2024 IEEE 34th International Workshop on Machine Learning for Signal Processing (MLSP) | 979-8-3503-7225-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/MLSP58920.2024.10734736

SELF SUPERVISED DICTIONARY LEARNING USING KERNEL MATCHING

Shubham Choudhary[1,2] *Paul Masset*[3] *Demba Ba*[1,2]

[1] School of Engineering and Applied Science, Harvard University
[2] Kempner Institute for the study of Natural and Artificial Intelligence, Harvard University
[3] Department of Psychology, McGill University

ABSTRACT

We introduce a self supervised framework for learning representations in the context of dictionary learning. We cast the problem as a kernel matching task between the input and the representation space, with constraints on the latent kernel. By adjusting these constraints, we demonstrate how the framework can adapt to different learning objectives. We then formulate a novel Alternate Direction Method of Multipliers (ADMM) based algorithm to solve the optimization problem and connect the dynamics to classical alternate minimization techniques. This approach offers a unique way of learning representations with kernel constraints, that enable us implicitly learn a generative map for the data from the learned representations which can have broad applications in representation learning tasks both in machine learning and neuroscience.

Index Terms— Dictionary Learning, Kernel Matching, Self-Supervised, Sparse coding, Manifold learning

1. INTRODUCTION

Self supervised approaches have recently been a popular framework in unsupervised learning to learn good and robust representations from given data samples and have found widespread applications developing models for artificial and biological intelligence. In the former case, these approaches have made deep strides in enhancing the performance for computer vision (CV) and natural language processing (NLP) [1, 2, 3, 4] tasks. While in neuroscience, such unsupervised approaches for representation learning have led to biologically plausible architectures that shed light how neuronal systems can encode external stimuli [5, 6] and thus far have not been explored for learning representations in a generative framework. The key idea in these methods is that representations of inputs that "look alike", should be *closer* to one another, whereas dissimilar data points should have representations that are far apart. Mathematically, the representations in such scenarios can be obtained by solving the following optimization problem:

$$\min_{\boldsymbol{Z}\in\mathcal{C}} d(\mathcal{K}_{\boldsymbol{X}}, \mathcal{K}_{\boldsymbol{Z}}) \tag{1}$$

where, $\boldsymbol{X} \in \mathbb{R}^{N\times T}$ is the input data, $\boldsymbol{Z} \in \mathbb{R}^{K\times T}$ are the learned representations, T denotes the number of samples, N is the dimensionality of the input data, K is the dimensionality of the representation space, $\mathcal{K}_{\boldsymbol{X}}, \mathcal{K}_{\boldsymbol{Z}} \in \mathbb{R}^{T\times T}$ capture the associations between different input samples and representations samples respectively (often called the *kernel* matrix) and d measures the notion of distance / similarity between the two kernel matrices, with the goal being to minimize the distance between two kernels i.e. *match* the two kernels.

Sparse Dictionary Learning is a popular generative framework widely used in the field of signal processing, machine learning and neuroscience to obtain parsimonious representations from the input data while learning a generative map from the representation to the input space (dictionary). Recent work [7] have also focused on extending these approaches to learn manifold approximations. In this paper, we provide a self supervised representation learning framework for dictionary learning. By using properties of bi-level optimization, we show that dictionary learning is equivalent to a self-supervised representation learning problem (shown in equation 1) with constraints on the representation kernel. We show the flexibility of the framework by incorporating different priors on the latent space leading to a piece-wise manifold approximation problem. A key bottleneck in the initial formulation is the need to access the entire dataset to compute the objective and constraints. To overcome this, we formulate a separable version of the problem, allowing us to split the objective over different samples. We propose an Alternate Direction Method of Multipliers (ADMM) based formulation to optimize this objective and show that under certain optimization *schedules*, the dynamics of the algorithm can be connected to classical alternate minimization techniques. The rest of the paper is organized as follows – in section 2, we discuss the kernel matching formulation for sparse dictionary learning and its extension to piece-wise manifold approximation. In section 3, we discuss the optimization

SC would like thank to Kempner Institute for the study of Natural and Artificial Intelligence, Harvard University for funding and compute resources.

algorithm and the different *schedules* for optimization. In section 4, we present experiments on simulated and real data to demonstrate our approach and conclude with discussion about limitations and future work in section 5.

2. SELF SUPERVISED DICTIONARY LEARNING USING KERNEL MATCHING

2.1. Kernel Matching Objective for Sparse Dictionary Learning

We start with a modified sparse dictionary learning problem as defined in [8] which is given by:

$$P_1 : \min_{\boldsymbol{A},\boldsymbol{Z}} \frac{1}{2T}\|\boldsymbol{X}-\boldsymbol{A}\boldsymbol{Z}\|_F^2+\frac{\lambda}{T}\|\boldsymbol{Z}\|_1+\frac{\omega}{2}\|\boldsymbol{A}\|_F^2 \quad \omega > 0 \tag{2}$$

where, $\boldsymbol{A} \in \mathbb{R}^{N\times K}$ is the dictionary. The constraint $\frac{\omega}{2}\|\boldsymbol{A}\|_F^2$ is added to avoid scaling ambiguities in the optimization process. Equation 2 is a bi-level optimization problem in $\boldsymbol{A}$ and $\boldsymbol{Z}$. We use the lemma described below to flip the order of optimization in $\boldsymbol{A}$ and $\boldsymbol{Z}$.

Lemma 1. *Let $f(\boldsymbol{x},\boldsymbol{y})$ by a function with finite minima where $\boldsymbol{x} \in \mathbb{R}^m$ and $\boldsymbol{y} \in \mathbb{R}^n$ for $m, n \in \mathbb{N}$, then the following holds:*

$$\min_{\boldsymbol{x}} \min_{\boldsymbol{y}} f(\boldsymbol{x},\boldsymbol{y}) = \min_{\boldsymbol{y}} \min_{\boldsymbol{x}} f(\boldsymbol{x},\boldsymbol{y})$$

Applying Lemma 1 to equation 2 and solving the inner optimization problem w.r.t $\boldsymbol{A}$, we obtain a closed form optima $\boldsymbol{A}^* = \left[\frac{\boldsymbol{X}\boldsymbol{Z}^T}{T}\left(\frac{\boldsymbol{Z}\boldsymbol{Z}^T}{T}+\omega\boldsymbol{I}\right)^{-1}\right]$. Substituting this back into the objective in equation 2 we arrive at the following proposition:

Proposition 1.1. *Let P_1 be the sparse dictionary problem as defined in equation 2, and Q_1 be the optimization problem described below. Then P_1 and Q_1 are equivalent.*

$$\begin{aligned} Q_1 : \min_{\boldsymbol{Z},\boldsymbol{H}} \; & -\frac{1}{2T^2}\operatorname{Tr}(\underbrace{\boldsymbol{X}^T\boldsymbol{X}}_{\mathcal{K}_{\boldsymbol{X}}}\underbrace{\boldsymbol{Z}^T\boldsymbol{H}^{-1}\boldsymbol{Z}}_{\mathcal{K}_{\boldsymbol{Z}}})+\frac{\lambda}{T}\|\boldsymbol{Z}\|_{1,1} \\ \textit{s.t. } & \boldsymbol{H} = \frac{\boldsymbol{Z}\boldsymbol{Z}^T}{T}+\omega\boldsymbol{I} \end{aligned} \tag{3}$$

We see that equation 3 follows the structure defined in equation 2 where the input kernel, $\mathcal{K}_X = \boldsymbol{X}^T\boldsymbol{X}$ and the representation kernel, $\mathcal{K}_Z = \boldsymbol{Z}^T\boldsymbol{H}^{-1}\boldsymbol{Z}$, and $\boldsymbol{H}^{-1}$ determines the representation kernel structure. The trace term captures the matrix inner product which tries to align / *match* the representation kernel subject to constraints on the representation space. Additionally, solving the objective in equation 3 leads to an **implicit dictionary** learned by the model, given by $A = \frac{1}{T}\boldsymbol{X}\boldsymbol{Z}^T\boldsymbol{H}^{-1}$, determined completely by the learned representations $\boldsymbol{Z}$ and the representation kernel matrix $\boldsymbol{H}^{-1}$.

2.2. Extension to different priors (Manifold Learning)

We adapt the objective in equation 2 to different structural priors imposed on the latent space, which can lead to interesting behaviors. Specifically, we focus on a particular case where constraining the representations on a probability simplex gives rise to manifold learning behavior. We start with the objective proposed in a recent work [7] where a piece-wise linear manifold approximation problem using a dictionary learning framework. The objective is given by:

$$\begin{aligned} P_2 : \min_{\boldsymbol{A},\boldsymbol{Z}} \; & \frac{1}{2T}\|\boldsymbol{X}-\boldsymbol{A}\boldsymbol{Z}\|_F^2+\frac{\omega}{T}\sum_{k=1}^{K}\sum_{l=1}^{T} z_{kl}\|\boldsymbol{x}_l-\boldsymbol{a}_k\|_2^2 \\ \text{s.t.} \; & \sum_{k=1}^{K} z_{kl} = 1,\; z_{kl} \geq 0, \quad \forall \quad l \in \{1,2,\ldots,T\} \end{aligned} \tag{4}$$

The dictionary $\boldsymbol{A}$ here represents the anchor points on the data manifold and piece-wise approximate using convex polygons with these anchor points serving as the vertices of these polygons. Using lemma 1, we can arrive at an equivalent kernel matching formulation as shown in the proposition below.

Proposition 1.2. *Let P_2 be the piece-wise manifold approximation problem as defined in equation 4, and Q_2 be the optimization problem described below. Then P_2 and Q_2 are equivalent.*

$$\begin{aligned} Q_2 : \min_{\boldsymbol{Z},\boldsymbol{H}} \; & -\frac{1}{2T^2}\operatorname{Tr}(\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{Z}^T\boldsymbol{H}^{-1}\boldsymbol{Z}) \\ \textit{s.t.} \; & \boldsymbol{Z}^T\boldsymbol{1}_K = \boldsymbol{1}_T \\ & \boldsymbol{Z} \geq 0 \\ & \boldsymbol{H} = \frac{\boldsymbol{Z}\boldsymbol{Z}^T}{T}+\omega\boldsymbol{D} \textit{ where } \boldsymbol{D} = \textit{diag}\left(\frac{\boldsymbol{Z}\boldsymbol{1}_T}{T}\right) \end{aligned} \tag{5}$$

Equation 5 follows a similar structure as equation 3 and equation 2, with the representations now constraint to a probability simplex. Similar to before, the implicit dictionary learned by the model can be given by $\boldsymbol{A} = \frac{1}{T}\boldsymbol{X}\boldsymbol{Z}^T\boldsymbol{H}^{-1}$, which serve as anchor points on the data manifold.

2.3. A Separable Formulation

A key challenge in solving the optimization problem described in equation 3 and equation 5 is that the objective computation requires access to the entire dataset to compute $\mathcal{K}_{\boldsymbol{X}}$ and $\mathcal{K}_{\boldsymbol{Z}}$ and $\boldsymbol{H}$. We introduce variables $\boldsymbol{W}$ (sample-independent) and $\boldsymbol{P}_i$ (sample-dependent) to separate the objective and constraints over different samples. This is inspired by online formulation proposed in [5, 6] which involves taking the Legendre transformation of the trace term in the objective. This leads us to the following propositions:

Proposition 1.3. *Let R_1 be the optimization problem described below. Then R_1 is equivalent to Q_1.*

$$\begin{aligned} R_1 : \min_{\boldsymbol{z}_i,\boldsymbol{P}_i,\boldsymbol{W}} \; & -\frac{1}{T}\sum_{i=1}^{T}\left(\boldsymbol{x}_i^T\boldsymbol{W}^T\boldsymbol{z}_i\right) \\ & +\frac{1}{2T}\sum_{i=1}^{T}\operatorname{Tr}\left(\boldsymbol{W}^T\boldsymbol{P}_i\boldsymbol{W}\right)+\frac{\lambda}{T}\sum_{i=1}^{T}\|\boldsymbol{z}_i\|_1 \\ \textit{s.t. } & \boldsymbol{P}_i = \boldsymbol{z}_i\boldsymbol{z}_i^T+\omega\mathbf{I} \qquad \forall\, i \in \{1,2,\ldots,T\} \end{aligned} \tag{6}$$

Proposition 1.4. *Let R_2 be the optimization problem described below. Then R_2 is equivalent to Q_2.*

$$R_2 : \min_{\boldsymbol{z}_i, \boldsymbol{P}_i, \boldsymbol{W}} \; -\frac{1}{T}\sum_{i=1}^{T}\left(\boldsymbol{x}_i^T \boldsymbol{W}^T \boldsymbol{z}_i\right) + \frac{1}{2T}\sum_{i=1}^{T}\mathrm{Tr}\left(\boldsymbol{W}^T \boldsymbol{P}_i \boldsymbol{W}\right)$$
$$s.t. \sum_{j=1}^{K} z_{ij} = 1, \; z_{ij} \geq 0$$
$$\boldsymbol{P}_i = \boldsymbol{z}_i \boldsymbol{z}_i^T + \omega \, diag(\boldsymbol{z}_i) \quad \forall \; i \in \{1, 2, \ldots, T\} \tag{7}$$

3. SOLVING THE OPTIMIZATION PROBLEM

In this section, we discuss the optimization procedure to solve the problems described by R_1 and R_2. For brevity, we would discuss the optimization procedure for R_1 which can be adapted for R_2 with small modifications. Since the problems described in equation 6 and equation 7 are separable constraint optimization problems, we resort to Alternate Direction Method of Multipliers (ADMM) [9] to formulate the dynamics for optimization. Consequently, the augmented Lagrangian for R_1 can be written as:

$$\mathbb{L} = \frac{1}{T}\sum_{i=1}^{T}\mathcal{L}_i\left(\boldsymbol{z}_i, \boldsymbol{W}, \boldsymbol{P}_i, \boldsymbol{M}_i\right)$$
$$\text{where, } \mathcal{L}_i = -\boldsymbol{x}_i^T \boldsymbol{W} \boldsymbol{z}_i + \frac{1}{2}\mathrm{Tr}(\boldsymbol{W}^T \boldsymbol{P}_i \boldsymbol{W}) + \lambda \|\boldsymbol{z}_i\|_1 \tag{8}$$
$$+ \mathrm{Tr}(\boldsymbol{M}_i^T\left(\boldsymbol{z}_i \boldsymbol{z}_i^T + \omega \boldsymbol{I} - \boldsymbol{P}_i\right))$$
$$+ \frac{\rho}{2}\|\boldsymbol{z}_i \boldsymbol{z}_i^T + \omega \boldsymbol{I} - \boldsymbol{P}_i\|_F^2$$

The separable formulation of the Lagrangian, allows us to use batch approximations to update the sample independent term $\boldsymbol{W}$ in the ADMM formulation. The full ADMM algorithm for optimizing the Lagrangian is presented in Algorithm 1.

Since the *Z-Step*, *P-Step* and *M-Step* only depend on the current sample i, we can leverage parallel computing to do these steps in parallel over all the samples. The *Z-Step* in the above algorithm consists of a quartic function in $\boldsymbol{z}_i$ for which closed form minimization non-trivial. We discuss proximal gradient based approaches for this step in the Appendix. Similarly, we perform gradient based optimization for the *W-Step*, to avoid matrix inversion in the closed form minimization. Further, algorithm 1 defers slightly from the standard ADMM formulation as the primal variable $\boldsymbol{W}$ is updated (*W-Step*) after the Lagrange multiplier $\boldsymbol{M}_i$ is updated (M-Step). This is possible because the *M-Step* does not depend on the current value of $\boldsymbol{W}$ and vice-versa (See Appendix).

4. EXPERIMENTS

4.1. Optimization dynamics and relation to classical methods

A key bottleneck in the formulation in the ADMM formulation is optimizing the quartic function in the *Z-Step*. To help tackle this issue (further details discussed in the Appendix) and understand how the optimization dynamics of Algorithm 1, we formulate three optimization schedules as follows:

Algorithm 1: ADMM steps for solving R_1 and R_2

Input: data matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_T], \boldsymbol{x}_i \in \mathbb{R}^N$
Output: latent matrix $\boldsymbol{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots \boldsymbol{z}_T]$, $\boldsymbol{P}_i \; \forall \, i \in \{1, 2, \ldots, T\}$
for $k = 1, 2, \ldots$ **do**
  for $i = 1, 2, \ldots T$ **do**
    $\boldsymbol{z}_i^{k+1} \leftarrow \arg\min_{\boldsymbol{z}} \; \mathcal{L}_i(\boldsymbol{z}_i, \boldsymbol{P}_i^k, \boldsymbol{M}_i^k)$ (Z-Step)
    $\boldsymbol{P}_i^{k+1} \leftarrow$ $\arg\min_{\boldsymbol{P}} \; \mathcal{L}_i(\boldsymbol{z}_i^{k+1}, \boldsymbol{P}, \boldsymbol{M}_i^k)$ (P-Step)
    $\boldsymbol{M}_i^{k+1} \leftarrow \boldsymbol{M}_i^k +$ $\rho\left(\frac{\boldsymbol{z}_i^{k+1}\boldsymbol{z}_i^{k+1^T}}{T} + \omega \boldsymbol{I} - \boldsymbol{P}_i^{k+1}\right)$ (M-Step)
  end
  $\boldsymbol{W}^{k+1} \leftarrow$ $\arg\min_{\boldsymbol{W}} \; \frac{1}{T}\sum_{i=1}^{T}\mathcal{L}_i(\boldsymbol{z}_i^{k+1}, \boldsymbol{P}_i^{k+1}, \boldsymbol{W}, \boldsymbol{M}_i^{k+1})$ (W-Step)
end

1. **Schedule 1**: In this schedule, we perform the *P-Step* for every gradient update in the *Z-Step*, i.e. the variable $\boldsymbol{P}_i$ tracks the updates to $\boldsymbol{z}_i$. We perform a modified *M-Step* after the *W-Step*, by updating $\boldsymbol{M}_i^{k+1} \leftarrow \frac{\boldsymbol{W}^{k+1}\boldsymbol{W}^{(k+1)^T}}{2}$. Under this scheme the algorithm performs *alternate minimization with Iterative Soft Thresholding (ISTA)* for the sparse coding step and $\boldsymbol{W}$ acting similar to the dictionary.

2. **Schedule 2**: In this schedule, we perform the *P-Step* for every gradient update in the *Z-Step* before moving to the *M-Step* and *W-Step* as described in Algorithm 1. Under this scheme the algorithm performs a slightly modified version of ISTA for the sparse coding step. Both this and the previous schedule enforce that the constraints on the representation structure captured by $\boldsymbol{P}_i$ are updated as soon as the latent representations $\boldsymbol{Z}$ get updated.

3. **Schedule 3**: In this schedule, we perform the optimization steps as described in Algorithm 1. In this case, $\boldsymbol{P}_i$ is updated only after aggregating the updates to $\boldsymbol{Z}$ over successive iterations in the *Z-step*, affecting the dynamics in the *Z-Step*.

To study the different optimization schedules we generate a synthetic dataset with $N = 30, K = 16$ and $T = 1500$. The true dictionary $\boldsymbol{A}$ is generated from a $\mathcal{N}(0, \boldsymbol{I})$ and the true representations have l_0 norm of 2. For every schedule,

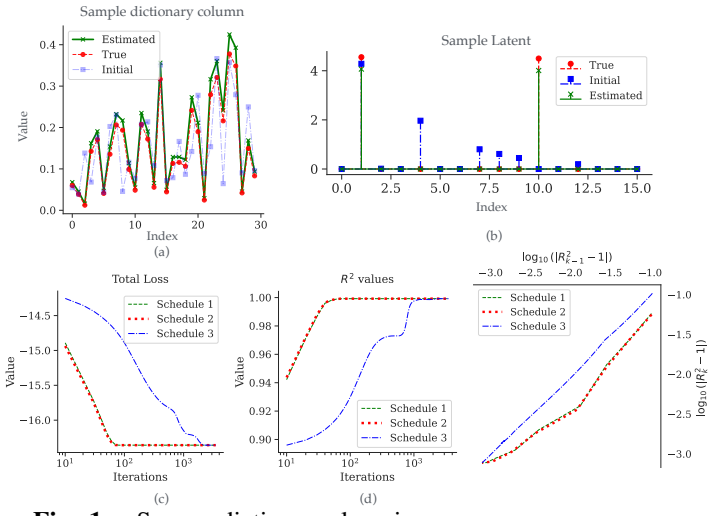

Fig. 1. Sparse dictionary learning o
(b) sample estimated dictionary atom
Total Loss curve (d) R^2 values (e) l
$\log 10(|R^2_{k-1} - 1|)$ (order of convergen
curve ≈ 1) for different optimization sch

we warm start the algorithm by initializi
ter centers obtained from K means on
malize them to unit ball and solve the sp
corresponding to $\boldsymbol{W}_{init}$ to obtain $\boldsymbol{Z}_{init}$
after, we initialize Lagrange multiplier $\boldsymbol{M}$
$\boldsymbol{P}_i^{init} = \boldsymbol{z}_i^{init}\boldsymbol{z}_i^{init^T} + \omega\boldsymbol{I}$ with the batc
$0.5, \eta_{\boldsymbol{W}} = 0.01$, and $\omega = 10^{-4}$ as the cl
rameters. We run the proximal gradient
and updated the parameter $\boldsymbol{W}$ using mi
scent. We run the algorithm for each c
convergence. Figure 1 shows the results
the synthetic dataset alongside a compa
optimization schedules.

Figure 1 (c), (d) shows that both *Schedule 1* and *Schedule 2* have similar rates of convergence and are faster (slope increases as $R^2_k \to 1$) than *Schedule 3*, suggesting tracking the updates to $\boldsymbol{P}_i$ can help in faster convergence. In addition, the order of convergence is approximately linear (Figure 1 (e)).

4.2. Sparse Dictionary Learning on Real Data

We next use the objective in proposition 1.1 to learn representations from a real data set in significantly higher dimension. For this task, we choose the MNIST[10] handwritten digit dataset and patches extracted from natural scenes [11].

Experiment with MNIST: We chose a subset of the MNIST dataset (classes [0, 3, 4, 6, 7], 9000 samples) to optimize the objective in proposition 1.1. The dimension of the representation space K was chosen to be 500, $\lambda = 0.1$, the batch size $B = 128$, $\omega = 10^{-4}$ and $\eta_{\boldsymbol{W}} = 10^{-4}$ (learning rate for $\boldsymbol{W}$ update). We warm start the algorithm through the initialization scheme described in the synthetic data experiment. The proximal gradient for the *Z-Step* is run for 15 iterations, and we run the algorithm for 8000 iterations using *Schedule 1*. We obtain an $R^2 \sim 0.75$ at the end of 8000 iterations. We visualize the estimated dictionary in figure 2 (a) and observe that the atoms of the estimated dictionary resemble the digits in the MNIST dataset indicating the algorithm has learned a good generative map for the data.

Experiment with patches: We train the objective in proposition 1.1 on 4000 samples of each of size 16×16 extracted randomly from 10 natural scene images of size 512×512. The dimension of the representation space K was chosen to be 192[11], $\lambda = 0.9$, the batch size $B = 128$, $\omega = 10^{-4}$ and $\eta_{\boldsymbol{W}} = 10^{-4}$. We again warm start the algorithm similar to previous experiments and let the algorithm run under *Schedule 1* for 5000 iterations, with the proximal gradient for the *Z-Step* performed for 15 iterations respectively. We obtain an

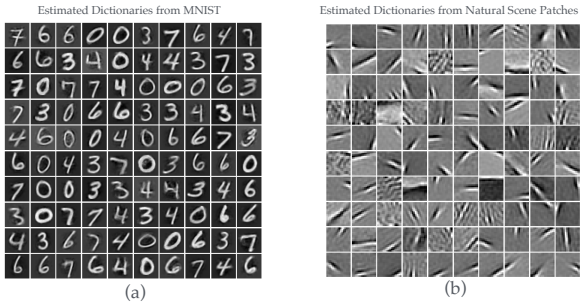

Fig. 2. Estimated atoms of the dictionary learned from the MNIST dataset (a) and patches from natural scenes (b) (We show the first 100 atoms)

4.3. Reconstructing Data Manifolds

Finally, we demonstrate the flexibility of the formulation in equation 2 by using the objective in proposition 1.2 to learn a piece-wise manifold approximation on a synthetic dataset. We use the *moons* dataset [12] with 2500 samples. We also set the noise level to 0.01. K is chosen to be 12, $\omega = 0.1$, $B = 128, \eta_{\boldsymbol{W}} = 2 \times 10^{-3}$. The projected gradient update for the *Z-Step* is performed for 15 iterations and the entire algorithm is run for 5500 iterations under *Schedule 1*. Figure 3(a) shows the estimated atoms of the dictionary on the manifold and the reconstructed data points (in green) alongside a histogram of the L_0 norm of the representations in figure 3 (b) which shows that most samples have an L_0 norm of 2. This suggests that the data manifold can be piece-wise approximated using the lines suggesting that the intrinsic dimensionality of the data manifold is 1.

5. CONCLUSION

We introduce a self supervised framework for dictionary learning where the representations are learned by aligning the input and the latent kernels which in turn leads to an implicit learning of the dictionary through the representations.

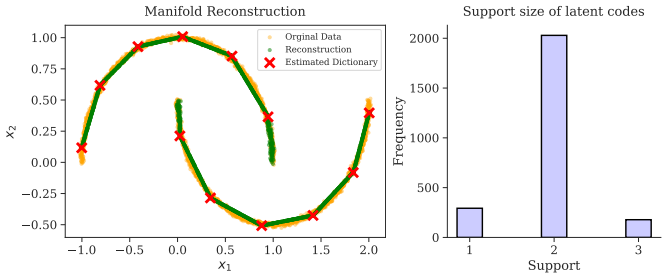

Fig. 3. Reconstruction obtained by solving the objective R_2 for the manifold learning problem (a) and the latent support histogram (b). The atoms form anchor points on the surface of the manifold through which the data points are approximated as a convex combination.

We apply this framework to different priors on the representation space and show that the leaned representations are able to capture good generative maps. To solve the objective, we propose a novel ADMM based optimization algorithm and show that classical alternate minimization techniques for solving dictionary learning are a special case of the ADMM framework under certain optimization schedules. In conclusion, our work provides a novel perspective on dictionary learning by extending the kernel matching objectives to a generative paradigm opening up avenues of exploring neurally plausible architectures for generative modelling by expanding upon existing kernel based representation learning approaches ([5]). Future work would involve investigating the theoretical properties of the algorithm and developing a neurally plausible architecture that could mimic the dynamics as stated in the algorithm.

6. APPENDIX

6.1. Proof of Lemma 1

Proof. Let us assume WLOG, $\min_{\boldsymbol{x}} \min_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y}) > \min_{\boldsymbol{y}} \min_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{y})$. Let $f(\boldsymbol{x}^*, \boldsymbol{y}^*) = \min_{\boldsymbol{x}} \min_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y})$ and $(\boldsymbol{x}^\dagger, \boldsymbol{y}^\dagger) = \min_{\boldsymbol{y}} \min_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{y})$ then, we have $f(\boldsymbol{x}^*, \boldsymbol{y}^*) > f(\boldsymbol{x}^\dagger, \boldsymbol{y}^\dagger)$. However,

$$\begin{aligned} f(\boldsymbol{x}^\dagger, \boldsymbol{y}^\dagger) &\geq \min_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{y}^\dagger) \\ &\geq \min_{\boldsymbol{x}} \min_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y}) \\ &= f(\boldsymbol{x}^*, \boldsymbol{y}^*) \implies \text{A contradiction} \end{aligned}$$

Therefore, the only possibility is $f(\boldsymbol{x}^*, \boldsymbol{y}^*) = f(\boldsymbol{x}^\dagger, \boldsymbol{y}^\dagger)$ or $\min_{\boldsymbol{x}} \min_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y}) = \min_{\boldsymbol{y}} \min_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{y})$ □

6.2. Proof of Proposition 1.1

$$\begin{aligned} R_1 : \min_{\boldsymbol{z}_i, \boldsymbol{P}_i} \min_{\boldsymbol{W}} \; & -\frac{1}{T}\sum_{i=1}^{T} \left(\boldsymbol{x}_i^T \boldsymbol{W}^T \boldsymbol{z}_i\right) \\ & + \frac{1}{2T}\sum_{i=1}^{T} \operatorname{Tr}\left(\boldsymbol{W}^T \boldsymbol{P}_i \boldsymbol{W}\right) + \frac{\lambda}{T}\sum_{i=1}^{T} \|\boldsymbol{z}_i\|_1 \\ \text{s.t. } & \boldsymbol{P}_i = \boldsymbol{z}_i \boldsymbol{z}_i^T + \omega \mathbf{I} \qquad \forall\, i \in \{1, 2, \ldots, T\} \end{aligned}$$

Note, $\boldsymbol{H} = \frac{1}{T}\sum_{i=1}^{T} \boldsymbol{P}_i$. Solving the inner optimization w.r.t $\boldsymbol{W}$ gives $\boldsymbol{W}^* \overset{i=1}{=} \frac{1}{T}\boldsymbol{H}^{-1}\boldsymbol{Z}\boldsymbol{X}^T$. Substituting this back into the objective we get Q_1. The proof of Proposition 1.2 follows similarly.

6.3. Optimization steps for R_1

- **Z-Step**: The Z-Step involves solving the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{z}_i} \; & -\boldsymbol{x}_i^T \boldsymbol{W}^k \boldsymbol{z}_i + \frac{1}{2}\operatorname{Tr}(\boldsymbol{W}^T \boldsymbol{P}_i^k \boldsymbol{W}^k) + \frac{\lambda}{T}\|\boldsymbol{z}_i\|_1 \\ & + \operatorname{Tr}(\boldsymbol{M}_i^{k^T}\left(\boldsymbol{z}_i \boldsymbol{z}_i^T + \omega \boldsymbol{I} - \boldsymbol{P}_i^k\right)) \\ & + \frac{\rho}{2}\|\boldsymbol{z}_i \boldsymbol{z}_i^T + \omega \boldsymbol{I} - \boldsymbol{P}_i\|_F^2 \end{aligned}$$

Implementing proximal gradient method for quartic function

Because the objective is quartic in $\boldsymbol{z}_i$, we resort to gradient based optimization to solve this problem. In order to deal with the non-smooth l_1 norm, we use the proximal gradient method to solve the problem. The update rule for the τ^{th} iteration of the proximal gradient method is given by:

$$\begin{aligned} \boldsymbol{z}_i^{\tau+1} = \mathcal{S}_{\lambda\eta}\big[\boldsymbol{z}_i^\tau - \\ \eta\left(2\rho\left(\boldsymbol{z}_i^\tau \boldsymbol{z}_i^{\tau^T} + \omega \boldsymbol{I} - \boldsymbol{P}_i^k + \frac{\boldsymbol{M}_i^k}{\rho}\right)\boldsymbol{z}_i^\tau - \boldsymbol{W}^k \boldsymbol{x}_i\right)\big] \end{aligned} \tag{9}$$

where S_λ is the element wise soft thresholding operation and $\eta = \frac{1}{L}$ where, $L = \sigma_{\max}(\boldsymbol{W}\boldsymbol{W}^T)$ where $\sigma_{\max}$ denotes the maximum singular value. The value of η as described works well for problems where the objective has a uniform Lipschitz upper bound, which is not the case for the given objective. However, warm starting the algorithm with the initialization scheme described before, allows us to be close to the proximal solution obtained by ISTA, allowing the choice of η to be appropriate for our case.

- **P-Step**: The P-Step update is given as:

$$\boldsymbol{P}_i^{k+1} = \frac{1}{\rho}\left(\boldsymbol{M}^k - \frac{\boldsymbol{W}^k \boldsymbol{W}^{k^T}}{2}\right) + \boldsymbol{z}_i^{k+1} \boldsymbol{z}_i^{(k+1)^T} + \omega \boldsymbol{I} \tag{10}$$

- **W-Step**: The update for $\boldsymbol{W}$ is again done by standard gradient descent as follows:

$$\boldsymbol{W}^{k+1} = \boldsymbol{W}^k - \eta_{\boldsymbol{W}}\left(\frac{1}{B}\sum_{i=1}^{B}\left(-\boldsymbol{z}_i^{k+1}\boldsymbol{x}_i^T + \boldsymbol{P}_i^{k+1}\boldsymbol{W}^k\right)\right) \tag{11}$$

For sparse coding analysis, to avoid tuning the hyperparameter ω, we normalize the rows of $\boldsymbol{W}$ to unit norm after every update to ensure representation magnitudes are correctly estimated.

- **M-Step**: The update for $\boldsymbol{M}_i$ is given by:

$$\boldsymbol{M}_i^{k+1} = \boldsymbol{M}_i^k + \rho \left(\boldsymbol{z}_i^{k+1} \boldsymbol{z}_i^{k+1^T} + \omega \boldsymbol{I} - \boldsymbol{P}_i^{k+1} \right) \tag{12}$$

Note that equation 12 does not depend on the value of $\boldsymbol{W}^k$ and similarly equation equation 11 does not depend on the value of $\boldsymbol{M}_i^k$. This allows us to update $\boldsymbol{W}$ in a batch manner after the $\boldsymbol{M}_i$ update, thereby speeding up the optimization process.

6.4. Optimization Steps for R_2

The optimization step for R_2 is similar to R_1 with slight modifications to the *Z-Step* and *P-Step* to account for the probability simplex constraint. The *Z-Step* is given by:

$$\begin{aligned} z_i^{k+1} = \mathcal{P}_s \big[z_i^k - \eta \big(& -\boldsymbol{W}^k \boldsymbol{x}_i \\ & 2\rho \left(\boldsymbol{z}_i^k \boldsymbol{z}_i^{k^T} + \omega \boldsymbol{D}_i^k - \boldsymbol{P}_i^k + \frac{\boldsymbol{M}_i^k}{\rho} \right) \\ & \rho\omega \operatorname{diag} \left(\boldsymbol{z}_i^k \boldsymbol{z}_i^{k^T} + \omega \boldsymbol{D}_i^k - \boldsymbol{P}_i^k + \frac{\boldsymbol{M}_i^k}{\rho} \right) \mathbf{1} \big) \big] \end{aligned} \tag{13}$$

where, $\boldsymbol{D}_i = \operatorname{diag}(\boldsymbol{z}_i)$ is a diagonal matrix and $\mathcal{P}_s$ is the projection operator onto the probability simplex [13]. The *P-Step* is given by:

$$\boldsymbol{P}_i^{k+1} = \frac{1}{\rho} \left(\boldsymbol{M}^k - \frac{\boldsymbol{W}^k \boldsymbol{W}^{k^T}}{2} \right) + \boldsymbol{z}_i^{k+1} \boldsymbol{z}_i^{(k+1)^T} + \omega \, \boldsymbol{D}_i^{k+1} \tag{14}$$

The *W-Step* and *M-Step* remain the same as described in the previous section.

7. REFERENCES

[1] Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli, "Data2vec: A general framework for self-supervised learning in speech, vision and language," in *International Conference on Machine Learning*. PMLR, 2022, pp. 1298–1312.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[3] Aaron van den Oord, Yazhe Li, and Oriol Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[4] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, June 2005, vol. 1, pp. 539–546 vol. 1, ISSN: 1063-6919.

[5] Cengiz Pehlevan, Anirvan M. Sengupta, and Dmitri B. Chklovskii, "Why Do Similarity Matching Objectives Lead to Hebbian/Anti-Hebbian Networks?," *Neural Computation*, vol. 30, no. 1, pp. 84–124, Jan. 2018.

[6] Kyle Luther and Sebastian Seung, "Kernel similarity matching with Hebbian networks," *Advances in Neural Information Processing Systems*, vol. 35, pp. 2282–2293, Dec. 2022.

[7] Abiy Tasissa, Pranay Tankala, James M. Murphy, and Demba Ba, "K-Deep Simplex: Manifold Learning via Local Dictionaries," *IEEE Transactions on Signal Processing*, vol. 71, pp. 3741–3754, 2023, Conference Name: IEEE Transactions on Signal Processing.

[8] Bahareh Tolooshams and Demba Ba, "Stable and Interpretable Unrolled Dictionary Learning," Aug. 2022, arXiv:2106.00058 [cs, eess, stat].

[9] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, July 2011, Publisher: Now Publishers, Inc.

[10] Li Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[11] Bruno A. Olshausen and David J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, June 1996, Number: 6583 Publisher: Nature Publishing Group.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[13] Andre Martins and Ramon Astudillo, "From softmax to sparsemax: A sparse model of attention and multi-label classification," in *International conference on machine learning*. PMLR, 2016, pp. 1614–1623.