Learning Feature Representations for Discriminative Clustering with Limited Supervision

Corinne Jones, Vincent Roulet, Zaid Harchaoui

Technical Report no. 651
Department of Statistics
University of Washington

June 25, 2019

Abstract

We introduce an approach for discriminative clustering that allows one to jointly learn the metric or similarity from unlabeled data (with or without labeled data) and learn the clustering assignment. The metric or similarity is assumed to be specified in a differentiable programming framework, that is, a layer defined as a parameterized functional amenable to automatic differentiation. We define a discriminative clustering layer that enjoys better conditioning than the more commonly used k-means layer. The proposed approach can be used for any amount of labeled or unlabeled data, gracefully adjusting to the amount of supervision. We present experimental results assessing the effectiveness of our method.

1 Introduction

Deep networks trained end-to-end now produce state-of-the-art results on a wide variety of tasks, from lung cancer screening to music transcription to pose estimation [1, 37, 28]. These advances have come from a wide variety of architectures, including various convolutional networks and recurrent networks [17]. However, one common theme is that in order to achieve these top results the models need to be trained on vast quantities of labeled data. Unfortunately, for most tasks only a small amount of labeled data exists. However, there often exist vast quantities of unlabeled data that are left untouched.

Recent work exploits this unlabeled data in a variety of ways to learn deep feature representations, either with unsupervised or semi-supervised methods [10, 33, 8]. While these methods are often effective, we would ideally like to have a single method that works regardless of the quantity of labeled data. No previous work presents a clear, principled framework that addresses end-to-end learning with any level of supervision. The crux of learning with unlabeled data is to ensure that it does not result in trivial solutions. Specifically, the algorithm must avoid assigning all points to the same class and must also avoid mapping all of the raw features to the same embedded feature vector. In the past these problems were often dealt with by solutions such as randomly creating a new cluster when a cluster becomes empty.

In this work we propose a unified learning framework that naturally reduces to deep clustering in the case of unsupervised learning and traditional end-to-end learning in the case of supervised learning. We compare our objective function to the main alternative and show that ours is generally smoother. To train our objective function we develop an effective algorithm that simultaneously learns the parameters of a deep network and estimates the labels of the unlabeled data. We avoid trivial solutions by enforcing cluster balancing constraints and penalizing the norm of the centered embeddings. Using this framework, we present results demonstrating the benefit of our algorithm.

2 Related Work

This paper develops an end-to-end semi-supervised learning algorithm that learns deep feature representations for clustering. Two general approaches to developing semi-supervised methods exist: starting from a supervised algorithm and modifying it to use unlabeled data; and starting from an unsupervised algorithm and modifying it to use labeled data.

Unsupervised learning. Most unsupervised deep feature learning methods can be broadly classified into one of two categories: methods that optimize a surrogate loss, often based on known structure in the data; and methods that directly optimize a loss function of interest. Early examples of the former set of methods include autoencoders, which attempt to reconstruct the input observations [25, 17]. Other more recent examples attempt to approximate a kernel at each layer of a network [5, 29, 12]. Most recently, many papers have been taking advantage of structure in the data. This includes training to distinguish between multiple views of images or patches and other images or patches [14, 39, 36], learning to predict the relative location of patches in images [13, 31], and predicting color from grayscale images [47]. It also includes learning to distinguish segments within time series or predict future observations in time series [21, 38]. The downside to these approaches is that they do not learn the features specifically for the task of interest. Hence, these methods may be suboptimal. Moreover, many of the aforementioned methods do not generalize to work on all types of data.

The second category of unsupervised methods typically alternately optimizes the parameters of the network and the labels or cluster assignments for the observations. In this thread, several papers alternate between obtaining assignments or soft assignments and optimizing the parameters of a loss function aimed at creating well-separated clusters [43, 16, 45, 20]. In contrast, [7] randomly generates outputs and then alternately optimizes over the parameters of the model and the assignment of labels to outputs. The most straightforward approach is that of [8], which alternately clusters the data to obtain pseudo-labels and takes steps to optimize the multinomial logistic loss on the observations with the given pseudo-labels. The downside to the first group of papers is that they use an ad-hoc loss to encourage clustering. Moreover, none of the above papers use a standard unified loss function to learn both the labels and the parameters of the deep network. In this paper we propose such a loss function and to do so we build on work from the semi-supervised learning literature.

Semi-supervised learning. A broad array of approaches to semi-supervised learning exist [10, 33]. Currently popular methods create a semi-supervised algorithm by modifying a supervised algorithm. These methods include adding a penalty to a supervised loss to encourage similar inputs to be close together in feature space [4, 3], adding a penalty to encourage high-confidence outputs [18], and rounding outputs to obtain pseudo-labels [27]. In addition, [11] constructs a nearest neighbor graph based on labeled and unlabeled data and updates the estimated labels as new observations are added. In contrast to these approaches, there is a set of principled shallow approaches that optimize over both the parameters of the classifier and the labels. This can be performed in a variety of ways, including with constrained "forward prediction" using ridge regression or penalized multinomial logistic regression where the labels are also learned [2, 15, 24] and with constrained "reverse prediction" using a k-means formulation or generalizations thereof [44, 41].

Outline. In this work we build off of [2], stacking their semi-supervised classifier on top of a deep network. Such a formulation allows us to learn feature representations for the inputs regardless of the amount of labeled data. In the case of only unlabeled data it boils down to a clustering problem with some similarities to k-means. Given some labeled data, additional constraints on the labels are included in the objective function. We improve upon the training of the classifier in [2] by proposing an alternative method for handling the constraints on the labels in the unsupervised case. We also show that this "forward prediction" formulation can be better conditioned than the "reverse prediction" formulation with k-means. The latter was proposed in [44] and used for unsupervised learning in [8].

3 Discriminative Clustering with any Level of Supervision

A fundamental problem with most clustering methods is that they require a similarity measure or a metric between inputs [40]. However, it is often not clear how to best measure the distance between inputs, especially in the case of structured inputs such as images. Here we show how we can *learn* such a similarity measure by taking advantage of a limited amount of labeled data and a large amount of unlabeled data.

3.1 Deep discriminative clustering

Consider clustering inputs $x_1, \dots, x_n \in \mathbb{R}^d$, each having a corresponding label $y_i^\star \in \{0,1\}^k$ that may or may not be observed. We denote by $\mathcal{S}$ the set of indices corresponding to the labeled data and by $\mathcal{U}$ the set of indices corresponding to the unlabeled data. We aim to use both the labeled and unlabeled data to learn a feature map $\phi(\cdot; V) : \mathbb{R}^d \to \mathbb{R}^D$ of a kernel parameterized by a set of matrices $V = \{V_1, \dots, V_m\}$ for some m.

For that purpose, we use the square loss $\ell(y, \hat{y}) = (y - \hat{y})^2/2$ to learn the parameters $W \in \mathbb{R}^{D \times k}$ and $b \in \mathbb{R}^k$ of a classifier on the output features $\phi(x_i; V)$. Formally, we aim to optimize the parameters of the feature representation, the parameters of the loss function, and the unknown labels by minimizing the following objective function:

$$\min_{Y \in \mathcal{C}, V, W, b} \quad \frac{1}{n} \sum_{i=1}^{n} \ell\left(y_i, W^T \phi(x_i; V) + b\right) + \lambda \Omega([V; W]) \tag{1}$$

with $\mathcal{C} = \{Y \in \{0,1\}^k : Y \mathbb{1}_k = 1, y_i = y_i^\star \text{ for } i \in \mathcal{S}\}$ and $\Omega([V;W]) = \alpha \|W\|_F^2 + (1-\alpha) \sum_{j=1}^m \|V_j\|_F^2$. Here $Y = [y_1, \dots, y_n]^T$ is the matrix of labels, which is constrained to assign a point to a unique cluster, and $\alpha \in [0,1]$, λ and μ are regularization parameters

The advantage of this objective is that it simultaneously captures three regimes: the unsupervised regime, in which learning entails deep clustering, the supervised regime, in which learning entails supervised training of a deep network, and the semi-supervised regime, in which learning entails a combination of clustering and supervised training. Given an optimization algorithm for this objective, we may therefore proceed with training regardless of the amount of labeled data.

Avoiding trivial solutions. Even when the features are fixed, the above objective may lead to trivial solutions that assign all points to the same cluster, as noted by [2]. To avoid this behavior we add balancing constraints on the sizes of the clusters that enforce that clusters have the same size. During the optimization phase we relax the discrete constraints and obtain a convex problem that can be solved efficiently by alternating minimization.

When learning features by clustering a second issue is their collapsing, i.e., the possibility that all points are mapped via ϕ to the same single point. To avoid this behavior we add a penalty on the feature map of the batch of input points.

Formally, we consider then the problem

$$\min_{Y\in\mathcal{C},V,W,b} \quad \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i, W^T\phi(x_i;V)+b\right) + \lambda\Omega([V;W]) - \mu\|\Pi_n\Phi(X;V)\|_2^2 \tag{2}$$

where $\Phi(X;V) = (\phi(x_1;V), \dots \phi(x_n;V)) \in \mathbb{R}^{n\times D}$ is the design matrix produced by the kernel and $\Pi_n = \mathrm{I}_n - \mathbb{1}_n\mathbb{1}_n^T/n$ is the centering projection matrix.

Comparison with k-means. Since we use the square loss, in the shallow case the objective function (1) boils down to the DIFFRAC objective [2]. In the case of centered data Z, the objective for fixed labels $Y \in \{0,1\}^{n\times k}$ with $Y\mathbb{1}_k = \mathbb{1}_n$ is given by

$$F_d(Z) = \min_W \frac{1}{n}\|Y - ZW\|_F^2 + \lambda\|W\|_F^2 \ .$$

We could alternatively use the k-means objective

$$F_k(Z) = \min_C \frac{1}{n}\|Z - YC\|_F^2$$

for fixed labels $Y \in \{0,1\}^{n\times k}$ with $Y\mathbb{1}_k = \mathbb{1}_n$. As noted by [2], the DIFFRAC objective normalizes the data, whereas the k-means objective normalizes the labels. The following proposition suggests that the DIFFRAC loss is generally smoother than the k-means loss. For a more detailed proof, see Appendix A.

Proposition 1. *The Lipschitz constant of F_k can be estimated by*

$$\|\nabla F_k(Z)\|_2 \le \ell_k := \frac{2}{n}\|Z\|_2$$

while the Lipschitz constant of F_d can be estimated by

$$\|\nabla F_d(Z)\|_2 \le \ell_d := 2\frac{n_{\max}}{\lambda n^2}\|Z\|_2 \ ,$$

where n_{max} is a bound on the maximum number of points in a cluster. Then for any $\lambda \ge n_{\max}/n$, we have $\ell_k \ge \ell_d$.

Proof. First, recall that the k-means objective for fixed cluster assignments Y may be written as $F_k(Z) = \frac{1}{n}\operatorname{tr}[(\mathrm{I} - P_Y)ZZ^T]$ where $P_Y = Y(Y^TY)^{-1}Y^T$ is an orthonormal projector. Its gradient, $\nabla F_k(Z) = \frac{2}{n}(\mathrm{I} - P_Y)Z$, can therefore be bounded as

$$\|\nabla F_k(Z)\|_2 \le \frac{2}{n}\|Z\|_2 =: \ell_k.$$

Next, after minimizing in the classifier variable W, the DIFFRAC objective reads $F_d(Z) = \lambda\operatorname{tr}[YY^T(ZZ^T + n\lambda\,\mathrm{I})^{-1}]$. Define $G(Z) = \left(ZZ^T + n\lambda\,\mathrm{I}_n\right)^{-1}$. Its gradient is then $\nabla F_d(Z) = -2\lambda G(Z)YY^\top G(Z)Z$. Since $\|G(Z)\|_2 \le 1/(n\lambda)$, $\|YY^T\|_2 \le n_{\max}$, and $\|G(Z)Z\|_2 \le \|Z\|_2/(n\lambda)$, we obtain

$$\|\nabla F_d(Z)\|_2 \le 2\frac{n_{\max}}{\lambda n^2}\|Z\|_2 =: \ell_d.$$

Hence, taking $\lambda \ge n_{\max}/n$, we have $\ell_k \ge \ell_d$. □

3.2 Optimization

The optimization of the objective function (2) consists of two main parts: implicitly solving for W, b in terms of V and then optimizing V given fixed Y; and optimizing Y given fixed V, W, b. The first optimization is performed via the ultimate layer reversal stochastic gradient optimization (ULR-SGO) method, which we review next. The second optimization is performed by enforcing balancing constraints. We term the overall algorithm XSDC for “X-Supervised Discriminative Clustering” where “X” can be “un”, “semi” or “-”, hence convering all cases.

Ultimate layer reversal. The ultimate layer reversal method, first introduced in [23], differentiates through the result of the classification performed at the final layer of a network. Consider optimizing (2) over V, W, and b for fixed Y. We may write the corresponding objective function as

$$f_{\text{ulr}}(V, W, b) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i, W^T\phi(x_i; V) + b\right) + \lambda\Omega([V; W]) - \mu\|\Pi_n\Phi(X; V)\|_F^2 \ .$$

Rather than optimizing over V, W, b simultaneously, we can instead solve for W, b in terms of V and then differentiate through the resultant objective. That is, we may write

$$\min_{V,W,b} f_{\text{ulr}}(V, W, b) = \min_{V} \hat{f}_{\text{ulr}}(V) \ ,$$

where $\hat{f}_{\text{ulr}}(V) = \min_{W,b} f_{\text{ulr}}(V, W, b)$ and solve the problem on the right hand side. As long as f_{ulr} is twice differentiable and f_{ulr} viewed as a function of W, b is strongly convex, gradient descent on this objective converges to a stationary point and the resultant $\varepsilon-$ stationary points are $\varepsilon-$ stationary points of the original problem. If $\hat{f}(V)$ is not available in closed form we may estimate it using a quadratic approximation of the loss around the current estimate of V. In addition, if V is constrained this method can still be applied.

Performing the optimization using ULR-SGO has two main benefits. First, it was demonstrated to outperform standard stochastic gradient optimization in [23]. Second, in the case of the square loss it allows us to avoid estimating the assignment matrix Y during the second part of the optimization, instead allowing us to optimize the equivalence matrix $M := YY^T$. To see this, note that with the square loss we have

$$\hat{f}_{\text{ulr}}(V) = \alpha\lambda \operatorname{tr}[M\Pi_n(\Pi_n\Phi(X; V)\Phi(X; V)^T\Pi_n + n\alpha\lambda\,\mathrm{I})^{-1}\Pi_n] + g(X, V)$$

where $g(X, V) = (1-\alpha)\lambda\sum_{j=1}^{m}\|V_j\|_F^2 - \mu\|\Pi_n\Phi(X; V)\|_F^2$. Since we only need to optimize over M we can avoid dealing with the problem of there being many solutions $Y^\star$ caused by the optimal objective value being the same if the columns of Y are permuted. In practice this means that we avoid performing a rounding step to obtain the assignment matrix.

Matrix balancing. Now consider the objective function (1) when fixing V, W, b and optimizing only over the labels for the unlabeled data, $Y_{\mathcal{U}}$. Here we will optimize only over the portion of the objective corresponding to the unlabeled examples. However, it is also possible to optimize over the entire objective. We will also assume here that the classes are balanced; for the more general case see Appendix B.2. Define the equivalence matrix $M_{\mathcal{U}} = Y_{\mathcal{U}}Y_{\mathcal{U}}^T \in \mathbb{R}^{n\times n}$ and the matrix $A = \Pi_n(\Pi_n\Phi(X_{\mathcal{U}}; V)\Phi(X_{\mathcal{U}}; V)^T\Pi_n + n\alpha\lambda\,\mathrm{I})^{-1}\Pi_n$. Relaxing the discrete constraints, requiring a minimum number c of observations in each cluster, and adding an entropic regularization, we obtain the problem

$$\min_{M} \quad \frac{1}{2}\operatorname{tr}(MA) + \mu D_h(M; M_0)$$

Algorithm 1 XSDC

```
1: Input: Labeled data $X_{\mathcal{S}}, Y_{\mathcal{S}}$
2:        Unlabeled data $X_{\mathcal{U}}$
3:        Number of iterations $T$
4: Initialize: $V_1, W_1, b_1 \leftarrow$ Optimize (2) over $V, W, b$ using $X_{\mathcal{S}}, Y_{\mathcal{S}}$
5: for $t = 1, \ldots, T$ do
6:     $M_t \leftarrow \text{MatrixBalancing}(A_t(X_{\mathcal{U}_t}; V_t))$
7:     $V_{t+1} \leftarrow \text{ULR-SGO step}(\Phi(X_{\mathcal{U}_t}; V_t), M_t)$
8: end for
9: $\hat{Y}_{\mathcal{U}} \leftarrow \text{NearestNeighbor}(\Phi(X; V_T), Y_{\mathcal{S}})$
10: $\hat{W}, \hat{b} \leftarrow \text{RegLeastSquares}(X; [Y_{\mathcal{S}}, \hat{Y}_{\mathcal{U}}])$
11: Output: $\hat{Y}, V_T, \hat{W}, \hat{b}$
```

Algorithm 2 Matrix Balancing

```
1: Input: Matrix $A \in \mathbb{R}^{n \times n}$
2:        Entropic regularization parameter $\mu$
3:        Number of clusters $k$
4:        Number of iterations $T_e$
5: Initialize: $M_0 = \mathbb{1}_n \mathbb{1}_n^T / k$
6:             $N = \exp(\mu^{-1} A) \odot M_0$
7:             $u_1 = \mathbb{1}_n$
8: for $t = 1, \ldots, T_e$ do
9:     $(v_t)_i = (n/k)/(N_{\cdot,i}^\top u_t)$    $\forall i = 1, \ldots, n$
10:    $(u_{t+1})_i = (n/k)/(N_{i,\cdot}^T v_t)$    $\forall i = 1, \ldots, n$
11: end for
12: Output: $M = \operatorname{diag}(u_{T_e}) N \operatorname{diag}(v_{T_e})$
```

$$\text{subject to} \quad M\mathbb{1}_n = \frac{n}{k}\mathbb{1}_n, \quad M^\top \mathbb{1}_n = \frac{n}{k}\mathbb{1}_n. \quad M \geq 0, \quad M \succeq 0 \,.$$

Here $D_h(M; M_0) = h(M) - h(M_0) - \langle \nabla h(M_0), M - M_0 \rangle = h(M) - \langle \mathbb{1}_n \mathbb{1}_k^\top + \log(M_0), M \rangle + \mathrm{C}$ where $h(M) = \sum_{ij} M_{ij} \log(M_{ij})$ with $\nabla h(M) = \mathbb{1}_n \mathbb{1}_k^\top + \log(M)$, C is a constant, and M_0 is given by $M_{ij} = 1/k$. Optimizing the dual of this problem via alternating minimization (see Appendix B), we obtain Algorithm 2. In practice we find that 10 steps of the alternating minimization suffices.

Overall algorithm. The overall XSDC algorithm is summarized in Algorithm 1. For conciseness we omit the details regarding the parameters, which are described in the main text. The algorithm proceeds as follows. First, we initialize the filters V randomly and then optimize the objective on the labeled data to estimate V, W, b. Next, we proceed to optimize using the unlabeled data. At each iteration we compute the matrix $A = \Pi_n(\Pi_n \Phi(X_{\mathcal{U}_t}; V)\Phi(X_{\mathcal{U}_t}; V)^T \Pi_n + n\alpha\lambda \mathrm{I})^{-1}\Pi_n$ on a minibatch of inputs $X_{\mathcal{U}_t}$. We then perform matrix balancing to obtain M_t. Fixing M_t, we then take a gradient step through the ULR-SGO objective. At the end we obtain labels for the unlabeled data using 1-nearest neighbor on the feature representations $\Phi(X; V)$. Finally, we estimate the parameters W, b by computing the solution to the least squares problem with X and $[Y_{\mathcal{S}}, \hat{Y}_{\mathcal{U}}]$.

4 Experiments

In the experiments we demonstrate the effectiveness of our proposed methods on two datasets and focus on two main topics: how the performance of our XSDC algorithm varies as the amount of labeled data changes; and whether the performance of our XSDC algorithm improves upon baselines in which the feature representation is not learned.

4.1 Choice of ϕ

One benefit of the XSDC algorithm is that it can learn a similarity measure for clustering. Typical clustering methods either do not transform the features or use a kernel-based method. However, clustering in the original feature space is often ineffective and clustering using the Gram matrix on the inputs is infeasible when there are a large number of points. Consequently, one often uses an approximation of the kernel. Many methods for approximating kernels exist, including random

Table 1: Details regarding the datasets used in the experiments

Dataset	Training size	Validation size	Test size	Dimension	# Classes
Gisette	4,800	1,200	1,000	5,000	2
MAGIC	8,026	2,006	4,755	10	2

Fourier features and the Nyström method [35, 42, 30]. Random Fourier features are data-independent and the parameters of the Nyström method are typically selected at random or via clustering [32].

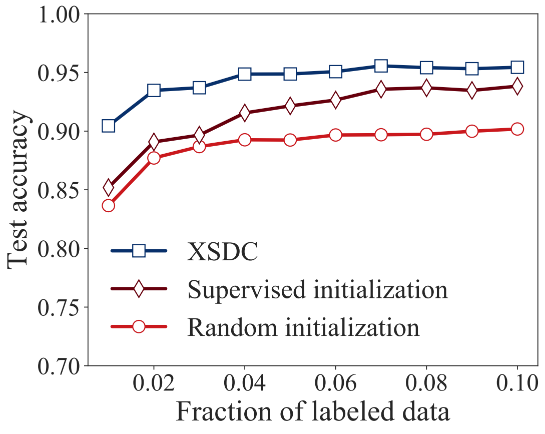
In our experiments we aim to show that learning the feature representations produced via the Nyström method outperforms using the usual unlearned representations. That is, we are able to learn how to represent the data in order to best cluster it. To this end, we consider the Gaussian RBF kernel $k(x, x') = \exp(-1/(2\sigma^2)\|x - x'\|^2)$. The regularized Nyström method approximates the kernel k by computing the inner products of features $\phi(x)$ defined by $\phi(x) = (k(Z^T Z) + \epsilon I)^{-1/2} k(Z^T x)$ for some small $\epsilon > 0$ where the matrix Z contains the parameters. In contrast, random Fourier features used to approximate k are given by $\psi(x) = \sqrt{2/n} \cos(x^T z_1 + z_2)$ where $z_1 \sim N(0, 1/\sigma^2)$ and $z_2 \sim \text{Unif}(0, 2\pi)$.

4.2 Experimental details

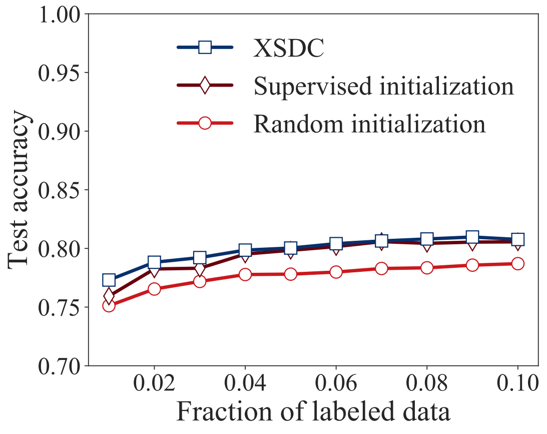
Experimental setup. The experiments focus on two datasets: the vectorial datasets Gisette and MAGIC [19, 6]. Gisette contains features derived from MNIST images [26] and the goal is to distinguish between observations corresponding to the numbers four and nine. MAGIC contains measurements related to simulated particles observed by a gamma telescope. The goal is to distinguish between gamma particles and hadrons. The sizes and dimensions of each dataset may be found in Table 1. For the MAGIC dataset, which does not have a train/test split, we randomly split the data 75%/25% into train/test sets. For each dataset we set aside 20% of the training set to use as a validation set. The datasets are transformed prior to usage as follows. The Gisette dataset is the scaled version found in the LibSVM database [9]. The MAGIC dataset is standardized. As the main version of our algorithm assumes that the classes are balanced, for MAGIC we randomly removed 4,223 observations in the training set with label 1. We leave the evaluation of the performance of the version of our algorithm with constrained assignments to future work.

The architectures we use in the experiments are single-layer kernel networks. We use single-layer networks that approximate an RBF kernel using the Nyström method. For each of these networks we use 32 filters per layer for the hidden layers. The bandwidths for the networks are set to the median pairwise distance between the first 1000 inputs. The features output by the network ϕ are centered and normalized so that on average they have unit ℓ_2 norm, as in [29]. For details on the parameter values and cross-validation see Appendix C. The cross-validation was performed on the datasets for the case of 1% labeled data and the best parameters found were used for the other cases.

Training. The training is performed as follows. The initialization of the network parameters is performed by randomly sampling from the feature representations at each layer of the network. Then the network is trained for 100 iterations using labeled data. Finally, the network is trained using the ULR-SGO algorithm and matrix balancing on the unlabeled data for 400 iterations. We evaluate the performance of the learned representations every 10 iterations.



(a) Gisette



(b) MAGIC

Figure 1: Average performance across 10 trials of XSDC when varying the fraction of labeled data.

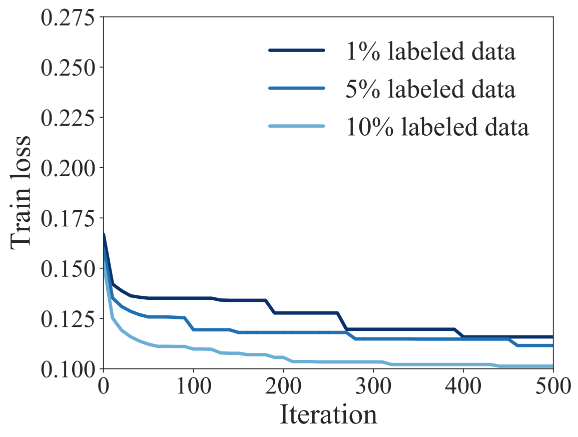
Code. The code for this project builds upon the `YesWeCKN` package from [23], which is written using PyTorch and Faiss [34, 22].[1] It will be made available upon publication. The experiments were run using GeForce GTX 1080Ti, Titan Xp and Titan V GPUs. The corresponding time to run all the experiments sequentially on a single NVIDIA Titan Xp GPU would be more than 3 days.

4.3 Results

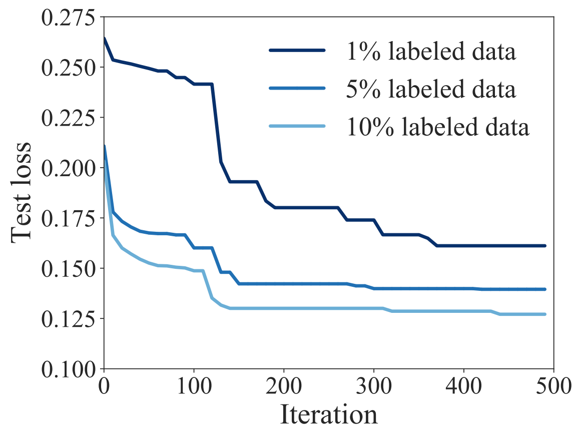
Comparison with baselines. In our experiments we compare the XSDC algorithm to two baselines: an initial supervised training of the classifier when the network has random weights ("random initialization") and an initial supervised training of both the network and the classifier ("supervised initialization"). In the latter case the network is trained on only the labeled data. In both cases, when evaluating the performance the labels of the unlabeled data are first estimated using 1-nearest neighbor with the labeled data. The classifier is then trained on the labeled and unlabeled data. The reported accuracy of the supervised initialization is the test accuracy after 100 iterations. In contrast, the reported accuracy of XSDC is the test accuracy observed at the iteration where the validation accuracy is highest. We report this value because the algorithm can overfit before 500 iterations. We performed 10 trials when varying the random seed and report the corresponding results.

Benefits of unlabeled data. We would expect that XSDC would provide an improvement over the supervised initialization when there are gains to be had from additional labeled data. Otherwise, we would expect training on additional unlabeled data to provide little to no benefit. This is exactly what we see in Figure 1. Figure 1 compares the accuracy of the XSDC algorithm to the initializations as the fraction of labeled data varies. From both plots we can see that the performance of XSDC relative to the supervised baseline is much larger when the fraction of labeled data is smaller. With 1% labeled data the accuracy on Gisette increases by 5% on average when using XSDC instead of the supervised baseline. On MAGIC the gain is more modest, at 1%. In contrast, for 10% labeled data XSDC outperforms the supervised baseline by 2% on Gisette but is only 0.2% better than

[1] `https://github.com/cjones6/yesweckn`



(a) Training loss on Gisette



(b) Test loss on Gisette

Figure 2: Loss of XSDC on the training and test sets of Gisette for the first trial. The reported loss is the smallest loss observed thus far throughout the iterations.

the baseline on MAGIC. The latter results make sense since the increase in performance of the supervised initialization with the fraction of labeled data has started leveling off by then. It is also important to recall that the cross-validation was performed for the case of 1% labeled data, with the resultant parameter values found being applied to the other cases. By instead tuning the parameters for each fraction of labeled data we could potentially improve the results.

Figure 2 plots the minimum cumulative training and test losses vs. iteration for the first trial on the Gisette dataset. Here the training loss is computed using the known labels for the labeled portion of the data and the estimated labels for the unlabeled portion. From the plot on the left we can see that the training loss decreases across the iterations, including after the unsupervised portion of the learning begins at iteration 100. Hence, XSDC is indeed decreasing the overall training loss even though it is only using the unlabeled data for much of the time. From the plot on the right, we can see that decreases in the training loss translate into decreases in the test loss.

Additional results. Appendix D contains additional plots summarizing the results. First, Figure 3 is a version of Figure 1 that includes error bars denoting one standard error from the mean. It also compares the performance of XSDC and its initializations to a baseline with random Fourier features (RFFs). The standard error of the difference in the performance tends to be larger when the gap in the performance between XSDC and the supervised baseline is larger, as expected. For example, on Gisette the standard error of the difference in the performance of XSDC and the supervised baseline is 3% in the case of 1% labeled data, but only 1% in the case of 10% labeled data. Regarding the bottom curve in the plot, the corresponding RFF network had 32 hidden nodes, just like the network used for XSDC. However, the accuracy of the RFF initialization on both Gisette and MAGIC is never above 71% on average. On Gisette it hovers between 50 and 60%, more than 20% below our initializations. This is consistent with previous evidence suggesting that when using random Fourier features a much larger number of hidden nodes is required in order to well-approximate the kernel [46]. We therefore conclude that learning a data-dependent feature representation greatly improves the performance of the algorithm over data-independent feature representations such as ones generated using random Fourier features and the Nyström method with random weights. Finally, Figure 4 plots the training and test accuracy across iterations for the first trial. We can see that the training proceeds fairly quickly, with the performance on Gisette for 1% labeled data increasing by more than 10% within the first 100 iterations after the initialization ends.

5 Conclusion

In this work we presented a unified learning algorithm called XSDC that proceeds regardless of the amount of labeled data. In the unsupervised case it boils down to clustering features from a deep network trained on unlabeled data, while in the supervised case it reduces to the traditional training of a deep network. The main hurdle lay in handling the constraints on the labels for the unlabeled data. For this we derived an effective sub-algorithm that approximates solutions based on a continuous relaxation of the problem. We demonstrated the effectiveness of XSDC on two datasets, showing that when adding additional labeled data would help, substituting it with unlabeled data can still yield large performance improvements.

Acknowledgements

This work was supported by NSF CCF-1740551, NSF DMS-1810975, the program "Learning in Machines and Brains" of CIFAR, and faculty research awards.

References

[1] D. Ardila, A. P. Kiraly, S. Bharadwaj, B. Choi, J. J. Reicher, L. Peng, D. Tse, M. Etemadi, W. Ye, G. Corrado, D. P. Naidich, and S. Shetty. End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography. *Nature Medicine*, 2019.

[2] F. R. Bach and Z. Harchaoui. DIFFRAC: a discriminative and flexible framework for clustering. In *Neural Information Processing Systems*, pages 49–56, 2007.

[3] P. Bachman, O. Alsharif, and D. Precup. Learning with pseudo-ensembles. In *Advances in Neural Information Processing Systems*, pages 3365–3373, 2014.

[4] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7: 2399–2434, 2006.

[5] L. Bo, K. Lai, X. Ren, and D. Fox. Object recognition with hierarchical kernel descriptors. In *Conference on Computer Vision and Pattern Recognition*, pages 1729–1736, 2011.

[6] R. Bock, A. Chilingarian, M. Gaug, F. Hakl, T. Hengstebeck, M. Jirina, J. Klaschka, E. Kotrc, P. Savicky, S. Towers, A. Vaicilius, and W. Wittek. Methods for multidimensional event classification: A case study using images from a Cherenkov gamma-ray telescope. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 516(2):511–528, 2004.

[7] P. Bojanowski and A. Joulin. Unsupervised learning by predicting noise. In *International Conference on Machine Learning*, pages 517–526, 2017.

[8] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision*, pages 139–156, 2018.

[9] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[10] O. Chapelle, B. Schlkopf, and A. Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010.

[11] I. Chiotellis, F. Zimmermann, D. Cremers, and R. Triebel. Incremental semi-supervised learning from streams for object classification. In *International Conference on Intelligent Robots and Systems*, pages 5743–5749, 2018.

[12] A. Daniely, R. Frostig, V. Gupta, and Y. Singer. Random features for compositional kernels. *CoRR*, abs/1703.07872, 2017.

[13] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *International Conference on Computer Vision*, pages 1422–1430, 2015.

[14] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. A. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(9):1734–1747, 2016.

[15] N. Flammarion, B. Palaniappan, and F. Bach. Robust discriminative clustering with sparse regularizers. *Journal of Machine Learning Research*, 18:80:1–80:50, 2017.

[16] K. Ghasedi Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *International Conference on Computer Vision*, pages 5747–5756, 2017.

[17] I. J. Goodfellow, Y. Bengio, and A. C. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016.

[18] Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems*, pages 529–536, 2004.

[19] I. Guyon, S. R. Gunn, A. Ben-Hur, and G. Dror. Result analysis of the NIPS 2003 feature selection challenge. In *Advances in Neural Information Processing Systems*, pages 545–552, 2004.

[20] P. Häusser, A. Mordvintsev, and D. Cremers. Learning by association - A versatile semi-supervised training method for neural networks. In *Conference on Computer Vision and Pattern Recognition*, pages 626–635, 2017.

[21] A. Hyvärinen and H. Morioka. Unsupervised feature extraction by time-contrastive learning and nonlinear ICA. In *Advances in Neural Information Processing Systems*, pages 3765–3773, 2016.

[22] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with GPUs. *CoRR*, abs/1702.08734, 2017.

[23] C. Jones, V. Roulet, and Z. Harchaoui. Kernel-based translations of convolutional networks. *arXiv preprint arXiv:1903.08131*, 2019.

[24] A. Joulin and F. R. Bach. A convex relaxation for weakly supervised classifiers. In *International Conference on Machine Learning*, 2012.

[25] Y. LeCun. *Modeles connexionnistes de l'apprentissage.* PhD thesis, Université P. et M. Curie (Paris 6), June 1987.

[26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press, 2001.

[27] D.-H. Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *International Conference on Machine Learning Workshop on Challenges in Representation Learning*, 2013.

[28] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox. DeepIM: Deep iterative matching for 6d pose estimation. In *European Conference on Computer Vision*, pages 695–711, 2018.

[29] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional kernel networks. In *Advances in Neural Information Processing Systems*, pages 2627–2635, 2014.

[30] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning.* Adaptive computation and machine learning. MIT Press, 2012.

[31] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84, 2016.

[32] D. Oglic and T. Gärtner. Nyström method with kernel k-means++ samples as landmarks. In *International Conference on Machine Learning*, pages 2652–2660, 2017.

[33] A. Oliver, A. Odena, C. A. Raffel, E. D. Cubuk, and I. J. Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems*, pages 3239–3250, 2018.

[34] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *Advances in Neural Information Processing Systems Workshop on Autodiff*, 2017.

[35] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pages 1177–1184, 2007.

[36] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, and S. Levine. Time-contrastive networks: Self-supervised learning from video. In *International Conference on Robotics and Automation*, pages 1134–1141, 2018.

[37] J. Thickstun, Z. Harchaoui, D. P. Foster, and S. M. Kakade. Invariances and data augmentation for supervised music transcription. In *International Conference on Acoustics, Speech and Signal Processing*, pages 2241–2245, 2018.

[38] A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018.

[39] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *International Conference on Computer Vision*, pages 2794–2802, 2015.

[40] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, 2009.

[41] M. White and D. Schuurmans. Generalized optimal reverse prediction. In *International Conference on Artificial Intelligence and Statistics*, pages 1305–1313, 2012.

[42] C. K. I. Williams and M. W. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, pages 682–688, 2000.

[43] J. Xie, R. B. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *International Conference on Machine Learning*, pages 478–487, 2016.

[44] L. Xu, M. White, and D. Schuurmans. Optimal reverse prediction: a unified perspective on supervised, unsupervised and semi-supervised learning. In *International Conference on Machine Learning*, pages 1137–1144, 2009.

[45] J. Yang, D. Parikh, and D. Batra. Joint unsupervised learning of deep representations and image clusters. In *Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2016.

[46] T. Yang, Y. Li, M. Mahdavi, R. Jin, and Z. Zhou. Nyström method vs random fourier features: A theoretical and empirical comparison. In *Advances in Neural Information Processing Systems*, pages 485–493, 2012.

[47] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666, 2016.

Appendix

In this appendix we provide additional details related to the conditioning of the objective function, the optimization of the cluster assignments, and the experiments.

A Conditioning comparison of k-means and DIFFRAC

In this appendix we estimate Lipschitz constants of the unsupervised objective with k-means and the unsupervised objective with DIFFRAC. This indicates that, generally, as the DIFFRAC objective uses a whitened design matrix to compute the clustering step, it provides a smoother objective.

Proposition 2. *Consider a centered design matrix Z. Denote the k-means objective by*

$$F_k(Z) = \min_C \frac{1}{n} \|Z - YC\|_F^2$$

and the Diffrac objective by

$$F_d(Z) = \min_W \frac{1}{n} \|Y - ZW\|_F^2 + \lambda \|W\|_F^2,$$

for fixed labels $Y \in \{0,1\}^{n \times k}$ with $Y\mathbb{1}_k = \mathbb{1}_n$. The Lipschitz constant of F_k can be estimated by

$$\|\nabla F_k(Z)\|_2 \leq \ell_k := \frac{2}{n} \|Z\|_2$$

while the Lipschitz constant of F_d can be estimated by

$$\|\nabla F_d(Z)\|_2 \leq \ell_d := 2\frac{n_{\max}}{\lambda n^2} \|Z\|_2 \ ,$$

where n_{max} is a bound on the maximum number of points in a cluster. Then for any $\lambda \geq n_{\max}/n$, we have $\ell_k \geq \ell_d$.

Proof. First, recall that the k-means objective for fixed cluster assignments Y may be written as

$$\begin{aligned} F_k(Z) &:= \min_C \frac{1}{n}\|Z - YC\|_F^2 \\ &= \frac{1}{n}\|Z - Y(Y^TY)^{-1}Y^TZ\|_F^2 \\ &= \frac{1}{n}\operatorname{tr}[(\mathrm{I} - P_Y)ZZ^T] , \end{aligned}$$

where $P_Y = Y(Y^TY)^{-1}Y^T$ is an orthonormal projector. Its gradient, $\nabla F_k(Z) = \frac{2}{n}(\mathrm{I} - P_Y)Z$, can then be bounded as

$$\|\nabla F_k(Z)\|_2 = \frac{2}{n}\|(\mathrm{I} - P_Y)Z\|_2 \leq \frac{2}{n}\|Z\|_2 =: \ell_k.$$

Next, after minimizing in the classifier variable W, the DIFFRAC objective reads

$$F_d(Z) = \lambda \operatorname{tr}[YY^T(ZZ^T + n\lambda\,\mathrm{I})^{-1}] .$$

Define $G(Z) = \left(ZZ^T + n\lambda\,\mathrm{I}_n\right)^{-1}$. Its gradient is then

$$\nabla F_d(Z) = -2\lambda G(Z)YY^\top G(Z)Z .$$

We may bound the norm of the gradient as follows:

$$\begin{aligned} \|\nabla F_d(Z)\|_2 &\leq 2\lambda\|G(Z)YY^TG(Z)Z\|_2 \\ &\leq 2\lambda\|G(Z)\|_2\|YY^T\|_2\|G(Z)Z\|_2 \end{aligned}$$

Now observe that $\lambda_{\max}(YY^T) = \lambda_{\max}(Y^TY) \leq n_{\max}$, where $n_{\max}$ is a bound on the maximum number of points in a cluster. Moreover, letting $Z = USV^T$ be the singular value decomposition of the matrix Z, with $S = \operatorname{diag}(s_1, \ldots, s_n)$ where $s_1, \ldots, s_n$ are the singular values of S, we have

$$\begin{aligned} \|G(Z)\|_2 &= \left\|\left(ZZ^T + n\lambda\,\mathrm{I}_n\right)^{-1}\right\|_2 \\ &= \left\|\left(US^2U^T + n\lambda\,\mathrm{I}_n\right)^{-1}\right\|_2 \\ &= \left\|U\left(S^2 + n\lambda\,\mathrm{I}_n\right)^{-1}U^T\right\|_2 \\ &\leq \frac{1}{s_{\min}^2 + n\lambda} \\ &\leq \frac{1}{n\lambda} . \end{aligned}$$

Finally, note that

$$\begin{aligned} \|G(Z)Z\|_2 &= \left\|\left(ZZ^T + n\lambda\,\mathrm{I}_n\right)^{-1}Z\right\|_2 \\ &= \left\|U\left(S^2 + n\lambda\,\mathrm{I}_n\right)^{-1}SV^T\right\|_2 \\ &\leq \max_i \frac{s_i}{s_i^2 + n\lambda} \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{n\lambda} \max_i \frac{s_i}{1 + (n\lambda)^{-1} s_i} \\
&\leq \frac{\|Z\|_2}{n\lambda}
\end{aligned}
$$

Therefore we have

$$
\|\nabla F_d(Z)\|_2 \leq 2 \frac{n_{\max}}{\lambda n^2} \|Z\|_2 =: \ell_d
$$

Hence, taking $\lambda \geq n_{\max}/n$, we have $\ell_k \geq \ell_d$. □

B Solving the assignment problem

In the appendix we address the problem of optimizing the clustering assignments. We consider the problem for two relaxations of the constraints: one that assumes the clusters are balanced and one that assumes a minimum number of points per cluster.

B.1 Balanced assignments

The balanced assignment problem for a similarity matrix A reads

$$
\begin{aligned}
\min_{M \in \mathbb{R}^{n \times n}} \quad & \mathbf{Tr}(M^\top A) \\
\text{subject to} \quad & M\mathbb{1} = \frac{n}{k}\mathbb{1}, \quad M^\top \mathbb{1} = \frac{n}{k}\mathbb{1}, \quad M \geq 0
\end{aligned}
$$

where $\mathbb{1} = \mathbb{1}_n$ is the vector of ones in $\mathbb{R}^n$, k is the number of clusters and we ignored the positive semi-definite constraint. We consider an entropic regularization of the problem. Specifically we use the entropic regularizer

$$
h(M) = \sum_{ij} M_{ij} \log(M_{ij}) \quad \text{with} \quad \nabla h(M) = \mathbb{1}_n \mathbb{1}_n^T + \log(M),
$$

and make a proximal step from an initial guess M_0 by considering the Bregman divergence $D_h(M; M_0) = h(M) - h(M_0) - \langle \nabla h(M_0), M - M_0 \rangle = h(M) - \langle \mathbb{1}_n \mathbb{1}_n^T + \log(M_0), M \rangle + \mathrm{C}$, where C is a constant, and a feasible M_0 is given by $M_{0,ij} = 1/k$. Consider then the problem

$$
\begin{aligned}
\min_M \quad & \frac{1}{2} \operatorname{tr}(M^T A) + \mu D_h(M; M_0) \\
\text{subject to} \quad & M\mathbb{1}_n = \frac{n}{k}\mathbb{1}_n, \quad M^T \mathbb{1}_n = \frac{n}{k}\mathbb{1}_n, \quad M \geq 0,
\end{aligned}
$$

which is equivalent to

$$
\begin{aligned}
\min_M \quad & \frac{1}{2} \operatorname{tr}(M^T Q) + \mu h(M) \\
\text{subject to} \quad & M\mathbb{1}_n = \frac{n}{k}\mathbb{1}_n, \quad M^T \mathbb{1}_n = \frac{n}{k}\mathbb{1}_n, \quad M \geq 0 \; ,
\end{aligned}
$$

where $Q = A - \mu \mathbb{1}_n \mathbb{1}_n^T - \mu \log(M_0)$.

Dual minimization. Considering the problem scaled by μ^{-1} and introducing Lagrange multipliers, we obtain

$$\begin{array}{rl} \max\limits_{\alpha\in\mathbb{R}^n,\beta\in\mathbb{R}^n} \min\limits_{M\in\mathbb{R}^{n\times n}} & \mu^{-1}\,\mathbf{Tr}(M^TQ) + h(M) + \mathbf{Tr}(M(\alpha\mathbb{1}_n^T + \mathbb{1}_n\beta^T)) - n/k\alpha^T\mathbb{1}_n - n/k\beta^T\mathbb{1}_n \\ \text{subject to} & M \geq 0 \end{array}$$

The minimum in M for α, β fixed can be computed analytically. It is given by

$$\begin{aligned} M^\star(\alpha,\beta) &= \exp(-(\mu^{-1}Q + \mathbb{1}_n\mathbb{1}_n^T + \alpha\mathbb{1}_n^T + \mathbb{1}_n\beta^T)) \geq 0 \\ &= [\exp(-\alpha)\mathbb{1}_n^T] \odot \exp(-\tilde{Q}) \odot [\mathbb{1}_n \exp(-\beta)^T] \\ &= \operatorname{diag}[\exp(-\alpha)]\exp(-\tilde{Q})\operatorname{diag}[\exp(-\beta)] \end{aligned}$$

where $\exp(X)$ or $\exp(x)$ is to be understood element-wise, $\odot$ denotes the Hadamard (element-wise) product and

$$\tilde{Q} = \mu^{-1}Q + \mathbb{1}_n\mathbb{1}_n^T = \mu^{-1}A - \log(M_0)\ .$$

The dual problem then reads

$$\min_{\alpha\in\mathbb{R}^n,\beta\in\mathbb{R}^n} \quad h^*(-(\mu^{-1}Q + \alpha\mathbb{1}_n^\top + \mathbb{1}_n\beta^\top)) + n/k\alpha^\top\mathbb{1}_n + n/k\beta^\top\mathbb{1}_n$$

where $h^*(X) = \sum_{ij}\exp(X_{ij} - 1) = \mathbf{Tr}(\mathbb{1}_n^\top \exp(X - \mathbb{1}_n\mathbb{1}_n^\top)\mathbb{1}_n)$, such that

$$\begin{aligned} h^*(-(\mu^{-1}Q + \alpha\mathbb{1}_n^\top + \mathbb{1}_n\beta^\top)) &= \mathbf{Tr}(\mathbb{1}_n^\top \operatorname{diag}[\exp(-\alpha)]\exp(-\tilde{Q})\operatorname{diag}[\exp(-\beta)]\mathbb{1}_n) \\ &= \exp(-\alpha)^\top\exp(-\tilde{Q})\exp(-\beta) \end{aligned}$$

We can then rewrite the dual problem as

$$\min_{\alpha\in\mathbb{R}^n,\beta\in\mathbb{R}^n} \quad \exp(-\alpha)^\top N \exp(-\beta) + n/k\alpha^\top\mathbb{1}_n + n/k\beta^\top\mathbb{1}_n$$

where

$$N = \exp(-\tilde{Q}) = \exp(-\mu^{-1}A) \odot M_0 > 0.$$

Alternating minimization. Minimization in α for β fixed reads in variables $u = \exp(-\alpha)$, $v = \exp(-\beta)$,

$$u_i^\star(v) = (n/k)/(N_{i,\cdot}^T v) \ \text{ for all } i = 1,\ldots,n.$$

Minimization in β for α fixed reads in variables $u = \exp(-\alpha)$, $v = \exp(-\beta)$,

$$v_i^\star(u) = (n/k)/(N_{\cdot,i}^T u) \ \text{ for all } i = 1,\ldots,n.$$

Primal mapping. For given $u^\star = \exp(-\alpha^\star), v^\star = \exp(-\beta^\star)$, the primal solution is given by

$$M^\star = \operatorname{diag}[u^\star]N\operatorname{diag}[v^\star]\ .$$

B.2 Constrained assignments

The problem with minimum and maximum size assignments reads

$$\begin{aligned} \min_{M \in \mathbb{R}^{n\times n}} \quad & \mathbf{Tr}(M^\top A) \\ \text{subject to} \quad & n_{\min}\mathbb{1} \le M\mathbb{1} \le n_{\max}\mathbb{1}, \quad n_{\min}\mathbb{1} \le M^\top\mathbb{1} \le n_{\max}\mathbb{1}, \quad M \ge 0 \end{aligned}$$

The constants $n_{\min} \le n_{\max}$ are lower and upper bounds on the sizes of the clusters. The proximal step reads

$$\begin{aligned} \min_{M \in \mathbb{R}^{n\times n}} \quad & \mathbf{Tr}(M^\top A) + \mu D_h(M; M_0) \\ \text{subject to} \quad & n_{\min}\mathbb{1} \le M\mathbb{1} \le n_{\max}\mathbb{1}, \quad n_{\min}\mathbb{1} \le M^\top\mathbb{1} \le n_{\max}\mathbb{1}, \quad M \ge 0 \end{aligned} \tag{3}$$

Dual problem. This problem is equivalent to

$$\begin{aligned} \max_{\alpha,\beta,\gamma,\delta} \min_{M} \quad & \mu^{-1}\,\mathbf{Tr}(M^\top Q) + h(M) + \mathbf{Tr}(M^\top((\beta-\alpha)\mathbb{1}^\top + \mathbb{1}(\delta-\gamma)^\top) \\ & + n_{\min}(\alpha+\gamma)^\top\mathbb{1} - n_{\max}(\beta+\delta)^\top\mathbb{1} \\ \text{subject to} \quad & M \ge 0, \quad \alpha \ge 0, \quad \beta \ge 0, \quad \gamma \ge 0, \quad \delta \ge 0, \end{aligned}$$

where $Q = A - \mu\mathbb{1}\mathbb{1}^\top - \mu\log(M_0)$. The minimum in M for $\alpha, \beta, \gamma, \delta$ fixed can be computed analytically as before. It reads

$$M^\star(\alpha,\beta,\gamma,\delta) = \operatorname{diag}[\exp(-(\beta-\alpha))]\exp(-\tilde{Q})\operatorname{diag}[\exp(-(\delta-\gamma))],$$

where $\tilde{Q} = \mu^{-1}Q + \mathbb{1}\mathbb{1}^\top = \mu^{-1}A - \log(M_0)$. The dual problem then reads

$$\begin{aligned} \min_{\alpha,\beta,\gamma,\delta} \quad & \exp(-(\beta-\alpha))^\top N \exp(-(\delta-\gamma)) - n_{\min}(\alpha+\gamma)^\top\mathbb{1} + n_{\max}(\beta+\delta)^\top\mathbb{1} \\ \text{subject to} \quad & \alpha \ge 0, \quad \beta \ge 0, \quad \gamma \ge 0, \quad \delta \ge 0, \end{aligned}$$

where $N = \exp(-\tilde{Q}) = \exp(-\mu^{-1}A) \odot M_0$. Using the change of variables $a = \beta - \alpha$, $b = \beta + \alpha$, $c = \delta - \gamma$, $d = \delta + \gamma$, the dual problem is then

$$\begin{aligned} \min_{a,b,c,d} \quad & \exp(-a)^\top N \exp(-c) + \frac{n_{\max} - n_{\min}}{2}(b+d)^\top\mathbb{1} + \frac{n_{\max} + n_{\min}}{2}(a+c)^\top\mathbb{1} \\ \text{subject to} \quad & b \ge |a|, \quad d \ge |c|. \end{aligned}$$

Minimization in b and d can be performed analytically (using $n_{\max} \ge n_{\min}$) such that the problem reads

$$\min_{a,c} \quad \exp(-a)^\top N \exp(-c) + n_\delta(\|a\|_1 + \|c\|_1) + n_\Sigma(a+c)^\top\mathbb{1}, \tag{4}$$

where $n_\delta = (n_{\max} - n_{\min})/2$ and $n_\Sigma = (n_{\max} + n_{\min})/2$.

Alternating minimization. We consider alternating minimization schemes to solve (4). The complete algorithm is described in Algorithm 3. It is written in the variables $u = \exp(-a)$ and $v = \exp(-c)$.

Algorithm 3 Constrained balancing to solve (3)

```
**Inputs:** Matrix $A$
            Entropic regularizer $\mu$
            Minimum size $n_{\min}$
            Maximum size $n_{\max}$
            Initial feasible guess $M_0$
            Number of iterations $K$
**Initialize:** $N = \exp(-\mu^{-1}A) \odot M_0$
                $n_\delta = (n_{\max} - n_{\min})/2$
                $n_\Sigma = (n_{\max} + n_{\min})/2$
                $u_0 = \mathbb{1}$
**for** $k = 0, \dots, K$ **do**
    Compute $v_k = (n_\Sigma \mathbb{1} + n_\delta \operatorname{P}_{\mathcal{B}_\infty(n_\Sigma \mathbb{1}, n_\delta)}(N^\top u_k))/(N^\top u_k)$
    Compute $u_{k+1} = (n_\Sigma \mathbb{1} + n_\delta \operatorname{P}_{\mathcal{B}_\infty(n_\Sigma \mathbb{1}, n_\delta)}(N v_k)/(N v_k)$
**end for**
**Output:** $M = \operatorname{diag}(u_K) N \operatorname{diag}(v_K)$
```

Minimization in a. Minimization in a for c fixed reads, denoting $x = N\exp(-c)$,

$$\max_{\|\eta\|_\infty \leq 1} \min_a \exp(-a)^\top x + a^\top (n_\Sigma \mathbb{1} + n_\delta \eta).$$

Switching max and min, the minimum in a for η fixed (which is always defined since $n_\Sigma \mathbb{1} + n_\delta \eta \geq 0$ for $\|\eta\|_\infty \leq 1$) is given by

$$\exp(-a^\star(\eta)) = (n_\Sigma \mathbb{1} + n_\delta \eta)/x.$$

Denoting $\theta = n_\Sigma \mathbb{1} + n_\delta \eta$, the problem then reads

$$\max_{\|\theta - n_\Sigma \mathbb{1}\|_\infty \leq n_\delta} \theta^\top (\mathbb{1} + \log(x)) - \theta \log(\theta).$$

Its minimum is reached at

$$\theta^\star = \operatorname{P}_{\mathcal{B}_\infty(n_\Sigma \mathbb{1}, n_\delta)}(x),$$

where $\operatorname{P}_{\mathcal{B}_\infty(n_\Sigma \mathbb{1}, n_\delta)}$ is the projection on the infinity ball of radius n_δ centered at $n_\Sigma \mathbb{1}$. We get then

$$\exp(-a^\star(c)) = (n_\Sigma \mathbb{1} + n_\delta \operatorname{P}_{\mathcal{B}_\infty(n_\Sigma \mathbb{1}, n_\delta)}(N\exp(-c)))/(N\exp(-c)).$$

Minimization in c. Minimization in c is analogous and we get

$$\exp(-c^\star(a)) = (n_\Sigma \mathbb{1} + n_\delta \operatorname{P}_{\mathcal{B}_\infty(n_\Sigma \mathbb{1}, n_\delta)}(N^\top \exp(-a)))/(N^\top \exp(-a)).$$

Primal mapping. For given $u^\star = \exp(-a^\star), v^\star = \exp(-c^\star)$, the primal solution is given by

$$M^\star = \operatorname{diag}[u^\star] N \operatorname{diag}[v^\star] \ .$$

C Additional training details

The algorithm proposed in this paper and the models used require a large number of parameters to be set. Here we discuss the choices for these parameters.

Table 2: Optimal parameters found during cross-validation (on the log base 2 scale)

Dataset	$\eta_{\mathcal{S}}$	$\eta_{\mathcal{U}}$	$(1-\alpha)\lambda$	μ
Gisette	-4	-3	-4	-2
MAGIC	-2	-7	-4	-2

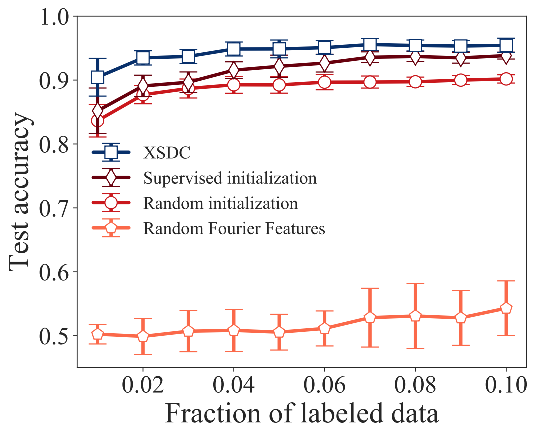
Fixed parameters. The parameters that were fixed throughout the experiments and not cross-validated were as follows. The number of hidden nodes in the networks was set to 32 and the network's parameters V were initialized layer-wise with 32 observations drawn uniformly at random. The networks use the Nyström method to approximate the kernel at each layer. The regularization in the Nyström approximation is set to 0.001 and 20 Newton iterations are used to compute the inverse square root of the Gram matrix on the parameters V_ℓ at each layer ℓ, as in [23]. The bandwidth was set to using the median pairwise distance between the first 1000 observations. The batch size was set to 4096. The initial training phase on just the labeled data is performed for 100 iterations, as the validation loss has typically started leveling off by 100 iterations. The entropic regularization parameter in the matrix balancing was set to the median absolute value of the entries in A. The number of iterations of alternating minimization in the matrix balancing algorithm was set to 10. The number of nearest neighbors used for estimating the labels on the unlabeled data was set to 1.

Cross-validation. The cross-validation was performed on the datasets for the case of 1% labeled data and the best parameters found were used for the other cases. The best parameter values as measured by the accuracy on the validation set are reported in Table 2. Due to the large number of parameters, the parameters were tuned sequentially as follows.

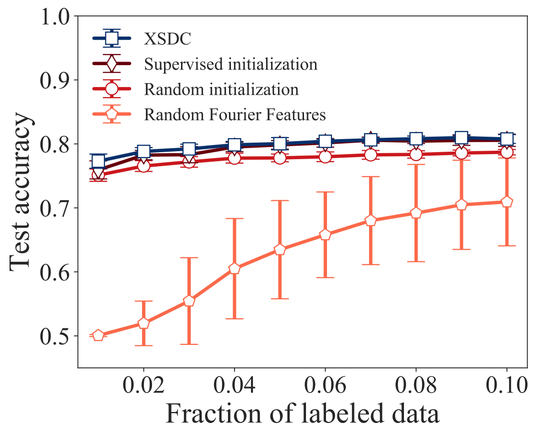
First, we tune the penalty on the classifier weights. To do so we train the classifier on only the labeled data using the initial random network parameters. We then fix the optimal value found across the iterations. Next, we tune the learning rate for the labeled data. For the modest values of $(1-\alpha)\lambda = \mu = 2^{-4}$ we cross-validate the fixed learning rate for the labeled data over the values 2^i for $i = -10, -9, \dots, 4$. To evaluate the performance the labels for the unlabeled data are estimated using 1-nearest neighbor. The labeled and unlabeled data are then used to train the classifier used to compute the performance. We then tune the unsupervised learning rate, again over the values 2^i for $i = -10, -9, \dots, 4$. Following this, we tune μ and then $(1-\alpha)\lambda$ over the values 2^i for $i = -10, -9, \dots, 10$.

D Additional experimental results

Figures 3 and 4 display additional results from the experiments on Gisette and MAGIC.

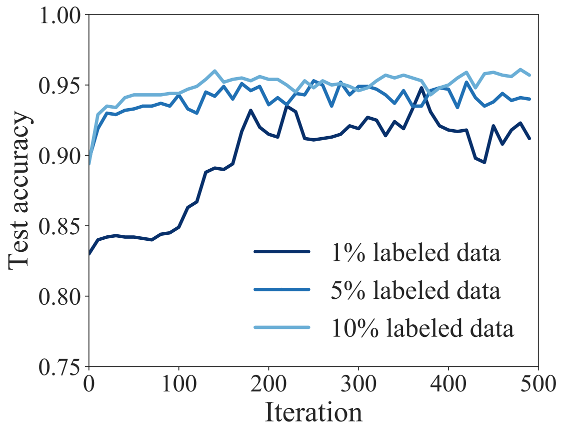


(a) Single-layer kernel network on Gisette

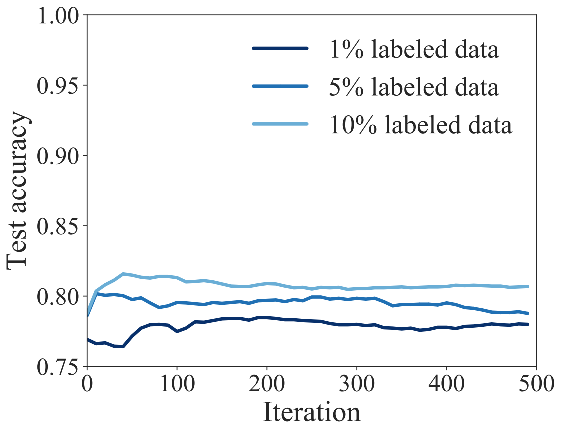


(b) Single-layer kernel network on MAGIC

Figure 3: Average performance across 10 trials of XSDC when varying the fraction of labeled data. The error bars show one standard deviation from the mean.



(a) Single-layer kernel network on Gisette



(b) Single-layer kernel network on MAGIC

Figure 4: Performance of XSDC across iterations.