow \max_{\hat{C}\in\mathcal{P}} \langle \hat{C}, FF^\top\rangle \quad (3)$$

by using the fact that all the matrices in $\mathcal{P}$ are orthogonal projection matrices (*i.e.* symmetric and idempotent): we then have $\forall \hat{C} \in \mathcal{P}, \|\hat{C}F\|^2 = \mathrm{tr}(\hat{C}^2FF^\top) = \mathrm{tr}(\hat{C}FF^\top)$.

3. Deep Spectral Clustering Learning

In this section, we introduce our method that we call *Deep Spectral Clustering Learning* (DSCL). We first relax the clustering problem in Eq. (2) and consider the set of solutions of the relaxed problem as the prediction function of our model. Then, we present our large margin supervised clustering problem and its efficient solver. Finally, we explain the connection of our approach with spectral clustering learning.

3.1. Constraint Relaxation

Optimizing Eq. (2) is a NP-hard problem (Aloise et al., 2009), we then approximate it. Following (Shi & Malik, 2000; Zha et al., 2001; Ng et al., 2002; Peng & Wei, 2007), we now present a spectral relaxation of this problem. In particular, we extend the domain of Eq. (2) so that the set of solutions of the resulting problem can be formulated in a convenient way.

We propose to relax the constraint $\hat{C} \in \mathcal{P}$ in Eq. (2) with the constraint $\hat{C} \in \mathcal{C}^{n,k}$ where the set $\mathcal{C}^{n,k}$ includes $\mathcal{P}$ and is defined as the set of $n \times n$ rank-k orthogonal projection matrices $\mathcal{C}^{n,k} := \{\hat{C} \in \mathbb{R}^{n\times n} : \hat{C}^2 = \hat{C}, \hat{C}^\top = \hat{C}, \mathrm{rank}(\hat{C}) = k\}$. The set of solutions of the resulting relaxed version of Eq. (2) can then be formulated:

$$f(F) := \arg\max_{\hat{C}\in\mathcal{C}^{n,k}} \mathbf{1}^\top \mathit{\Phi}(\hat{C}F) + \langle F - \hat{C}F, \nabla\Phi(\hat{C}F)\rangle \quad (4)$$

where the terms that do not depend on $\hat{C}$ are omitted.

The solutions in Eq. (4) can be found in closed-form. Let us note $s = \mathrm{rank}(F)$, we show in the supplementary material that if $s \leq k$, then the set of solutions of Eq. (4) is $f(F) = \{\hat{C} \in \mathcal{C}^{n,k} : \hat{C}F = F\} = \{\hat{C} \in \mathcal{C}^{n,k} : \hat{C} = FF^\dagger + VV^\top, VV^\top \in \mathcal{C}^{n,(k-s)}, VV^\top F = 0\}$. In particular, if $s = k$, then $VV^\top = 0$ and we have $f(F) = \{FF^\dagger\}$.

In the following, we consider only the case where the dimensionality d of training examples is not greater than k so that the property $s \leq k$ is satisfied. Nonetheless, if $s > k$, the set of solutions can be considered as $f(F) = \{FF^\dagger\}$.

3.2. Structured output prediction

In the following, we consider that we are given n examples $\mathbf{x}_i \in \mathcal{X}$ (*e.g.* n images) and an embedding function $\varphi_\theta : \mathcal{X} \to \mathcal{F}$ whose set of parameters is θ and such that $\forall i \in \{1, \cdots, n\}, \varphi_\theta(\mathbf{x}_i) = \mathbf{f}_i$ (*e.g.* $\mathbf{f}_i$ can be the output of a convolutional neural network). All these representations are concatenated into a single matrix $F = [\mathbf{f}_1, \cdots, \mathbf{f}_n]^\top \in \mathcal{F}^n$. We are also given a ground truth assignment matrix $Y \in \mathcal{Y}^{n\times k}$ which indicates the desired partition for the n examples. In other words, let $C = YY^\dagger \in \mathcal{P}$ be the ground truth clustering matrix of F, we would like to learn the embedding function φ_θ so that the matrix predicted with the (relaxed) clustering problem $f(F) \subseteq \mathcal{C}^{n,k}$ in Eq. (4) is as close as possible to the ground truth clustering matrix C.

Different evaluation metrics such as purity, rand index and normalized mutual information exist to evaluate clustering (see (Manning et al., 2008) [Chapter 16.3] for details). As in (Hubert & Arabie, 1985; Bach & Jordan, 2004; Lajugie et al., 2014; Law et al., 2016), we choose the Frobenius norm which has many advantages. As explained in (Lajugie et al., 2014)[Section 3.2], unlike rand index, the Frobenius norm between clustering (orthogonal projection) matrices takes into account the size of the different clusters (*i.e.* its value is not dominated by the performance on the largest clusters) and thus optimizes intra-class variance that is a rescaled indicator.

3.2.1. Problem formulation

In the following, we consider that the matrix F is a variable (that depends on φ_θ). The problem of minimizing the discrepancy between the ground truth clustering $C \in \mathcal{P}$ of the matrix F and the (relaxed) clustering $\hat{C}$ predicted with $f(F)$ in Eq. (4) can be formulated as the following empirical risk minimization problem:

$$\min_{F\in\mathcal{F}^n} \max_{\hat{C}\in f(F)} \|C - \hat{C}\|^2 \quad (5)$$

As all the matrices in $f(F)$ have the same rank, we can rewrite Eq. (5): $\|C - \hat{C}\|^2 = \|C\|^2 + \|\hat{C}\|^2 - 2\,\mathrm{tr}(C\hat{C})$ where $\|\hat{C}\|^2 = \mathrm{tr}(\hat{C}) = \mathrm{rank}(\hat{C}) = k$ is a constant. Eq. (5) is then equivalent to the problem:

$$\max_{F\in\mathcal{F}^n} \min_{\hat{C}\in f(F)} \mathrm{tr}(C\hat{C}) \quad (6)$$

Algorithm 1 Deep Spectral Clustering Learning (DSCL)

```
input : Set of training examples (e.g. images) in $\mathcal{X}$, embedding function $\varphi_\theta$, number of iterations $\tau$, learning rates $\eta_1, \cdots, \eta_\tau$
for iteration $t = 1$ to $\tau$ do
    Randomly sample $n$ training examples $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathcal{X}$
    Create representation matrix $F \leftarrow [\mathbf{f}_1, \cdots, \mathbf{f}_n]^\top \in \mathcal{F}^n$ s.t. $\forall i \in \{1, \cdots, n\}, \mathbf{f}_i = \varphi_\theta(\mathbf{x}_i)$
    Create rescaled gradient $G \leftarrow (I - FF^\dagger)C(F^\dagger)^\top$ (see Eq. (9)) where $C = YY^\dagger$ is the desired partition matrix of the $n$ examples
    Update the set of parameters $\theta$ of $\varphi_\theta$ by exploiting the rescaled gradient $G$ and perform gradient step with learning rate $\eta_t$
end for
```

which is naturally lower bounded by (see details in supplementary material):

$$\max_{F \in \mathcal{F}^n} \min_{\hat{C} \in f(F)} \operatorname{tr}(C\hat{C}FF^\dagger) = \max_{F \in \mathcal{F}^n} \operatorname{tr}(CFF^\dagger) \tag{7}$$

where we use the fact that $\forall \hat{C} \in f(F), \hat{C}FF^\dagger = FF^\dagger$. We note that Eq. (7) equals Eq. (6) if $\operatorname{rank}(F) = k$.

The difficulty of optimizing the problem in Eq. (7) is that it depends on both F and $F^\dagger$. If we assume that $\operatorname{rank}(F)$ is a constant (with F not necessarily full rank) in Eq. (7), then the problem is differentiable (Golub & Pereyra, 1973) and the gradient of Eq. (7) *w.r.t.* F is:

$$\nabla_F = 2(I - FF^\dagger)C(F^\dagger)^\top \tag{8}$$

where I is the identity matrix. Details on the gradient can be found in the supplementary material. Our matrix F is always full rank in our experiments[1], so the constant rank condition along the iterations is satisfied.

3.2.2. Low algorithmic complexity

We now show that in addition to having a closed-form expression, computing our gradient ∇_F is efficient: the complexity to compute it is linear in n and quadratic in d.

We note $C = YY^\dagger \in \mathcal{P}$ the ground truth partition matrix where the matrix $Y \in \mathcal{Y}^{n \times k}$ is given: let $\mathbf{y}_c$ be the c-th column of Y, then the c-th column of $(Y^\dagger)^\top$ can be written $\frac{1}{\max\{1, \mathbf{y}_c^\top \mathbf{1}\}}\mathbf{y}_c$. The complexity to compute $(Y^\dagger)^\top$ is linear in n due to the sparsity of Y. $(Y^\dagger)^\top$ is also sparse (*i.e.* it contains n nonzero elements). We can then write $\frac{\nabla_F}{2}$ as:

$$G := \frac{\nabla_F}{2} = \left(Y - F[F^\dagger Y]\right)[F^\dagger(Y^\dagger)^\top]^\top \tag{9}$$

where $[\cdot]$ indicates $d \times k$ matrices which are computed efficiently due to the sparsity of Y. The complexity to compute $F^\dagger$ is $O(nd\min\{n, d\})$ (*i.e.* $O(nd^2)$ as we assume $d \leq n$).

3.2.3. Direct loss minimization

Our method computes in closed-form the gradient of the structured output prediction problem in Eq. (5) when $\operatorname{rank}(F) = k$, and its algorithmic complexity is low.

To the best of our knowledge, although we exploit classic spectral relaxation results, our method is the first approach that includes the closed-form solution of the relaxed kmeans problem (see Section 3.1) within a large margin method for structured output. Even Mahalanobis metric learning methods (Lajugie et al., 2014) cannot use such a simplification due to the nature of their model, and the formulation of their subgradient is then different.

Including the closed-form solution of the relaxed kmeans algorithm results in a simple problem formulation (see Eq. (7)). The resulting large margin optimization problem is easy to optimize as we do not have to perform loss-augmented inference during training.

3.3. Learning deep models

Our method can be used to learn neural networks with conventional gradient-based methods by exploiting chain rule. Our approach is illustrated in Algorithm 1.

In detail, we note a mini-batch matrix $F = [\mathbf{f}_1, \cdots, \mathbf{f}_n]^\top \in \mathcal{F}^n$ the concatenation into a single matrix of the n different embedding representations $\varphi_\theta(\mathbf{x}_i) = \mathbf{f}_i$, where, for instance, $\varphi_\theta : \mathcal{X} \to \mathcal{F}$ is a neural network embedding function and $\mathbf{x}_i \in \mathcal{X}$ is an image. We can rewrite Eq. (7) as a minimization problem by defining our loss function as a function of the mini-batch representation matrix:

$$\operatorname{Loss}(F) := k - \operatorname{tr}(CFF^\dagger) \tag{10}$$

which is nonnegative and where $C = YY^\dagger \in \mathcal{P}$ is the ground truth clustering matrix of F. The neural network φ_θ is then learned via backpropagation as illustrated in Algorithm 1 (*i.e.* using stochastic gradient descent with rescaled gradient $-G$ where G is defined in Eq. (9)).

We note that the structure of the learned neural network is not limited to the case where $\mathcal{F}$ is $\mathbb{R}^d$. For instance, if the goal is to partition probability distributions with KL-divergence, the set $\mathcal{F}$ can be constrained to be the d-dimensional simplex $\{\mathbf{x} \in [0,1]^d : \mathbf{x}^\top \mathbf{1} = 1\}$ by using a softmax regression at the last layer of the neural network.

Regression problem: It is worth noting that ∇_F is also the gradient *w.r.t.* F of the nonlinear least squares problem:

$$\max_{F \in \mathcal{F}^n} -\frac{1}{2}\|FF^\dagger - C\|^2 \tag{11}$$

[1] In our experiments, since the number of rows of F is greater than its number of colums, F has full column rank.

where we assume that the rank of F is constant (otherwise, the problem is not differentiable). Our solver can then be seen as a gradient-based solver for the nonlinear regression problem that iteratively decreases the distance $\|FF^\dagger - C\|$. In particular, the spectral relaxation proposed in Section 3.1 allows to write the set of predicted clusterings $f(F)$ as a function of $FF^\dagger$, and the choice of the Frobenius norm to compare clusterings makes our problem similar to a least squares problem.

Given the least squares formulation of Eq. (11), one can see that our problem focuses more on the similarity between the matrices C and $FF^\dagger$ than between C and the individual matrix F. Our problem can then be seen as a spectral clustering algorithm as explained in the following.

3.4. Connection with spectral clustering

We now explain how our proposed method can be seen as learning spectral clustering in a supervised way.

As explained in (Bach & Jordan, 2004; Von Luxburg, 2007), spectral clustering does not refer to one particular method but to a family of methods (*e.g.* (Shi & Malik, 2000)) that partition a dataset by exploiting the leading eigenvectors of a similarity matrix. They rely on the eigenstructure of a similarity matrix to partition examples into disjoint clusters, with examples in the same cluster having high similarity and examples in different clusters having low similarity. In our case, as can be seen in Eq. (7), the (kernel) similarity matrix is $K = FF^\dagger = UU^\top$ where $U \in \mathbb{R}^{n \times s}$ is a matrix whose columns are the left-singular vectors of F corresponding to its nonzero singular values and where $s = \operatorname{rank}(F)$. By definition, these left-singular vectors of F are also the s leading eigenvectors of K.

It is worth noting that our approach is different from classic spectral clustering approaches that perform clustering by exploiting some Laplacian matrix of the similarity matrix. Our approach directly performs clustering by exploiting the leading eigenvectors of the similarity matrix K.

By increasing the value $\operatorname{tr}(CFF^\dagger) = \operatorname{tr}(CUU^\top)$ at each backpropagation iteration, the leading eigenvectors of $UU^\top = K$ become *more similar* to the leading eigenvectors of C. As C and $K = UU^\top$ are both orthogonal projection matrices, in the ideal case, $\operatorname{tr}(CUU^\top)$ is maximized when the column space of one of the two matrices C or $(UU^\top)$ is included in the column space of the other matrix. In particular, if $\operatorname{rank}(C) = \operatorname{rank}(UU^\top)$, then $\operatorname{tr}(CUU^\top)$ is maximized iff $C = UU^\top$ (Fan, 1949). In this ideal case, we have $\arg\max_{\hat{C} \in \mathcal{P}} \langle \hat{C}, UU^\top \rangle = \{C\}$, which corresponds to the solution of Eq. (3) when replacing F by U; in other words, partitioning the rows of U with kmeans will return the desired clustering matrix C.

It is then clear that comparing the similarity between the rows of U (*i.e.* using spectral clustering) is at least as relevant as comparing the rows of F to partition the dataset. In our experiments, our method obtains better performance when the left-singular vectors of the test set matrix are used for partitioning rather than the learned features.

4. Experiments

The goal of metric learning is to learn a metric that can be used to compare new examples, possibly from categories that were not in the training dataset. In this context, the model is learned on a training dataset and tested on a dataset that shares the same "semantical" metric and represents similar concepts. This allows to evaluate the generalization ability of the model. Following the experimental protocol described in (Song et al., 2016; 2017), we evaluate our method on the following fine-grained datasets and use the exact same train/test splits:

• The Caltech-UCSD Birds (CUB-200-2001) dataset (Wah et al., 2011) is composed of 11,788 images of birds from 200 different species/categories. We split the first 100 categories for training (5,864 images) and the rest for test (5,924 images).

• The CARS196 dataset (Krause et al., 2013) is composed of 16,185 images of cars from 196 model categories. The first 98 categories are used for training (8,054 images), the rest for test (8,131 images).

• The Stanford Online Product (Song et al., 2016) dataset is composed of 120,253 images from 22,634 online product categories. It is partitioned into 59,551 images from 11,318 categories for training and 60,502 images from 11,316 categories for test.

In these experiments, the categories for training and the categories for testing are disjoint although they belong to the same context (*i.e.* they all represent birds, cars or products). This makes the problem challenging as deep models may *overfit* on the training categories[2]. In the same way as the baselines, we then perform early stopping.

4.1. Implementation details

We closely follow the experimental setup described in (Song et al., 2016; 2017). In particular, we implemented our method with the Tensorflow package (Abadi et al., 2016) and used the Inception (Szegedy et al., 2015) network with batch normalization (Ioffe & Szegedy, 2015) pretrained on ImageNet/ILSVRC 2012-CLS (Russakovsky et al., 2015), we fine-tuned the network on the training datasets. We perform two types of fine-tuning:

[2] For instance, our model obtains more than 90% performance for all the evaluation metrics on the training categories after 1000 iterations.

Algorithm 2 DSCL Normalized Spectral Clustering

```
input : Test set $F_t \in \mathcal{F}^{n_t}$, $n_t$ is the number of test examples
Create $M_t \in \mathbb{R}^{n_t \times d}$ by mean centering $F_t$
Create $r = \text{rank}(M_t)$
Create $U_t = [\mathbf{u}_1, \cdots, \mathbf{u}_{n_t}]^\top \in \mathbb{R}^{n_t \times r}$ s.t. $M_t M_t^\dagger = U_t U_t^\top$
Create $T = [\mathbf{t}_1, \cdots, \mathbf{t}_{n_t}]^\top \in \mathbb{R}^{n_t \times r}$ s.t. $\forall i, \mathbf{t}_i = \frac{1}{\|\mathbf{u}_i\|_2}\mathbf{u}_i$
partition the rows of $T$ into $k$ clusters with kmeans
```

- **end-to-end:** we fine-tune the model and update the parameters of all the layers of the neural network during backpropagation. In this case, we perform 100 iterations of gradient descent.

- **last layer:** we freeze all the parameters of the neural network (pretrained on ImageNet) except those in the last layer. Only the parameters in the last layer of the model are updated. We perform 200 iterations of gradient descent.

In both cases, we remove the softmax function at the end of the last layer.

The images are first resized to square size (256 × 256) and cropped at 227 × 227. For the dataset augmentation, we use a random horizontal mirroring for training and a single center crop for test. As in (Song et al., 2017) and unlike (Sohn, 2016), we use a single crop per image.

We ran our experiments on a single Tesla P100 GPU with 16GB RAM, and used a standard Stochastic Gradient optimizer. Our batch size is set to $n = o \times p = 18 \times 70 = 1260$, our method backpropagates the loss in Eq. (10) for all the examples in the batch. As Inception is a large model and to fit into memory, we iteratively compute submatrices $F_i \in \mathcal{F}^o$ and concatenate them into a single matrix $F = [F_1, \cdots, F_p]^\top \in \mathcal{F}^n$. Our method then computes the gradient matrix $G = [G_1, \cdots, G_p]^\top \in \mathbb{R}^{n \times d}$ defined in Eq. (9), and minimizes the loss function $\langle F, -G \rangle = \sum_{i=1}^{p} \langle F_i, -G_i \rangle$ with gradient descent where $-G$ is fixed.

Different embedding dimensionality values $d \in \{64, 128, 256, 512\}$ are tested in (Song et al., 2016), it is reported that d does not play a crucial role. In our case, we then set our dimensionality d to be equal to the number of categories k for the Birds and Cars datasets.

For the Products dataset which contains more than 10k categories, we observed as in (Sohn, 2016) that the larger the value of d, the better the results. We then set $d = 512$ in order to be fair with the other models and randomly subsample 512 training categories.

4.2. Partitioning a test dataset

Partitioning a test dataset $X_t = [\mathbf{x}_1, \cdots, \mathbf{x}_{n_t}]^\top \in \mathcal{X}^{n_t}$ (where $\mathbf{x}_i \in \mathcal{X}$ is an image in these experiments, and n_t is the number of test examples) is done by first computing the test representation matrix $F_t = [\varphi_\theta(\mathbf{x}_1), \cdots, \varphi_\theta(\mathbf{x}_{n_t})]^\top \in \mathcal{F}^{n_t}$ where φ_θ is the learned embedding function, and then applying a partition algorithm. The partition algorithm can either be a standard clustering algorithm as described in (Banerjee et al., 2005), or can exploit the connection of our method with spectral clustering as explained in Section 3.4.

The spectral clustering (SC) algorithm that we use, which is inspired by (Ng et al., 2002), is illustrated in Algorithm 2: we first mean center F_t (the mean of each column of the resulting matrix M_t is zero), then we extract the matrix U_t that contains the leading left-singular vectors of the resulting matrix M_t, the rows of U_t are then ℓ_2-normalized and partitioned with the usual kmeans algorithm.

To test our method without spectral clustering, we ℓ_2-normalize the output representations as done in (Song et al., 2017) and then apply the usual kmeans algorithm with the squared Euclidean distance.

4.3. Quantitative results

We compare our method to current state-of-the-art metric learning approaches in Tables 1 to 3 by evaluating the Normalized Mutual Information (NMI) and Recall@K performances. In particular, Recall@1 is a useful metric in zero-shot learning contexts, it allows to assign the category of a test image to the category of its nearest neighbor, it then evaluates the generalization performance of a model to new similar concepts. The scores for the following baselines (Schroff et al., 2015; Song et al., 2016; Sohn, 2016; Song et al., 2017) are reported from (Song et al., 2017) where the methods are tested in a similar setup. As explained in the related work (Section 5), the NMI-based approach (Song et al., 2017) is the only baseline that is explicitly learned to optimize a clustering criterion.

We also report the performance of the vanilla GoogLeNet/Inception features pretrained on ImageNet, and GoogLeNet features fine-tuned for classification with softmax regression. In both cases, we report the results obtained with Spectral Clustering (SC) as illustrated in Algorithm 2, and without SC (as explained in the last paragraph of Section 4.2). We report the scores for ℓ_2-normalized features as they obtain better performance than unnormalized features in our experiments.

Recognition performance: Our spectral clustering approach obtains state-of-the-art Recall@K performance on all the datasets. Only the method in (Song et al., 2017) obtains better NMI performance on Birds and Products, this can be expected as the model in (Song et al., 2017) is specifically learned to optimize the NMI evaluation metric. Our choice to optimize the Frobenius norm which allows to compute a gradient in closed-form then seems to be a good trade-off for scalability as it obtains competitive NMI results and state-of-the-art Recall@K performance.

Table 1. NMI and Recall@K evaluation on the Birds (CUB-200-2011) dataset

Method	NMI	R@1	R@2	R@4	R@8
Triplet s.h. (Schroff et al., 2015)	55.38	42.59	55.03	66.44	77.23
Lifted struct (Song et al., 2016)	56.50	43.57	56.55	68.59	79.63
Npairs (Sohn, 2016)	57.24	45.37	58.41	69.51	79.49
NMI-based (Song et al., 2017)	**59.23**	48.18	61.44	71.83	81.92
Vanilla GoogLeNet (with SC)	53.53	42.56	55.55	67.86	78.39
Vanilla GoogLeNet (without SC)	50.28	40.11	53.17	66.02	76.59
Softmax regression (with SC)	55.91	45.49	58.64	70.65	79.17
Softmax regression (without SC)	55.74	46.00	58.03	69.32	78.28
Ours (end-to-end/with SC)	58.12	49.78	62.56	73.55	82.78
Ours (end-to-end/without SC)	56.99	47.57	59.66	71.57	81.28
Ours (last layer/with SC)	59.16	**53.22**	**66.09**	**76.70**	**85.33**
Ours (last layer/without SC)	56.87	50.08	62.24	73.38	82.38

Table 2. NMI and Recall@K evaluation on the Cars196 dataset

Method	NMI	R@1	R@2	R@4	R@8
Triplet s.h. (Schroff et al., 2015)	53.35	51.54	63.78	73.52	82.41
Lifted struct (Song et al., 2016)	56.88	52.98	65.70	76.01	84.27
Npairs (Sohn, 2016)	57.79	53.90	66.76	77.75	86.35
NMI-based (Song et al., 2017)	59.04	58.11	70.64	80.27	87.81
Vanilla GoogLeNet (with SC)	44.53	35.37	47.32	60.60	71.69
Vanilla GoogLeNet (without SC)	47.99	35.56	47.27	59.37	72.16
Softmax regression (with SC)	53.13	50.11	60.49	71.68	80.14
Softmax regression (without SC)	52.13	48.75	58.52	70.97	78.37
Ours (end-to-end/with SC)	58.04	59.37	71.25	80.62	88.32
Ours (end-to-end/without SC)	56.08	57.08	69.23	79.39	87.46
Ours (last layer/with SC)	**64.25**	**73.07**	**82.19**	**89.01**	**92.99**
Ours (last layer/without SC)	61.12	67.54	77.77	85.74	90.95

Table 3. NMI and Recall@K evaluation on the Products (Stanford Online Products) dataset

Method	NMI	R@1	R@10	R@100
Triplet s.h. (Schroff et al., 2015)	89.46	66.67	82.39	91.85
Lifted struct (Song et al., 2016)	88.65	62.46	80.81	91.93
Npairs (Sohn, 2016)	89.37	66.41	83.24	93.00
NMI-based (Song et al., 2017)	**89.48**	67.02	83.65	**93.23**
Vanilla GoogLeNet (with SC)	56.01	44.09	61.32	77.82
Vanilla GoogLeNet (without SC)	55.32	43.73	60.84	76.54
Softmax regression (with SC)	63.53	51.95	69.83	85.36
Softmax regression (without SC)	63.10	51.65	69.75	85.32
Ours (end-to-end/with SC)	89.40	**67.59**	**83.71**	**93.25**
Ours (end-to-end/without SC)	88.70	64.52	81.53	92.35
Ours (last layer/with SC)	88.75	65.30	81.50	92.40
Ours (last layer/without SC)	86.95	64.90	78.05	91.50

The usual kmeans algorithm applied on the representations of our learned model obtains worse results than our proposed spectral clustering (SC) approach. It obtains better performance than most baselines on Birds and Cars, but also poor results on the Products dataset. This may be explained by the fact that there are only 5.3 images per category in this dataset, which is 10 times less than for the other datasets. Some categories also contain only 2 images, which is hard to generalize on. Our approach is then more appropriate when the categories are large (*i.e.* more than 50 images) than when there are many (very) small clusters.

We also note that when our model updates only the last layer during fine-tuning, it obtains state-of-the-art performance on Birds and Cars, but not as good as our fully learned model on the Products dataset. The strong results on the former two is likely due to their small size (fewer than 10k training images); which makes learning all the layers prone to overfitting. Learning in all layers seems beneficial on larger datasets such as Products.

We observe that most baselines (Schroff et al., 2015; Sohn, 2016; Song et al., 2017) and our spectral method obtain comparable results on the Products dataset. There is then not a clear way to learn deep models in contexts with small categories. On the other hand, there is a huge gap in Recall@K performance on the other datasets between approaches that optimize clustering and approaches that do not. The largest performance gap is observed on the Cars dataset which contains the largest number of images per category (more than 80).

It is also worth noting that unlike usual softmax regression for classification that promotes centroids to be one-hot vectors in the ideal case, our approach takes as input the current representations and tries to group similar examples together without fixing the desired centroids. It can then be easily combined with other approaches.

Training time: Once the matrix representation of the mini-batch $F \in \mathcal{F}^n$ has been computed, computing the gradient G takes 1 second (with $n = 1280$ and $d = 100$). Since we backpropagate our loss for all the examples in the mini-batch, each iteration takes about 50 seconds due to the large architecture of Inception. Nevertheless, multiple GPUs can be used in parallel to speed up training.

Qualitative results: t-SNE (Van Der Maaten, 2014) plots are available in the supplementary material.

5. Related work

As explained in Section 1, many approaches have been proposed to learn a similarity metric (Xing et al., 2002; Bar-Hillel et al., 2005; Lajugie et al., 2014; Law et al., 2016; 2017b) or a nonlinear embedding function (Schroff et al., 2015; Song et al., 2017) optimized to perform clustering. However, most of them belong to the semi-supervised setting (Xing et al., 2002; Bar-Hillel et al., 2005; Schroff et al., 2015), *i.e.* they are designed to learn from small sets of pairwise or triplet-wise relations and do not account for the global clustering performance on the training dataset. In pairwise approaches, the model is given pairs of examples $(\mathbf{x}_i, \mathbf{x}_j)$ which are either similar (*e.g.* the examples are in the same category) or dissimilar (*e.g.* the examples are in different categories), the model is then learned so that the distances between representations of similar objects are smaller than the distances between dissimilar objects.

In the triplet-wise approach (Schultz & Joachims, 2003; Weinberger et al., 2006), triplets of examples $(\mathbf{x}_i, \mathbf{x}_i^+, \mathbf{x}_i^-)$ are provided and the model is learned so that the distance between the representations of the pair $(\mathbf{x}_i, \mathbf{x}_i^+)$ is smaller than for the pair $(\mathbf{x}_i, \mathbf{x}_i^-)$. We focus on the supervised clustering setting which considers all the possible similar and dissimilar pairs and takes into account the global clustering structure of the training dataset (here a mini-batch).

In the shallow metric learning literature, (Lajugie et al., 2014; Law et al., 2016) learn a linear transformation in the supervised clustering setting so that the partition obtained when using kmeans on a dataset is as close as possible to the desired partition. However, they both consider that the data representation is fixed and are limited by the complexity of their linear model. Moreover, their solvers are very different from ours. Lajugie et al. (2014) propose an extension of the structural SVM (Tsochantaridis et al., 2005) for Mahalanobis distances, and their projected subgradient method thus has high complexity. Law et al. (2016) propose a closed-form solver but the method is limited to the case where there is a single training dataset and they do not provide gradient-based strategies to optimize other types of models such as neural networks.

In the deep learning literature, Song et al. (2016) proposed to approximate a loss function that considers all the positive and negative pairs. To this end, they iteratively randomly sample a few similar pairs, and then actively add their difficult neighbors to the training mini-batch. This idea is similar to the idea of active set of constraints used in (Weinberger & Saul, 2009; Law et al., 2017a) to limit the number of active constraints (*i.e.* the number of constraints that have nonzero subgradients) and be able to optimize over large numbers of triplets. Although their approach considers most of the similarity relations, it is not optimized to group all the similar examples into the same unique cluster. Indeed, each category can be divided in multiple subclusters as explained in (Song et al., 2017).

Sohn (2016) proposed to tackle the problem of slow convergence of triplet-wise approaches (caused by hard negative mining) by optimizing losses over $(n + 1)$-tuplets. In particular, an efficient batch construction method is proposed to require $2n$ examples instead of the naïve $(n + 1)n$ to build n tuplets of length $(n + 1)$. The loss function recruits multiple distances between dissimilar examples from different categories and approximates the ideal loss that minimizes the distances between similar examples while maximizing the distances between dissimilar examples. This approach considers the global structure of the representation of mini-batches better than triplet-wise approaches. However, although it does take into account most distances in the mini-batch, it does not explicitly optimize the model so that it obtains good clustering performance.

In the deep metric learning literature, the most similar approach to ours is (Song et al., 2017). They select for each category one unique example that will be the medoid (*i.e.* representative example). The choice of the medoids is not discussed and may be problematic if there is noise in the labels. In contrast, our centroids are learned so that they are the mean vectors of the training examples. Indeed, our set of centroids is written $Z = \hat{Y}^\dagger F$ where $F \in \mathcal{F}^n$ is the representation of our mini-batch and $\hat{Y}$ is implicitly included in the formulation of the set $\mathcal{C}^{n,k}$. Our optimization problem then learns a data representation such that training examples are projected close to their respective centroids. Moreover, the gradient in (Song et al., 2017) requires an iterative greedy algorithm and each iteration of their greedy algorithm has higher complexity than the complexity of computing our gradient (we set the dimensionality d of our learned representations to be k). Each iteration of the greedy algorithm is linear in the size of the mini-batch and cubic in the number of categories in the mini-batch.

Deep learning was also used in (Ionescu et al., 2015) to learn a convolutional neural network optimized for clustering. However, it is applied to unsupervised image segmentation with normalized cuts (Shi & Malik, 2000). We are interested in this paper in the supervised clustering setting where the ground truth partition is provided. Another deep clustering approach was proposed in (Hershey et al., 2016). However, their model is not optimized to be robust to the size of the different clusters as discussed in Section 3.2.

6. Conclusion

We have presented a novel deep learning approach optimized for the supervised clustering task. Our method is simple to implement and scalable thanks to its low algorithmic complexity. It also obtains state-of-the-art recall@K performance on different standard fine-grained datasets. Future work includes improving our proposed solver by exploiting the results in (Ionescu et al., 2015) which take into account the structure of the neural network instead of using a standard stochastic gradient descent solver.

Acknowledgments: We thank David Duvenaud, Stavros Tsogkas, Dimitris Vlitas and the anonymous reviewers for their helpful comments. This work was supported by Samsung and the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/Interior Business Center (DoI/IBC) contract number D16PC00003. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/IBC, or the U.S. Government.

References

Abadi, Martín, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S, Davis, Andy, Dean, Jeffrey, Devin, Matthieu, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Aloise, Daniel, Deshpande, Amit, Hansen, Pierre, and Popat, Preyas. Np-hardness of euclidean sum-of-squares clustering. *Machine learning*, 75(2):245–248, 2009.

Bach, Francis and Jordan, Michael. Learning spectral clustering. *NIPS*, 16:305–312, 2004.

Banerjee, Arindam, Merugu, Srujana, Dhillon, Inderjit S, and Ghosh, Joydeep. Clustering with bregman divergences. *Journal of machine learning research*, 6(Oct): 1705–1749, 2005.

Bar-Hillel, Aharon, Hertz, Tomer, Shental, Noam, and Weinshall, Daphna. Learning a Mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6(6):937–965, 2005.

Buzo, Andrés, Gray, A, Gray, R, and Markel, John. Speech coding based upon vector quantization. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28 (5):562–574, 1980.

Chopra, Sumit, Hadsell, Raia, and LeCun, Yann. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pp. 539–546. IEEE, 2005.

Dhillon, Inderjit S, Mallela, Subramanyam, and Kumar, Rahul. A divisive information-theoretic feature clustering algorithm for text classification. *Journal of machine learning research*, 3(Mar):1265–1287, 2003.

Fan, Ky. On a theorem of weyl concerning eigenvalues of linear transformations i. *Proceedings of the National Academy of Sciences of the United States of America*, 35 (11):652, 1949.

Golub, Gene H and Pereyra, Victor. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM Journal on numerical analysis*, 10(2):413–432, 1973.

Hershey, John R, Chen, Zhuo, Le Roux, Jonathan, and Watanabe, Shinji. Deep clustering: Discriminative embeddings for segmentation and separation. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pp. 31–35. IEEE, 2016.

Hubert, Lawrence and Arabie, Phipps. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Ionescu, Catalin, Vantzos, Orestis, and Sminchisescu, Cristian. Matrix backpropagation for deep networks with structured layers. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2965–2973, 2015.

Krause, Jonathan, Stark, Michael, Deng, Jia, and Fei-Fei, Li. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 554–561, 2013.

Lajugie, R., Bach, F., and Arlot, S. Large margin metric learning for constrained partitioning problems. *Proc. International Conference on Machine Learning*, 2014.

Law, Marc Teva, Yu, Yaoliang, Cord, Matthieu, and Xing, Eric Poe. Closed-form training of mahalanobis distance for supervised clustering. In *CVPR*. IEEE, 2016.

Law, Marc Teva, Thome, Nicolas, and Cord, Matthieu. Learning a distance metric from relative comparisons between quadruplets of images. *IJCV*, 121(1): 65–94, 2017a. ISSN 1573-1405. doi: 10.1007/s11263-016-0923-4. URL http://dx.doi.org/10.1007/s11263-016-0923-4.

Law, Marc Teva, Yu, Yaoliang, Urtasun, Raquel, Zemel, Richard, and Xing, Eric Poe. Efficient multiple instance metric learning using weakly supervised data. In *Computer Vision and Pattern Recognition (CVPR)*, 2017b.

Manning, Christopher D, Raghavan, Prabhakar, Schütze, Hinrich, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

Ng, Andrew Y, Jordan, Michael I, Weiss, Yair, et al. On spectral clustering: Analysis and an algorithm. In *NIPS*, volume 14, pp. 849–856, 2002.

Nielsen, Frank and Nock, Richard. Sided and symmetrized bregman centroids. *IEEE transactions on Information Theory*, 55(6):2882–2904, 2009.

Peng, J. and Wei, Y. Approximating k-means-type clustering via semidefinite programming. *SIAM Journal on Optimization*, 18:186–205, 2007.

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, et al.

Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Schroff, Florian, Kalenichenko, Dmitry, and Philbin, James. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815–823, 2015.

Schultz, Matthew and Joachims, Thorsten. Learning a distance metric from relative comparisons. In *NIPS*, volume 1, pp. 2, 2003.

Shi, Jianbo and Malik, Jitendra. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

Sohn, Kihyuk. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, pp. 1849–1857, 2016.

Song, Hyun Oh, Xiang, Yu, Jegelka, Stefanie, and Savarese, Silvio. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4004–4012, 2016.

Song, Hyun Oh, Jegelka, Stefanie, Rathod, Vivek, and Murphy, Kevin. Deep metric learning via facility location. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, and Rabinovich, Andrew. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

Tsochantaridis, Ioannis, Joachims, Thorsten, Hofmann, Thomas, and Altun, Yasemin. Large margin methods for structured and interdependent output variables. *Journal of machine learning research*, 6(Sep):1453–1484, 2005.

Van Der Maaten, Laurens. Accelerating t-sne using tree-based algorithms. *Journal of machine learning research*, 15(1):3221–3245, 2014.

Von Luxburg, Ulrike. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

Wah, Catherine, Branson, Steve, Welinder, Peter, Perona, Pietro, and Belongie, Serge. The caltech-ucsd birds-200-2011 dataset. 2011.

Weinberger, Kilian Q and Saul, Lawrence K. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 10:207–244, 2009.

Weinberger, Kilian Q, Blitzer, John, and Saul, Lawrence. Distance metric learning for large margin nearest neighbor classification. *Advances in neural information processing systems*, 18:1473, 2006.

Xing, Eric P, Jordan, Michael I, Russell, Stuart, and Ng, Andrew Y. Distance metric learning with application to clustering with side-information. In *NIPS*, pp. 505–512, 2002.

Zha, Hongyuan, He, Xiaofeng, Ding, Chris, Gu, Ming, and Simon, Horst D. Spectral relaxation for k-means clustering. In *NIPS*, pp. 1057–1064, 2001.