Contrastive Learning Can Find An Optimal Basis For Approximately Invariant Functions

Daniel D. Johnson [1 2] **Ayoub El Hanchi** [2] **Chris J. Maddison** [2]

Abstract

Contrastive learning is a powerful framework for learning self-supervised representations that generalize well to downstream supervised tasks. We show that multiple existing contrastive learning methods can be reinterpeted as learning a positive-definite kernel that approximates a particular *contrastive kernel* defined by the positive pairs. The principal components of the data under this kernel exactly correspond to the eigenfunctions of a positive-pair Markov chain, and these eigenfunctions can be used to build a representation that provably minimizes the worst-case approximation error of linear predictors under the assumption that positive pairs have similar labels. We give generalization bounds for downstream linear prediction using this optimal representation, and show how to approximate this representation using kernel PCA. We also explore kernel-based representations on a noisy MNIST task for which the positive pair distribution has a closed form, and compare the properties of the true eigenfunctions with their learned approximations.

1. Introduction

Contrastive learning models, such as SimCLR (Chen et al., 2020), have shown great success at learning self-supervised representations for downstream supervised learning tasks, by learning to map "positive pairs" (generally augmentations of the same image) close together in an embedding space. Inspired by their success, a variety of theoretical analyses have been proposed, including uniformity and alignment of hyperspherical embeddings (Wang and Isola, 2020), conditional independence structure with landmark embeddings (Tosh et al., 2021), and spectral analysis of an augmentation graph (HaoChen et al., 2021). These analyses are generally quite specific to particular loss functions, and primarily focus on classification tasks. In this work, we generalize and extend these analyses using techniques from kernel methods, showing that multiple contrastive learning methods can be interpreted as approximating a particular "contrastive kernel", and that this kernel can be used to build good representations for downstream classification or regression.

We assume dataset examples are drawn from a distribution $x \sim p(X)$ over a (discrete) set $\mathcal{X}$, and that from each example we sample multiple augmented views $a_1, a_2 \sim p(A|X = x)$ over another (discrete) set $\mathcal{A}$. At pretraining time, we sample positive pairs a_1, a_2 from the distribution $p_+(a_1, a_2) = \sum_x p(x)p(a_1|x)p(a_2|x)$, and negative pairs independently from the distribution $p(a) = \sum_x p(x)p(a|x)$, which we use to train a contrastive learning model; for convenience we assume population access to $p(X)$. Afterward, we seek to approximate a particular target function $g : \mathcal{A} \to \mathbb{R}$ using a linear predictor, given a small number of labeled (augmented) views $(a_i, y_i) \sim p(A)p(Y|A)$ with $E[y_i|a_i] = g(a_i)$. Our main assumption is that the target function g has similar values for positive pairs:

Assumption 1.1 (Approximate invariance / small positive–pair discrepancy)**.** *The target function g satisfies*

$$\mathbb{E}_{p_+}\left[\big(g(a_1) - g(a_2)\big)^2\right] \le \varepsilon.$$

where the expectation is over paired views $p_+(a_1, a_2|x)$ of examples x drawn from the true data distribution $p(x)$.

We prove that a particular representation based on Kernel PCA (Schölkopf et al., 1997) is optimal under this assumption, in the sense that it minimizes least-squares approximation error of the worst-case target function, and give generalization bounds for its performance. We then show that existing contrastive learning methods can be used to approximately recover this representation.

2. Contrastive Learning Models Are Kernels

We start by identifying a common structure between common contrastive learning models and objectives (see Table

[1]Google Research, Toronto ON, Canada [2]Department of Computer Science, University of Toronto, Toronto ON, Canada. Correspondence to: Daniel D. Johnson <ddjohnson@cs.toronto.edu>.

First Workshop of Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML 2022, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

NT-XEnt (Chen et al., 2020; Van den Oord et al., 2018)	Loss	$\mathbb{E}\left[-\log\frac{\widehat{K}_\theta(a_1^+,a_2^+)}{\widehat{K}_\theta(a_1^+,a_2^+)+\sum_{a_i^-}\widehat{K}_\theta(a_1^+,a_i^-)}\right]$
	Kernel	$\widehat{K}_\theta(a_1,a_2) = \exp(h_\theta(a_1)^T h_\theta(a_2)/\tau)$
	Minimum	$\widehat{K}_\theta(a_1,a_2) = \frac{p_+(a_1,a_2)}{p(a_1)p(a_2)} \cdot Z_{[a_1]}$
NT-Logistic (Chen et al., 2020; Tosh et al., 2021)	Loss	$\mathbb{E}\left[-\log\sigma(\log\widehat{K}_\theta(a_1^+,a_2^+))\right] + \mathbb{E}\left[-\log\sigma(-\log\widehat{K}_\theta(a_1^-,a_2^-))\right]$
	Kernel	$\widehat{K}_\theta(a_1,a_2) = \exp(h_\theta(a_1)^T h_\theta(a_2)/\tau)$
	Minimum	$\widehat{K}_\theta(a_1,a_2) = \frac{p_+(a_1,a_2)}{p(a_1)p(a_2)}$
Spectral (HaoChen et al., 2021)	Loss	$\mathbb{E}\left[-2\widehat{K}_\theta(a_1^+,a_2^+)\right]+E\left[(\widehat{K}_\theta(a_1^-,a_2^-))^2\right]$
	Kernel	$\widehat{K}_\theta(a_1,a_2) = h_\theta(a_1)^T h_\theta(a_2)$
	Minimum	$\widehat{K}_\theta(a_1,a_2) = \frac{p_+(a_1,a_2)}{p(a_1)p(a_2)}$

Table 1. Existing contrastive learning objectives, and their interpretations as kernel learning methods. $C_{[a_1]}$ is a equivalence-class-dependent proportionality constant, with $Z_{[a_1]} = Z_{[a_2]}$ whenever $p_+(a_1, a_2) > 0$. See Appendix A for details.

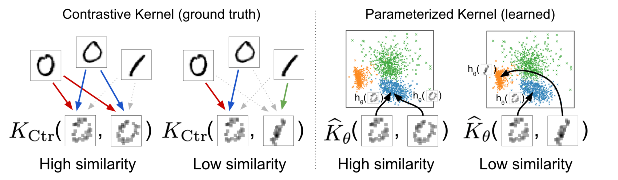
Contrastive Kernel (ground truth)
Parameterized Kernel (learned)
$K_{\mathrm{Ctr}}($, $)$
$K_{\mathrm{Ctr}}($, $)$
$\widehat{K}_\theta($, $)$
$\widehat{K}_\theta($, $)$
$h_\theta($ $)$
High similarity
Low similarity
High similarity
Low similarity

Figure 1. The distribution of positive pairs implicity defines a "contrastive kernel" K_{Ctr}, which assigns high similarity to examples that are likely to be positive pairs. Contrastive learning methods approximate this with a parameterized kernel $\widehat{K}_\theta$, which assigns high similarity to nearby points in a learned embedding space.

1 and Appendix A): many of them are a combination of a parameterized positive-definite kernel and an objective whose minimum (in the infinite data and capacity limit) is exactly the probability ratio $p_+(a_1, a_2)/p(a_1)p(a_2)$ (where all probabilities are with respect to the data distribution). Furthermore, this probability ratio is also a positive-definite kernel (i.e. it is an inner product in a transformed space):

Definition 2.1. *The **contrastive kernel** associated with distributions $p(x)$ and $p(a|x)$ is the probability ratio*

$$K_{Ctr}(a_1, a_2) = \frac{p_+(a_1, a_2)}{p(a_1)p(a_2)} = \langle\phi_{Ctr}(a), \phi_{Ctr}(b)\rangle\,, \qquad (1)$$

where $\phi_{Ctr}(a) \in \mathbb{R}^{|\mathcal{X}|}$ is the vector

$$\left[\frac{p(a|x_1)\sqrt{p(x_1)}}{p(a)} \quad \frac{p(a|x_2)\sqrt{p(x_2)}}{p(a)} \quad \cdots \quad \frac{p(a|x_{|\mathcal{X}|})\sqrt{p(x_{|\mathcal{X}|})}}{p(a)}\right]^T.$$

Data points a_1 and a_2 that are more likely to be a positive pair than a negative pair are mapped to vectors $\phi_{\mathrm{Ctr}}(a_1)$ and $\phi_{\mathrm{Ctr}}(a_2)$ with a high inner product. Conversely, if a_1 and a_2 have zero probability of being a positive pair, $\phi_{\mathrm{Ctr}}(a_1)$ and $\phi_{\mathrm{Ctr}}(a_2)$ are orthogonal. Since multiple contrastive learning approaches have this kernel as their optimum, we can try to understand the behavior of contrastive learning by investigating properties of this kernel. See Figure 1 for a comparison of the true and approximate kernels.

3. Kernel Principal Components Are Positive-Pair Eigenfunctions

To obtain a lower-dimensional representation from the kernel K_{Ctr} or an approximation $\widehat{K}_\theta$, we can use Kernel Principal Components Analysis (Schölkopf et al., 1997), which implicitly computes the directions of maximum variance of a dataset within the kernel's implicit inner product space.[1]

The output of kernel PCA is a set of principal component projection functions $f_1, f_2, \ldots$ and corresponding eigenvalues $\lambda_1, \lambda_2, \ldots$, such that $f_i(a)$ measures the projection of $\phi_{\mathrm{Ctr}}(a)$ onto the ith eigenvector of an implicit covariance matrix, and λ_i measures the variance in that direction. (We will assume the f_i are scaled so that $\mathbb{E}[f_i(a)^2] = 1$.)

Given a dataset of augmentations and a learned kernel $\widehat{K}_\theta$, we can run Kernel PCA and extract the first d projection functions $[\hat{f}_1(a), \hat{f}_2(a), \ldots, \hat{f}_d(a)]$ to construct a "kernel PCA representation" of any new augmented data point a. As our dataset grows, and as our learned kernel $\widehat{K}_\theta$ approaches the true contrastive kernel K_{Ctr}, we would expect this representation to approximate the first d projection functions $[f_1(a), f_2(a), \ldots, f_d(a)]$ of the population covariance of the augmentation distribution under K_{Ctr} (i.e. the covariance of $\phi_{\mathrm{Ctr}}(A)$). It turn out that this representation is a particularly good choice for downstream supervised linear prediction under Assumption 1.1, because these population-level PCA projection functions are closely connected to a particular positive-pair Markov chain described below.

Starting with an example a_1, we could sample another example a_2 proportional to how likely it (a_1, a_2) would be a positive pair, e.g. according to $p_+(a_2|a_1) = \sum_x p(a_2|x)p(x|a_1)$. We can then repeat the process, sampling a_3 according to $p_+(a_3|a_2)$. This defines a Markov chain over the space $\mathcal{A}$ of (augmented) examples, which we visualize in Figure 2. To the extent that some function g is invariant to the chosen augmentations, we would also expect the value of g to change slowly (or not at all) along this random walk: we would expect $g(a_1) \approx g(a_2) \approx g(a_3) \approx \cdots$.

Let $P : \mathbb{R}^{|\mathcal{A}|\times|\mathcal{A}|}$ be the Markov chain transition matrix, such that $P_{ij} = p(a_2 = i|a_1 = j)$. We can then compute the left eigenvectors $\mathbf{f}_i$ and eigenvalues λ'_i of this matrix, such that $\mathbf{f}_i P = \lambda_i \mathbf{f}_i$. Each element $[\mathbf{f}_i]_a$ of an eigenvector $\mathbf{f}_i$ is a value associated with augmentation $a \in \mathcal{A}$, and the eigenvector equation implies that

$$\mathbb{E}_{p_+(a_2|a_1)}\Big[[\mathbf{f}_i]_{a_2}\Big] = \lambda'_i[\mathbf{f}_i]_{a_1}. \qquad (2)$$

[1]For instance, kernel PCA for a dataset $\{a_1, \ldots, a_k\}$ under K_{Ctr} is equivalent to ordinary PCA on $\{\phi_{\mathrm{Ctr}}(a_1), \ldots, \phi_{\mathrm{Ctr}}(a_k)\}$, but does not require computing the vectors $\phi_{\mathrm{Ctr}}(a_i)$ directly.

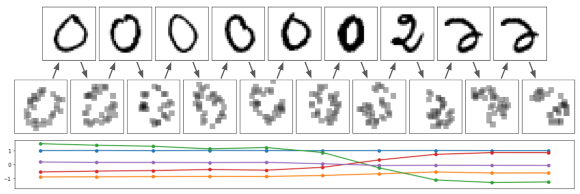

Figure 2. Samples from the positive pair Markov chain for pixel-sampling augmentations. At each step we condition on an augmentation a_t (middle row) to sample an uncorrupted example x_t (top row), then sample a_{t+1} from x_t so that (a_t, a_{t+1}) is a positive pair. Below, we plot the five slowets-varying eigenfunctions $f_1, \dots, f_5$ at each step of the chain.

In other words, for a fixed $a_1 \in \mathcal{A}$, taking the expected value of entries of $\mathbf{f}_i$ over positive-pair neighbors a_2 of a_1 is equivalent to scaling the entry for a_1 by λ_i. Note that we are free to choose each $\mathbf{f}_i$ so that $\mathbb{E}\big[[\mathbf{f}_i]_a^2\big] = 1$.

Our key insight is that the values of the principal component projection functions f_i are the same as the entries of the Markov chain eigenvectors $\mathbf{f}_i$, and that expressing other functions in terms of them allows you to measure invariance:

Theorem 3.1. *$f_i(a) = [\mathbf{f}_i]_a$ and $\lambda_i = \lambda'_i$ for all i and all $a \in \mathcal{A}$. Furthermore, given any function $g : \mathcal{A} \to \mathbb{R}$, if we let $c_i = E[g(a) f_i(a)]$, then the following are true:*

$$g(a) = \sum_i c_i f_i(a), \qquad \mathbb{E}[g(a)^2] = \sum_i c_i^2, \tag{3}$$

$$\mathbb{E}_{p_+}\Big[\big(g(a_1) - g(a_2)\big)^2\Big] = 2\sum_i (1-\lambda_i) c_i^2. \tag{4}$$

This theorem has a few important consequences. First, if we let $g = f_i$ for some i, we find that

$$\mathbb{E}_{p_+}\Big[\big(f_i(a_1) - f_i(a_2)\big)^2\Big] = 2 - 2\lambda_i. \tag{5}$$

Thus, there is a *linear* relationship between the variance λ_i captured by the PCA projection functions and the positive-pair discrepancy of those functions.

Second, by substituting into Equation 2, we find that $\mathbb{E}_{p_+(a_2|a_1)}[f_i(a_2)] = \lambda_i f_i(a_1)$; the population-level PCA projections f_i are thus also *eigenfunctions* of the Markov chain, and the eigenvalues of the kernel's implicit covariance matrix $\mathbb{E}[\phi_{\text{Ctr}}(a)\phi_{\text{Ctr}}(a)^T]$ are also eigenvalues of the Markov transition matrix P.

Third, Equation 4 implies that any target function g satisfying Assumption 1.1 must satsify $2\sum_i(1-\lambda_i)c_i^2 \le \varepsilon$, and thus must have coefficients c_i concentrated on eigenfunctions with λ_i close to 1. Indeed, if $\varepsilon = 0$ (i.e. if g is perfectly invariant to augmentations) then the only eigenfunctions with nonzero weights must be those with $\lambda_i = 1$. Such eigenfunctions identify the subsets of augmentations for which positive pairs always remain within one subset; these are the *communicating classes* of the Markov chain, or equivalently, the connected components of the augmentation graph described by HaoChen et al. (2021).[2]

4. Eigenfunction Representations are Optimal

We now give a more precise analysis of the quality of these eigenfunctions for downstream supervised fine-tuning. We focus on the class of *linear predictors* on top of a particular k-dimensional representation $r : \mathcal{A} \to \mathbb{R}^d$, e.g. we will approximate g with a parameterized function $\hat{g}_\beta(a) = \beta^\top r(a)$. It turns out that the representation consisting of the d eigenfunctions with largest eigenvalues $\lambda_1 \ge \lambda_2 \ge \dots$ is the best choice under two simultaneous criteria.

Proposition 4.1. *Let $\mathcal{F}_r = \{a \mapsto \beta^\top r(a) : \beta \in \mathbb{R}^d\}$ be the family of linear predictors from representation r, and S_ϵ be the set of functions satisfying Assumption 1.1. Let $r_*^d(a) = [f_1(a), f_2(a), \dots, f_d(a)]$ be the representation consisting of the d largest eigenfunctions of the positive pair Markov chain. Then $\mathcal{F}_{r_*^d}$ maximizes the invariance of the least-invariant unit-norm predictor in $\mathcal{F}_{r_*^d}$:*

$$\mathcal{F}_{r_*^d} = \underset{\mathcal{F}\text{ rank } d}{\operatorname{argmin}} \max_{\substack{\hat{g}\in\mathcal{F},\\ \mathbb{E}[\hat{g}(a)^2]=1}} \mathbb{E}_{p_+}\Big[\big(\hat{g}(a_1) - \hat{g}(a_2)\big)^2\Big]. \tag{6}$$

Simultaneously, $\mathcal{F}_{r_*^d}$ minimizes the least-squares approximation error for the worst-case target function in S_ϵ:

$$\mathcal{F}_{r_*^d} = \underset{\mathcal{F}\text{ rank } d}{\operatorname{argmin}} \max_{g\in S_\epsilon} \min_{\hat{g}\in\mathcal{F}} \mathbb{E}_{p(a)}\Big[\big(g(a) - \hat{g}(a)\big)^2\Big]. \tag{7}$$

Equation 6 states that the function class $\mathcal{F}_r$ has an implicit regularization effect: it contains the functions that change as little as possible over positive pairs, relative to their norm. Equation 7 reveals that this function class is also the optimal choice for least-squares approximation of a function satisfying Assumption 1.1. Together, these findings suggest that this representation should give good generalization performance with only a few labeled examples, as long as Assumption 1.1 holds.

Consider the loss function $\ell(\hat{y}, y) := |\hat{y} - y|$, and the associated risk $\mathcal{R}(g) = \mathbb{E}\left[\ell(g(A), Y)\right]$ of a predictor $g : \mathcal{A} \to \mathbb{R}$. Let $g^* \in \operatorname{argmin} \mathcal{R}(g)$ be the optimal predictor with risk $\mathcal{R}^* = \mathcal{R}(g^*)$, and assume it satisfies Assumption 1.1. Expand g^* as a linear combination of the basis of eigenfunctions $(f_i)_{i=1}^K$ as $g^* = \sum_{i=1}^K \beta_i^* f_i$, and define the vector $\beta^* := (\beta_1^*, \dots, \beta_d^*)$. Then the following generalization bound holds for downstream supervised learning:

Proposition 4.2. *Let $(A_i, Y_i)_{i=1}^n$ be i.i.d. samples, choose $R \ge 0$, and consider the constrained empirical risk minimizer $\hat{\beta}_R \in \operatorname{argmin}_{\|\beta\|_2 \le R} n^{-1}\sum_{i=1}^n |\langle \beta, r_d^*(A_i)\rangle - Y_i|$.*

[2]See Levin and Peres (2017, Chapter 12) for further discussion on the eigenfunctions of a Markov chain.

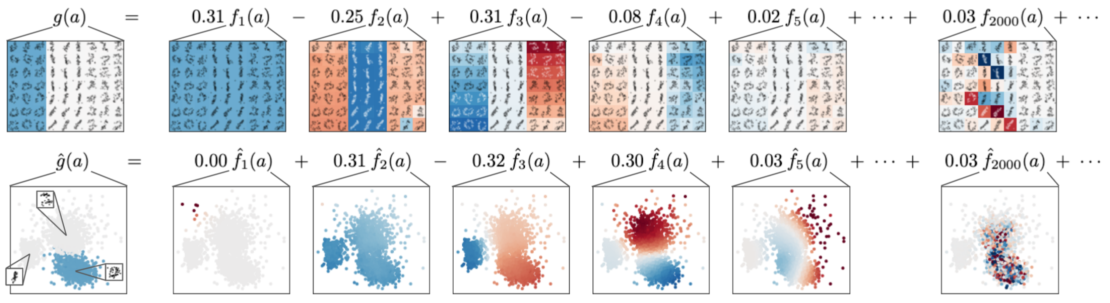

Figure 3. Visualization of eigenfunctions for the MNIST digits 0, 1, and 2 under our pixel-sampling augmentations. Top: A target function (here, an indicator for digit zero) can be expressed as a weighted sum of eigenfunctions of the true positive-pair Markov chain; approximately-invariant target functions must assign more weight to smoother eigenfunctions. Bottom: We can approximate the target function with a linear predictor on the kernel PCA representation of a learned model. As the model approaches the minimum of its objective, these principal components align with the true eigenfunctions. Note the similarity between f_2 and $\hat{f}_3$, and between f_3 and $\hat{f}_4$.

Then the expected excess risk of $\hat{\beta}_R$ is bounded by:

$$\mathbb{E}\left[\mathcal{E}(\hat{\beta}_R)\right] \leq \frac{2dR}{\sqrt{n}} + \sqrt{d}(\|\beta^*\|_2 - R)_+ + \sqrt{\frac{\varepsilon}{2(1-\lambda_{d+1})}}$$

where $\mathcal{E}(\beta) := \mathcal{R}(\beta) - \mathcal{R}^*$ is the excess risk and $(x)_+ :=$ $\max\{x, 0\}$.

Note that $\|\beta^*\|_2^2 = \mathbb{E}[g^*(a)^2]$ by Equation 3, so if $\mathbb{E}[g^*(a)^2] \leq R$ then the second term vanishes. The third term bounds the error incurred by using only the first d eigenfunctions, since $\beta_i^{*2} \leq \frac{\varepsilon}{2(1-\lambda_i)}$ by Equation 4.

5. Related work

Our work is closely connected to the spectral graph theory analysis by HaoChen et al. (2021). Their analysis focuses on the eigenvectors of the normalized adjacency matrix of the augmentation graph, and they introduce the spectral contrastive loss and show that its optimum recovers the top eigenvectors up to an invertible transformation. Our work extends theirs by unifying their loss with other losses under a kernel framework, and showing that the kernel principal components are optimal under Assumption 1.1. (Their eigenvectors turn out to be adjusted versions of our eigenfunctions; see Appendices A.3 and B.1.) We note that our assumption is closely related to the Laplacian matrix of the augmentation graph; similar assumptions have been used before for label propagation (Bengio et al., 2006) and Laplacian filtering (Zhou and Srebro, 2011). This assumption is also used as a "consistency regularizer" for semi-supervised learning (Sajjadi et al., 2016; Laine and Aila, 2016).

Tian (2022) gave a different unification of contrastive losses using a game-theoretic perspective, and showed that linear networks on contrastive losses are related to PCA in the input space (whereas our analysis applies to the unconstrained minimizer of those losses, with PCA in the contrastive kernel's implicit inner product space).

There has been a variety of work on learning kernels with neural networks (Wilson et al., 2016; Sun et al., 2018) and extracting kernels from neural net architectures (Jacot et al., 2018; Shankar et al., 2020; Amid et al., 2022), which may be relevant to contrastive learning in light of our analysis.

6. Experiments

Our theoretical analysis suggests that, to the extent that the learned kernel $\widehat{K}_\theta$ approximates the data-dependent contrastive kernel K_{Ctr}, kernel PCA on a learned kernel $\widehat{K}_\theta$ may yield a good approximation of the optimal eigenfunction basis for downstream prediction. To explore the accuracy of this approximation in practice, and evaluate the quality of the extracted representations, we constructed a toy problem for which the exact contrastive kernel is tractable. We selected a subset of MNIST digits (LeCun et al., 1998) as our set of base images $\mathcal{X}$, then carefully chose the augmentation distribution $p(a|x)$ so that we could compute the exact contrastive kernel K_{Ctr}. Specifically, we defined $p(a|x)$ for each $x \in \mathcal{X}$ by first selecting one of 64 fixed perturbations of x, then sampling a subset of k filled pixels from this perturbed copy. We can then tractably compute $p(a|x)$ for any a and x, enabling us to evaluate K_{Ctr} and also to sample from the positive pair Markov chain. As shown in Figure 2, there is often uncertainty about which original image generated a given augmentation due to the sparsity of pixels.

We next trained three neural network models with varying kernel heads and loss functions: a linear kernel head with the spectral loss (based on HaoChen et al. (2021)); a temperature-normalized exponential head $\exp(h_\theta(a_1)^T h_\theta(a_2)/\tau)$ with a weighted combination of the NT-XEnt and NT-Logistic losses (constraining $h_\theta(a)$ to be unit-norm, based on Chen et al. (2020)); and a *scaled* temperature-normalized exponential head $\exp(h_\theta(a_1)^T h_\theta(a_2)/\tau) \cdot s_\theta(a_1)s_\theta(a_2)$, which includes a learned adjustment $s_\theta : \mathcal{A} \to \mathbb{R}_+$, again trained with a

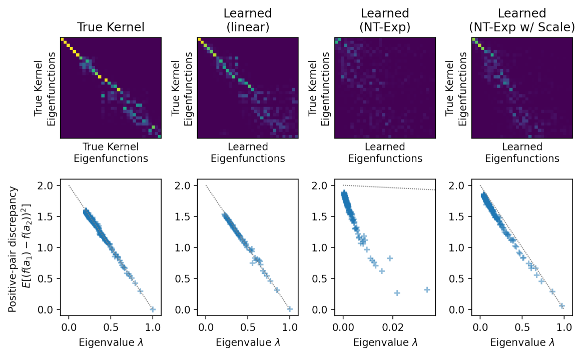

Figure 4. Alignments and discrepancy relationships for the MNIST task with $k = 20$. Top row: Alignment (squared dot products) of the first 32 principal component projection functions between runs of Kernel PCA, with perfect alignment corresponding to a diagonal matrix. Left column shows alignment between two independent runs of KPCA on K_{Ctr}; other columns show alignment between KCPA on K_{Ctr} and KPCA on a learned kernel $\widehat{K}_\theta$. Bottom row: Relationship between eigenvalue λ and positive-pair discrepancy for the first 256 principal components (omitting any with $\lambda = 0$), with the prediction from Equation 5 shown as a gray dashed line.

weighted combination of the NT-XEnt and NT-Logistic losses. All models used a simplified ResNet architecture followed by a fully-connected "projection head", and were trained on 512 MNIST digits from each of the 10 classes. For visualization purposes, we also trained a model with a two-dimensional embedding space and a scaled rational quadratic kernel head on MNIST digits 0, 1, and 2 only; this model is shown in Figures 1 and 3.

Properties of extracted principal components. We used Kernel PCA over a random set of augmentations to to estimate the first 256 principal component projection functions for K_{Ctr} and for each of our learned kernels. We then investigated (a) whether the learned principal components were aligned with the principal components of K_{Ctr}, and (b) whether their eigenvalues are related to their positive-pair discrepancies (estimated from a random sample of positive pairs) as predicted by Equation 5.

The results of these comparisons are shown in Figure 4, for $p(a|x)$ using $k = 20$ sampled pixels per digit. Since Kernel PCA is performed on a random subset, there is some measurement noise; thus even two runs of KPCA on K_{Ctr} produce slightly different results. The linear kernel with spectral loss is able to recover the top principal components quite accurately, and also shows the expected relationship between eigenvalues and positive-pair discrepancies. The temperature-normalized exponential kernel is less well-behaved, but this appears to be due to the constrained form of the kernel head; adding the scale adjustment $s_\theta(a_1)s_\theta(a_2)$ leads to better alignment and to the expected linear eigenvalue-discrepancy relationship. We also found that increasing augmentation strength (smaller k) leads to a better alignment for all kernels, whereas decreasing it (larger k) makes alignment worse; see Appendix D.6.

	Classification			Regression		
# of sampled pixels $k =$	10	20	50	10	20	50
True Kernel (K_{Ctr}) PCA	0.564	0.384	0.178	0.722	0.602	0.369
Learned (Linear):						
Kernel PCA	0.589	0.398	0.254	0.724	0.603	0.362
ResNet Emb.	0.553	0.375	0.229	0.730	0.567	0.459
Learned (NT-Exp):						
Kernel PCA	0.610	0.380	0.392	0.730	0.568	0.608
ResNet Emb.	0.553	0.370	0.203	0.732	0.570	0.476
Learned (NT-Exp w/ Scale):						
Kernel PCA	0.583	0.392	0.276	0.718	0.552	0.524
ResNet Emb.	0.575	0.380	0.206	0.742	0.599	0.522

Table 2. Classification error (fraction misclassified) and regression error (squared error) on MNIST task with multinomial augmentations, across augmentation strengths $k = 10, k = 20, k = 50$.

Downstream prediction ability. We next compare the quality of various representations for two downstream prediction tasks: classification with a linear layer, and linear least-squares regression on the one-hot indicator vectors for each digit class. We consider three types of representation: the PCA projection functions for K_{Ctr}, the PCA projection functions for each learned kernel $\widehat{K}_\theta$, and the intermediate layer embedding vector between the ResNet layers and the projection head for each model as proposed by Chen et al. (2020). For each, we fit a regularized linear predictor on 160 labeled training examples (16 augmented samples from each class), using 160 additional validation examples to tune the regularization strength. For PCA representations, we additionally tune the representation dimension d.

The results are shown in 2. Performance is fairly similar across representations, suggesting that the contrastive kernel K_{Ctr} captures much of the variability between augmentation strengths, although some learned representations achieve better accuracy. Interestingly, adding the scale parameter to the NT-Exp model improves the kernel PCA representation but degrades performance for the ResNet embedding.

7. Discussion

We have shown that multiple existing contrastive objectives approximate a particular kernel, and that its principal component projection functions give an optimal representation for supervised linear prediction under the assumption that positive pairs have similar labels. This analysis is based on the "worst" target function that satisfies our assumption; if we have additional knowledge about the target function, incorporating it may improve the representation. Empirically, we find that Kernel PCA with learned models can produce good approximations of the largest-eigenvalue principal components, but that the quality of the approximation depends on the constraints of the kernel head as well as the strength of the augmentations. Overall, we hope that our kernel and Markov chain perspective is a useful lens for further study of contrastive learning representations.

References

Ehsan Amid, Rohan Anil, Wojciech Kotłowski, and Manfred K Warmuth. Learning from randomly initialized neural network features. *arXiv preprint arXiv:2202.06438*, 2022.

Francis Bach. Learning theory from first principles, 2021.

Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. Label propagation and quadratic criterion. In *Semi-Supervised Learning*, 2006.

Rajendra Bhatia. *Matrix analysis*, volume 169. Springer Science & Business Media, 2013.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

Jeff Z HaoChen, Colin Wei, Adrien Gaidon, and Tengyu Ma. Provable guarantees for self-supervised deep learning with spectral contrastive loss. *Advances in Neural Information Processing Systems*, 34, 2021.

Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2020. URL `http://github.com/google/flax`.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Sham M. Kakade, Karthik Sridharan, and Ambuj Tewari. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. In *NIPS*, 2008.

Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The MNIST database of handwritten digits, 1998. URL `http://yann.lecun.com/exdb/mnist/`.

David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In *International Conference on Machine Learning*, pages 5171–5180. PMLR, 2019.

Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. *Advances in neural information processing systems*, 29, 2016.

Meyer Scetbon and Zaid Harchaoui. A spectral analysis of dot-product kernels. In *International Conference on Artificial Intelligence and Statistics*, pages 3394–3402. PMLR, 2021.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International conference on artificial neural networks*, pages 583–588. Springer, 1997.

Vaishaal Shankar, Alex Fang, Wenshuo Guo, Sara Fridovich-Keil, Jonathan Ragan-Kelley, Ludwig Schmidt, and Benjamin Recht. Neural kernels without tangents. In *International Conference on Machine Learning*, pages 8614–8623. PMLR, 2020.

Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. *Advances in neural information processing systems*, 29, 2016.

Shengyang Sun, Guodong Zhang, Chaoqi Wang, Wenyuan Zeng, Jiaman Li, and Roger Grosse. Differentiable compositional kernel learning for Gaussian processes. In *International Conference on Machine Learning*, pages 4828–4837. PMLR, 2018.

Yuandong Tian. Deep contrastive learning is provably (almost) principal component analysis. *arXiv preprint arXiv:2201.12680*, 2022.

Christopher Tosh, Akshay Krishnamurthy, and Daniel Hsu. Contrastive learning, multi-view redundancy, and linear models. In *Algorithmic Learning Theory*, pages 1179–1206. PMLR, 2021.

Aaron Van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pages 9929–9939. PMLR, 2020.

Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. *Advances in neural information processing systems*, 13, 2000.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR, 2016.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742, 2018.

Xueyuan Zhou and Nathan Srebro. Error analysis of Laplacian eigenmaps for semi-supervised learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 901–908. JMLR Workshop and Conference Proceedings, 2011.

A. Existing objectives are minimized by the contrastive kernel

In this section, we show that the contrastive kernel is the optimum of the objectives shown in Table 1 (although this optimum is not unique for cross-entropy loss). Throughout this section, we use $p(X)$ to denote the true data distribution over unperturbed examples, $p(A|X)$ to denote the distribution of augmented views conditioned on a particular unperturbed example, and $p(A)$ to denote the marginal distribution of augmented views, e.g.

$$p(A = a) = \sum_{x \in \mathcal{X}} p(X = x)p(A = a|X = x).$$

We use $p_+(A_1, A_2)$ to denote the positive pair distribution induced by $p(X)$ and $p(A|X)$, defined by

$$p_+(A_1 = a_1, A_2 = a_2) = \sum_{x \in \mathcal{X}} p(X = x)p(A = a_1|X = x)p(A = a_2|X = x).$$

For notational convenience, we will use shorthand $p(x)$ for $p(X = x)$, $p(a)$ for $p(A = a)$, $p_+(a_1, a_2)$ for $p_+(A_1 = a_1, A_2 = a_2)$, and so on.

A.1. NT-XEnt and the InfoNCE objective

The NT-XEnt objective described by Chen et al. (2020) (and previously used by Sohn (2016); Van den Oord et al. (2018); Wu et al. (2018)) has the form

$$\mathcal{L}_{\text{NT-Xent}}(\theta) = \mathbb{E}_{(a_1^+, a_2^+) \sim p_+(A_1, A_2), a_i^- \sim p(A)} \left[-\log \frac{\exp(h_\theta(a_1^+)^T h_\theta(a_2^+)/\tau)}{\exp(h_\theta(a_1^+)^T h_\theta(a_2^+)/\tau) + \sum_{a_i^-} \exp(h_\theta(a_1^+)^T h_\theta(a_i^-)/\tau)} \right].$$

For simplicity, we assume that all of the negative samples a_i^- are drawn independently from the marginal distribution when computing the loss for a_1^+ and a_2^+. (In practice, implementations often generate negative samples by taking elements of other positive pairs, e.g. $(a_1^-, a_2^-) \sim p_+(a_1^-, a_2^-)$, $(a_3^-, a_4^-) \sim p_+(a_3^-, a_4^-)$, and so on.)

We can decompose this objective into two parts: an InfoNCE-like loss (Van den Oord et al., 2018)

$$\mathcal{L}_{\text{InfoNCE}}(\widehat{K}_\theta) = \mathbb{E}_{(a_1^+, a_2^+) \sim p_+(A_1, A_2), a_i^- \sim p(A)} \left[-\log \frac{\widehat{K}_\theta(a_1^+, a_2^+)}{\widehat{K}_\theta(a_1^+, a_2^+) + \sum_{a_i^-} \widehat{K}_\theta(a_1^+, a_i^-)} \right]$$

combined with a particular parameterized function

$$\widehat{K}_\theta(a_1, a_2) = \exp\left(\frac{h_\theta(a_1)^T h_\theta(a_2)}{\tau} \right).$$

where $h_\theta : \mathcal{A} \to \mathbb{R}^{n+1}$ maps inputs to points on the n-dimensional hypersphere.

We first observe that $\widehat{K}_\theta(a_1, a_2)$ defines a positive definite kernel, within the family of "dot product kernels". Indeed, when h_θ is restricted to the unit hypersphere, this parameterization is equivalent to the squared-exponential kernel (also called radial-basis-function kernel)

$$\widehat{K}_\theta(a_1, a_2) = \exp\left(\frac{1 - \frac{1}{2}\|h_\theta(a_1) - h_\theta(a_2)\|^2}{\tau} \right) = \exp(1/\tau) \exp\left(-\frac{\|h_\theta(a_1) - h_\theta(a_2)\|^2}{2\tau} \right)$$

using the identity $\|h_\theta(a_1) - h_\theta(a_2)\|^2 = 2 - 2h_\theta(a_1)^T h_\theta(a_2)$. Although this kernel is positive definite, it has "infinite rank" and cannot be expressed as an inner product of finite-dimensional embedding vectors; nevertheless, we can still run algorithms such as Kernel PCA on a finite dataset. (See Bach (2021, Chapter 7) for some additional background on positive-definite kernels, and Scetbon and Harchaoui (2021) for discussion of other dot product kernels on the unit hypersphere.)

We next discuss the minimum of the NT-XEnt objective, under the unconstrained setting where we allow $\widehat{K}_\theta(a_1, a_2)$ to be an arbitrary symmetric function. The InfoNCE loss, in its more general form, is not necessarily symmetric: it is based on a distribution of contexts $p(C)$, positive samples $p(A|C)$, and negative samples drawn from the marginal distribution $P(A)$, and is given by

$$\mathcal{L}_{\text{InfoNCE}}(f_\theta) = \mathbb{E}_{c\sim p(c), a^+\sim p(A|C=c), a_i^- \sim p(A)}\left[-\log \frac{f_\theta(c, a^+)}{f(c, a^+) + \sum_{a_i^-} f_\theta(c, a_i^-)}\right]$$

Van den Oord et al. (2018) show that every minimizer of this objective is of the form

$$f_*(c, a) = \frac{p(a|c)}{p(a)} \cdot z(c) = \frac{p(a, c)}{p(a)p(c)} \cdot z(c)$$

for some function $z(c)$ that does not depend on a. In other words, holding c fixed, $f_*(c, a) \propto \frac{p(a,c)}{p(a)p(c)}$. Intuitively, this is because the exact probability of (c, a^+) being the positive pair given c and the set $\{a^+, a_1^-, \dots, a_K^-\}$ is also proportional to this density ratio, and the InfoNCE objective is a cross-entropy objective for identifying the positive pair. (See also Poole et al. (2019) for a different proof.)

In the case of the NT-XEnt contrastive objective, we choose the context C to be one of the augmentations A_1^+, and the positive sample to be the other augmentation A_2^+ drawn according to $p_+(A_2^+|A_1^+)$. We furthermore restrict our attention to symmetric functions $\widehat{K}_\theta$, e.g. functions for which $\widehat{K}_\theta(a_1, a_2) = \widehat{K}_\theta(a_2, a_1)$. In this case, the minimizer is

$$\widehat{K}_*(a_1, a_2) = \frac{p_+(a_1, a_2)}{p(a_1)p(a_2)} \cdot z(a_1) = \frac{p_+(a_1, a_2)}{p(a_1)p(a_2)} \cdot z(a_2).$$

so we must have $z(a_1) = z(a_2)$ for any pair (a_1, a_2) with positive probability under p_+.

If we assume that the positive pair Markov chain is irreducible, e.g. that there is a single communicating class and it is possible to reach any augmentation in $\mathcal{A}$ from any other augmentation over a long enough trajectory, then the function $z(a)$ must be constant everywhere, and thus

$$\widehat{K}_*(a_1, a_2) = f_*(a_1, a_2) = \frac{p_+(a_1, a_2)}{p(a_1)p(a_2)} \cdot Z$$

for some $Z \in \mathbb{R}^+$. In this case, $\widehat{K}_*$ is equivalent to K_{Ctr} up to a scaling constant.

If the Markov chain has multiple communicating classes (e.g. positive pairs are only sampled within a single communicating class), the function $z(a)$ may assign a different value to different equivalence classes. Nevertheless, any such minimizer is still a kernel, since we can write it as

$$\begin{aligned}\widehat{K}_*(a_1, a_2) &= \left\langle \sqrt{z(a_1)} \begin{bmatrix} \frac{p(a_1|x_1)\sqrt{p(x_1)}}{p(a_1)} \\ \frac{p(a_1|x_2)\sqrt{p(x_2)}}{p(a_1)} \\ \vdots \\ \frac{p(a_1|x_{|\mathcal{X}|})\sqrt{p(x_{|\mathcal{X}|})}}{p(a_1)} \end{bmatrix}, \sqrt{z(a_2)} \begin{bmatrix} \frac{p(a_2|x_1)\sqrt{p(x_1)}}{p(a_2)} \\ \frac{p(a_2|x_2)\sqrt{p(x_2)}}{p(a_2)} \\ \vdots \\ \frac{p(a_2|x_{|\mathcal{X}|})\sqrt{p(x_{|\mathcal{X}|})}}{p(a_2)} \end{bmatrix} \right\rangle \\ &= \left\langle \sqrt{z(a_1)} \cdot \phi_{\text{Ctr}}(a_1),\ \sqrt{z(a_2)} \cdot \phi_{\text{Ctr}}(a_2) \right\rangle\end{aligned}$$

Indeed, such a minimizer is equivalent to K_{Ctr} except that it scales the inner product by the value of $z(a)$ for each communicating class. It is still possible to extract the set of Markov chain eigenfunctions from the set of principal components of this kernel, although one must correct for the scaling factor when computing the eigenvalues; see Appendix B.2 for details. Alternatively, one can ensure a unique minimum by combining the NT-Xent/InfoNCE loss with either the spectral or logistic losses (discussed below).

A.2. Logistic losses and NT-Logistic

Logistic losses have also been proposed for contrastive learning, including the NT-Logistic objective as described in Chen et al. (2020) and other versions described by Mikolov et al. (2013) and Tosh et al. (2021). Such losses take the form

$$\mathcal{L}_{\text{Logistic}}(f_\theta) = \mathbb{E}_{(a_1^+, a_2^+)\sim p_+(A_1, A_2)}\left[-\log \sigma(f_\theta(a_1^+, a_2^+))\right] + \mathbb{E}_{a_1^-\sim p(A), a_2^- \sim p(A)}\left[-\log \sigma(-f_\theta(a_1^-, a_2^-))\right]$$

where negative samples are drawn independently from the marginal distribution $p(A)$. Tosh et al. (2021) motivates this loss based on a binary classification: choose a label Y to be 0 or 1 with probability $1/2$ each, sample a positive pair $(a_1, a_2) \sim p_+(A_1, A_2)$ if $Y = 1$ and a negative pair $a_1 \sim p(A), a_2 \sim p(A)$ if $Y = 0$, then use a learned model to predict Y given the pair. The minimizer of this loss is then the conditional log-odds-ratio

$$f_*(a_1, a_2) = \log \frac{p(Y=1|a_1,a_2)}{p(Y=0|a_1,a_2)} = \log \frac{p(a_1,a_2|Y=1)p(Y=1)}{p(a_1,a_2|Y=0)p(Y=0)} = \log \frac{p_+(a_1,a_2)\cdot\frac{1}{2}}{p(a_1)p(a_2)\cdot\frac{1}{2}} = \log \frac{p_+(a_1,a_2)}{p(a_1)p(a_2)}.$$

For the particular case of the NT-Logistic objective, we parameterize f_θ as

$$f_\theta(a_1, a_2) = \log \hat{K}_\theta(a_1, a_2) = \frac{h_\theta(a_1)^T h_\theta(a_2)}{\tau}$$

where we again define

$$\widehat{K}_\theta(a_1, a_2) = \exp\left(\frac{h_\theta(a_1)^T h_\theta(a_2)}{\tau}\right).$$

The optimum (if we ignore the constraints of this particular form of $\hat{K}$ and minimize over all functions of two variables) is then

$$\hat{K}_*(a_1, a_2) = \frac{p_+(a_1,a_2)}{p(a_1)p(a_2)}.$$

Note that in this case there is no proportionality constant.

A.3. The Spectral Contrastive Loss

HaoChen et al. (2021) propose the Spectral Contrastive Loss as an alternative to other contrastive losses with provable performance guarantees. The loss is defined as

$$\mathcal{L}_{\text{Spectral}}(\widehat{K}_\theta) = -2 \cdot \mathbb{E}_{(a_1^+, a_2^+)\sim p_+(A_1,A_2)}\left[\widehat{K}_\theta(a_1^+, a_2^+)\right] + \mathbb{E}_{a_1^-\sim p(A), a_2^-\sim p(A)}\left[\widehat{K}_\theta(a_1^-, a_2^-)^2\right]$$

where they choose

$$\widehat{K}_\theta(a_1, a_2) = h_\theta(a_1)^T h_\theta(a_2).$$

for a learned embedding function $h_\theta : \mathcal{A} \to \mathbb{R}^d$. We note that this directly satisfies the definition of a kernel, in that it is an inner product in a transformed space. One interesting property of this kernel approximation is that it can be negative, whereas the exponential-based kernel approximations in the previous sections are always nonnegative.

HaoChen et al. show that this loss can be rewritten as

$$\begin{aligned}
\mathcal{L}_{\text{Spectral}}(\widehat{K}_\theta) &= \sum_{a_1,a_2}\left(-2p_+(a_1,a_2)\widehat{K}_\theta(a_1,a_2) + p(a_1)p(a_2)\widehat{K}_\theta(a_1,a_2)^2\right) \\
&= \sum_{a_1,a_2}\left(\frac{p_+(a_1,a_2)^2}{p(a_1)p(a_2)} - 2p_+(a_1,a_2)\widehat{K}_\theta(a_1,a_2) + p(a_1)p(a_2)\widehat{K}_\theta(a_1,a_2)^2\right) - \sum_{a_1,a_2}\frac{p_+(a_1,a_2)^2}{p(a_1)p(a_2)} \\
&= \sum_{a_1,a_2}\left(\frac{p_+(a_1,a_2)}{\sqrt{p(a_1)p(a_2)}} - \sqrt{p(a_1)p(a_2)}\widehat{K}_\theta(a_1,a_2)\right)^2 - C,
\end{aligned}$$

where $C = \sum_{a_1,a_2} \frac{p_+(a_1,a_2)^2}{p(a_1)p(a_2)}$ is a constant independent of the model.

If we again ignore the constraints on $\hat{K}_\theta$, the minimum of the spectral loss must occur when

$$\frac{p_+(a_1,a_2)}{\sqrt{p(a_1)p(a_2)}} = \sqrt{p(a_1)p(a_2)}\widehat{K}_\theta(a_1,a_2),$$

for all a_1 and a_2, or in other words, when

$$\widehat{K}_\theta(a_1,a_2) = \frac{p_+(a_1,a_2)}{p(a_1)p(a_2)} = K_{\text{Ctr}}(a_1,a_2).$$

HaoChen et al. continue by expanding their definition of $\widehat{K}_\theta$ for a fixed representation d, and showing that it relates to the spectral decomposition of a particular augmentation graph. It turns out that this decomposition is equivalent to our decomposition in terms of eigenfunctions except for a scaling factor of $p(a)^{1/2}$; we discuss this connection more in Appendix B.1.

B. Relationship between contrastive kernel and Markov chain eigenfunctions

In this section, we describe the relationship between the contrastive kernel principal components and the Markov chain eigenfunctions in more detail. We start by introducing some notation that will be useful.

Throughout, we will identify functions $f : \mathcal{A} \to \mathbb{R}$ as vectors $\mathbf{f} : \mathbb{R}^{\mathcal{A}}$, which will allow us to use matrix notation for many of the relevant quantities. We will also use $\mathbf{e}_i$ to represent the vector that has a one at the ith position and zeros in all other positions.

We will let

$$D_X = \operatorname{diag}(p(x_1), p(x_2), \dots, p(x_{\{}|\mathcal{X}|\})),$$
$$D_A = \operatorname{diag}(p(a_1), p(a_2), \dots, p(a_{\{}|\mathcal{A}|\})),$$

be diagonal matrices containing the marginal probabilities of each element in $\mathcal{X}$ and $\mathcal{A}$, respectively, under the true data distribution. We will assume that the distribution has full support, and thus both D_X and D_A are invertible. We also define the matrices

$$[P_{X,A}]_{x,a} = p(x,a) \qquad [P_{X\to A}]_{x,a} = p(a|x) \qquad [P_{X\leftarrow A}]_{x,a} = p(x|a)$$
$$[P_{A,X}]_{a,x} = p(x,a) \qquad [P_{A\leftarrow X}]_{a,x} = p(a|x) \qquad [P_{A\to X}]_{a,x} = p(x|a)$$

Equivalently

$$P_{X\to A} = D_X^{-1}\, P_{X,A}, \qquad P_{X\leftarrow A} = P_{X,A}\, D_A^{-1},$$
$$P_{A\leftarrow X} = (P_{X\to A})^\top, \qquad P_{A\to X} = (P_{X\leftarrow A})^\top.$$

From these, we can construct the positive pair probability matrix

$$P_{A,A} = P_{A\leftarrow X}\, D_X\, P_{X\to A}$$
$$[P_{A,A}]_{i,j} = p(A_1 = i, A_2 = j) = \sum_x p(A=i|x)p(x)p(A=j|x).$$

The Contrastive Kernel. Writing the contrastive kernel K_{Ctr} in matrix form, such that $[K_{\text{Ctr}}]_{i,j} = K_{\text{Ctr}}(i,j)$, our definition $K_{\text{Ctr}}(a_1,a_2) = \frac{p_+(a_1,a_2)}{p(a_1)p(a_2)}$ becomes the matrix equation

$$K_{\text{Ctr}} = D_A^{-1} P_{A,A} D_A^{-1}.$$

One way to expand K_{Ctr} is as a product

$$K_{\text{Ctr}} = D_A^{-1} P_{A\leftarrow X} D_X P_{X\to A} D_A^{-1} = \left(D_X^{1/2} P_{X\to A} D_A^{-1}\right)^\top \left(D_X^{1/2} P_{X\to A} D_A^{-1}\right) = \Phi_{\text{Ctr}}^\top \Phi_{\text{Ctr}}$$

where $\Phi_{\text{Ctr}} \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ is a matrix whose columns are given by ϕ_{Ctr}:

$$\Phi_{\text{Ctr}} = D_X^{1/2} P_{X\to A} D_A^{-1} = \begin{bmatrix}\phi_{\text{Ctr}}(a_1) & \phi_{\text{Ctr}}(a_2) & \dots & \phi_{\text{Ctr}}(a_{|\mathcal{A}|})\end{bmatrix}$$
$$= \begin{bmatrix} \frac{p(a_1|x_1)\sqrt{p(x_1)}}{p(a_1)} & \frac{p(a_2|x_1)\sqrt{p(x_1)}}{p(a_2)} & \dots & \frac{p(a_{|\mathcal{A}|}|x_1)\sqrt{p(x_1)}}{p(a_{|\mathcal{A}|})} \\ \frac{p(a_1|x_2)\sqrt{p(x_2)}}{p(a_1)} & \frac{p(a_2|x_2)\sqrt{p(x_2)}}{p(a_2)} & \dots & \frac{p(a_{|\mathcal{A}|}|x_2)\sqrt{p(x_2)}}{p(a_{|\mathcal{A}|})} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{p(a_1|x_{|\mathcal{X}|})\sqrt{p(x_{|\mathcal{X}|})}}{p(a_1)} & \frac{p(a_2|x_{|\mathcal{X}|})\sqrt{p(x_{|\mathcal{X}|})}}{p(a_2)} & \dots & \frac{p(a_{|\mathcal{A}|}|x_{|\mathcal{X}|})\sqrt{p(x_{|\mathcal{X}|})}}{p(a_{|\mathcal{A}|})} \end{bmatrix}.$$

Equivalently, we have $\phi_{\text{Ctr}}(a) = \Phi_{\text{Ctr}}\mathbf{e}_a$.

Since $K_{\text{Ctr}}(a_1, a_2) = \phi_{\text{Ctr}}(a_1)^\top \phi_{\text{Ctr}}(a_2)$, ϕ_{Ctr} is called a *feature map* for K_{Ctr}. Note that there are multiple possible feature maps for K_{Ctr}: given any orthonormal matrix Q, the function $\phi_Q(a) = Q\phi_{\text{Ctr}}(a)$ is also a feature map for the kernel, since

$$\phi_Q(a_1)^\top \phi_Q(a_1) = \phi_{\text{Ctr}}(a)^\top Q^\top Q \phi_{\text{Ctr}}(a) = \phi_{\text{Ctr}}(a)^\top \phi_{\text{Ctr}}(a) = K_{\text{Ctr}}(a_1, a_2).$$

Performing kernel PCA under K_{Ctr} is equivalent to performing ordinary PCA over one of its feature maps (since the principal component projection functions are independent of the particular feature map chosen). We thus focus on analyzing the principal component projection functions for the feature map ϕ_{Ctr}.

The population level principal components are the eigenvectors of the (uncentered) covariance matrix

$$\Sigma = \mathbb{E}_{p(a)}[\phi_{\text{Ctr}}(a)\phi_{\text{Ctr}}(a)^\top] = \mathbb{E}_{p(a)}[\Phi_{\text{Ctr}}\mathbf{e}_a\mathbf{e}_a^\top\Phi_{\text{Ctr}}^\top] = \Phi_{\text{Ctr}}\mathbb{E}_{p(a)}[\mathbf{e}_a\mathbf{e}_a^\top]\Phi_{\text{Ctr}}^\top = \Phi_{\text{Ctr}} D_A \Phi_{\text{Ctr}}^\top.$$

Note that we are working with the *uncentered* principal components, as is commmon for kernel PCA: we do not subtract the mean before computing the covariance. Since Σ is positive semidefinite, it can be diagonalized as

$$\Sigma = U\Lambda U^\top = \sum_i \lambda_i \mathbf{u}_i \mathbf{u}_i^\top$$

where $U = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k]$ is orthonormal and $\Lambda = \operatorname{diag}(\lambda_1, \lambda_2, \dots, \lambda_k)$ is a diagonal matrix of eigenvalues (here $k = |\mathcal{X}|$ is the dimension of the feature map). Each of the vectors $\mathbf{u}_i$ is one of the population principal components of the transformed distribution $\phi_{\text{Ctr}}(A)$, giving the directions of maximum variance, and the λ_i measure the variance in that direction.

Given a new augmentation $a \in \mathcal{A}$, we can then compute the projection of $\phi_{\text{Ctr}}(a)$ into each of these principal component directions as $\tilde{f}_i(a) = \mathbf{u}_i^\top \phi_{\text{Ctr}}(a)$. Since the λ_i measure the variance in each direction, we can optionally rescale these projection functions as

$$f_i(a) = \lambda_i^{-1/2}\mathbf{u}_i^\top \phi_{\text{Ctr}}(a),$$

so that $\mathbb{E}[f_i(a)^2] = 1$. (These are only well defined for $\lambda_i > 0$, but we are free to extend our set with additional functions to span the space of all functions $\mathcal{A} \to \mathbb{R}$.)

The Markov Chain. We now redirect our attention to the positive pair Markov chain. The Markov chain transition matrix is defined by $[P_{A\leftarrow A}]_{a_1,a_2} = p_+(a_1|a_2)$, or in matrix form

$$P_{A\leftarrow A} = P_{A,A}D_A^{-1}.$$

We are interested in the left eigenvectors $\mathbf{f}_i^\top P_{A\leftarrow A} = \lambda'_i \mathbf{f}_i^\top$ of this matrix $P_{A\leftarrow A}$, or equivalently the right eigenvectors of its transpose $P_{A\leftarrow A}^\top = P_{A\rightarrow A}$, given by $P_{A\rightarrow A}\mathbf{f}_i = \lambda'_i \mathbf{f}_i$. Observe that then

$$D_A^{-1}P_{A,A}\mathbf{f}_i = \lambda'_i \mathbf{f}_i$$

so equivalently

$$D_A^{-1/2}\left(D_A^{-1/2}P_{A,A}D_A^{-1/2}\right)D_A^{1/2}\mathbf{f}_i = \lambda'_i D_A^{-1/2}\left(D_A^{1/2}\mathbf{f}_i\right).$$

It follows that $D_A^{1/2}\mathbf{f}_i$ is an eigenvector of the symmetric matrix

$$M = D_A^{-1/2}P_{A,A}D_A^{-1/2}$$

with the same eigenvalue λ'_i, and so we can diagonalize M as $M = V\Lambda' V^\top$, where

$$V = \begin{bmatrix} D_A^{1/2}\mathbf{f}_1 & D_A^{1/2}\mathbf{f}_2 & \dots & D_A^{1/2}\mathbf{f}_k \end{bmatrix} = D_A^{1/2}\begin{bmatrix} \mathbf{f}_1 & \mathbf{f}_2 & \dots & \mathbf{f}_k \end{bmatrix} = D_A^{1/2}F$$

where

$$F = \begin{bmatrix} \mathbf{f}_1 & \mathbf{f}_2 & \dots & \mathbf{f}_k \end{bmatrix}.$$

(We note that the matrix M is exactly the symmetrized adjacency matrix described by HaoChen et al. (2021).)

Putting them together. We can now prove our main result from Section 3.

Theorem 3.1. *$f_i(a) = [\mathbf{f}_i]_a$ and $\lambda_i = \lambda'_i$ for all i and all $a \in \mathcal{A}$. Furthermore, given any function $g : \mathcal{A} \to \mathbb{R}$, if we let $c_i = E[g(a)f_i(a)]$, then the following are true:*

$$g(a) = \sum_i c_i f_i(a), \qquad \mathbb{E}[g(a)^2] = \sum_i c_i^2, \tag{3}$$

$$\mathbb{E}_{p_+}\left[\left(g(a_1) - g(a_2)\right)^2\right] = 2\sum_i (1 - \lambda_i) c_i^2. \tag{4}$$

Proof. For the first claim, consider the matrix $B = \Phi_{\text{Ctr}} D_A^{1/2}$, where $\Phi_{\text{Ctr}} = D_X^{1/2} P_{X\to A} D_A^{-1}$ described above. Take the singular value decomposition $B = U\Lambda^{1/2}V^\top$, where U and V are orthonormal and $\Lambda^{1/2}$ is diagonal. Now observe that

$$BB^\top = \Phi_{\text{Ctr}} D_A \Phi_{\text{Ctr}}^\top = \Sigma = U\Lambda U^\top,$$

and

$$\begin{aligned} B^\top B = D_A^{1/2}\Phi_{\text{Ctr}}^\top \Phi_{\text{Ctr}} D_A^{1/2} &= D_A^{1/2} K_{\text{Ctr}} D_A^{1/2} \\ &= D_A^{1/2}\left(D_A^{-1} P_{A,A} D_A^{-1}\right) D_A^{1/2} \\ &= D_A^{-1/2} P_{A,A} D_A^{-1/2} = M = V\Lambda V^\top. \end{aligned}$$

Thus, Σ and M must have the same eigenvalues.[3] For any i for which $\lambda_i > 0$, we then have

$$\begin{aligned} f_i(a) &= \lambda_i^{-1/2}\mathbf{u}_i^\top \phi_{\text{Ctr}}(a) \\ &= \mathbf{e}_i^\top \Lambda^{-1/2} U^\top \Phi_{\text{Ctr}} \mathbf{e}_a \\ &= \mathbf{e}_i^\top \Lambda^{-1/2} U^\top \left(B D_A^{-1/2}\right) \mathbf{e}_a \\ &= \mathbf{e}_i^\top \Lambda^{-1/2} U^\top \left(U\Lambda^{1/2} V^\top D_A^{-1/2}\right) \mathbf{e}_a \\ &= \mathbf{e}_i^\top V^\top D_A^{-1/2} \mathbf{e}_a \\ &= \mathbf{e}_i^\top \left(D_A^{-1/2} V\right)^\top \mathbf{e}_a \\ &= \mathbf{e}_i^\top F^\top \mathbf{e}_a \\ &= \mathbf{e}_i^\top \begin{bmatrix}\mathbf{f}_1 & \mathbf{f}_2 & \dots & \mathbf{f}_k\end{bmatrix}^\top \mathbf{e}_a = \mathbf{f}_i^T \mathbf{e}_a = [\mathbf{f}_i]_a. \end{aligned}$$

(When $\lambda_i = 0$, the normalized principal component projection functions are not uniquely defined, so we are free to simply set them equal to $[\mathbf{f}_i]_a$.)

Next, consider an arbitrary function $g : \mathcal{A} \to \mathbb{R}$, and let $\mathbf{g} \in \mathbb{R}^{\mathcal{A}}$ be its vector form, so that $g(a) = \mathbf{g}_a = \mathbf{g}^\top \mathbf{e}_a$. Also define $c_i = \mathbb{E}[g(a)f_i(a)]$ and $\mathbf{c} = \begin{bmatrix}c_1 & c_2 & \dots & c_k\end{bmatrix}^\top$. Then

$$\mathbf{c} = \mathbb{E}\left[g(a)F^\top \mathbf{e}_a\right] = \mathbb{E}\left[F^\top \mathbf{e}_a \mathbf{e}_a^T \mathbf{g}\right] = F^\top D_A \mathbf{g} = V^\top D_A^{1/2}\mathbf{g}$$

so we must have

$$\mathbf{g} = D_A^{-1/2}\left(V^\top\right)^{-1}\mathbf{c} = D_A^{-1/2} V\mathbf{c} = F\mathbf{c}$$

and thus $g(a) = \sum_i c_i f_i(a)$. Additionally, we see that

$$\begin{aligned} \mathbb{E}\left[g(a)^2\right] &= \mathbb{E}\left[\mathbf{g}^T \mathbf{e}_a \mathbf{e}_a^\top \mathbf{g}\right] = \mathbf{g}^\top D_A \mathbf{g} = \mathbf{c}^\top F^\top D_A F \mathbf{c} \\ &= \mathbf{c}^\top (D_A^{1/2}F)^\top (D_A^{1/2}F)\mathbf{c} = \mathbf{c}^\top V^\top V \mathbf{c} = \mathbf{c}^\top \mathbf{c} = \sum_i c_i^2. \end{aligned}$$

[3]If the eigenvector decompositions of Σ and M are not unique, we are free to select U and V such that they form a valid singular value decomposition for B.

Note that this also implies that the functions f_i are orthonormal under the base measure $p(a)$, e.g. $\mathbb{E}[f_i(a)^2] = 1$ and $\mathbb{E}[f_i(a)f_j(a)] = 0$ for $i \neq j$.

Finally, consider the quantity $\mathbb{E}_{p_+}\left[\left(g(a_1) - g(a_2)\right)^2\right]$. Observe that

$$\begin{aligned} E_{p_+}\left[\left(g(a_1) - g(a_2)\right)^2\right] &= E_{p_+}\left[g(a_1)^2 - 2g(a_1)g(a_2) + g(a_2)^2\right] \\ &= 2E\left[g(a)^2\right] - 2\mathbb{E}_{p_+}\left[g(a_1)g(a_2)\right] \\ &= 2\sum_i c_i^2 - 2\mathbb{E}_{p_+}\left[g(a_1)g(a_2)\right] \end{aligned}$$

Expanding the second term, we have

$$\begin{aligned} \mathbb{E}_{p_+}\left[g(a_1)g(a_2)\right] &= \mathbb{E}_{p_+}\left[\mathbf{g}^\top \mathbf{e}_{a_1}\mathbf{e}_{a_2}^\top \mathbf{g}\right] = \mathbf{g}^\top \mathbb{E}_{p_+}\left[\mathbf{e}_{a_1}\mathbf{e}_{a_2}^\top\right]\mathbf{g} \\ &= \mathbf{g}^\top P_{A,A}\mathbf{g} \\ &= \mathbf{g}^\top D_A^{1/2} M D_A^{1/2}\mathbf{g} \\ &= \mathbf{g}^\top D_A^{1/2} V\Lambda V^T D_A^{1/2}\mathbf{g} \\ &= \mathbf{c}^\top F^\top D_A^{1/2} V\Lambda V^T D_A^{1/2} F\mathbf{c} \\ &= \mathbf{c}^\top (D_A^{1/2}F)^\top V\Lambda V^T (D_A^{1/2}F)\mathbf{c} \\ &= \mathbf{c}^\top V^\top V\Lambda V^T V\mathbf{c} \\ &= \mathbf{c}^\top \Lambda \mathbf{c} = \sum_i \lambda_i c_i^2. \end{aligned}$$

We conclude that $E_{p_+}\left[\left(g(a_1) - g(a_2)\right)^2\right] = 2\sum_i (1-\lambda_i)c_i^2$. □

B.1. Relationship to the eigenvectors of the symmetrized adjacency matrix

Interestingly, the matrix M described above is exactly the symmetrized adjacency matrix discussed by HaoChen et al. (2021). HaoChen et al. motivate their loss as estimating the eigenvectors of M up to a scaling term by $p(a)^{1/2}$, due to prior work showing that eigenvalues give information about clustering structure in graphs.

The connection between the symmetrized adjacency matrix M and the positive-pair Markov chainis well known; indeed, HaoChen et al. briefly discuss the positive-pair Markov chain in their Section 2, and Levin and Peres (2017, Chapter 12) introduce the matrix M when discussing the spectral decomposition of a general symmetric Markov chain.

Another way of thinking about this reweighting is as a *change of measure*. The eigenvectors of M are orthonormal with respect to the counting measure over $\mathcal{A}$, e.g. if you sum squared values over all of $\mathcal{A}$, you obtain 1, and the dot product of different eigenvectors is zero. On the other hand, the eigenvectors $\mathbf{f}_i$ (or, equivalently, the eigenfunctions f_i) of the Markov chain are orthonormal with respect to the measure $p(A)$, e.g. if you take the expectation of squared values over random augmentations, you obtain 1, and the uncentered covariance of different eigenfunctions is zero. We believe that including the reweighting terms is a more natural perspective which gives additional insight into what contrastive losses are fundamentally doing.

We also note that our Assumption 1.1 is related to the probability-weighted Laplacian matrix of the augmentation graph, given by $L = D_A - P_{A,A}$. Indeed, we have

$$E_{p_+}\left[\left(g(a_1) - g(a_2)\right)^2\right] = 2\mathbf{g}^\top L\mathbf{g}.$$

B.2. Recovering proportionality constants

As described in Appendix A.1, minimizing the NT-XEnt / InfoNCE loss may not exactly produce the contrastive kernel K_{Ctr}, but may instead learn a scaled version

$$\widehat{K}_*(a_1, a_2) = \left\langle \sqrt{z(a_1)} \cdot \phi_{\text{Ctr}}(a_1),\ \sqrt{z(a_2)} \cdot \phi_{\text{Ctr}}(a_2) \right\rangle = \sqrt{z(a_1)}\sqrt{z(a_2)} K_{\text{Ctr}}(a_1, a_2),$$

where $z : \mathcal{A} \to \mathbb{R}^+$ is some function which is constant on each communicating class on the Markov chain. (Since $K_{\text{Ctr}}(a_1, a_2) = 0$ whenever a_1 and a_2 are in separate communicating classes, we could equivalently say $\widehat{K}_*(a_1, a_2) = z(a_1)K_{\text{Ctr}}(a_1, a_2) = z(a_2)K_{\text{Ctr}}(a_1, a_2)$.)

When the Markov chain has one communicating class, $\widehat{K}_*$ is simply a scaled version of K_{Ctr}. In this case, all of the principal component projection functions for $\widehat{K}_*$ are still the eigenfunctions of the Markov chain, but the eigenvalues may be scaled by that constant. The true eigenvalues of the eigenfunctions can then be estimated using equation Equation 5, which states that $\mathbb{E}[(f_i(a_1) - f_i(a_2))^2] = 2(1 - \lambda_i)$.

When the Markov chain has multiple communicating classes, we can partition the eigenfunctions so that each eigenfunction is nonzero on a single communicating class. Since the scaling function z acts as a scaling factor for each communicating class, the principal component functions will then be scaled copies of these partitioned eigenfunctions. We can then similarly estimate the true eigenvalues for each of these eigenfunctions using Equation 5.

C. Generalization properties of the eigenfunction representation

C.1. Min-max optimality of eigenfunctions

We now prove the min-max optimality of the eigenfunctions with respect to their L2 norm.

Proposition 4.1. *Let $\mathcal{F}_r = \{a \mapsto \beta^\top r(a) : \beta \in \mathbb{R}^d\}$ be the family of linear predictors from representation r, and S_ϵ be the set of functions satisfying Assumption 1.1. Let $r_*^d(a) = [f_1(a), f_2(a), \dots, f_d(a)]$ be the representation consisting of the d largest eigenfunctions of the positive pair Markov chain. Then $\mathcal{F}_{r_*^d}$ maximizes the invariance of the least-invariant unit-norm predictor in $\mathcal{F}_{r_*^d}$:*

$$\mathcal{F}_{r_*^d} = \underset{\mathcal{F} \text{ rank } d}{\operatorname{argmin}} \max_{\substack{\hat{g} \in \mathcal{F}, \\ \mathbb{E}[\hat{g}(a)^2] = 1}} \mathbb{E}_{p_+}\left[\left(\hat{g}(a_1) - \hat{g}(a_2)\right)^2\right]. \tag{6}$$

Simultaneously, $\mathcal{F}_{r_*^d}$ minimizes the least-squares approximation error for the worst-case target function in S_ϵ:

$$\mathcal{F}_{r_*^d} = \underset{\mathcal{F} \text{ rank } d}{\operatorname{argmin}} \max_{g \in S_\epsilon} \min_{\hat{g} \in \mathcal{F}} \mathbb{E}_{p(a)}\left[\left(g(a) - \hat{g}(a)\right)^2\right]. \tag{7}$$

Proof. We will start by deriving Equation 7, and derive Equation 6 afterward. We can think of Equation 7 as equivalent to the following adversarial game:

1. Player chooses a dimension-d subspace $\mathcal{F} \subset \mathcal{A} \to \mathbb{R}$ of functions.
2. Adversary chooses a function $g \in \mathcal{A} \to \mathbb{R}$ with a fixed level of invariance $\mathbb{E}[(g(a_1) - g(a_2))^2] = 2g^T(D_A - P_{A,A})g = \epsilon$. Without loss of generality, we let $\epsilon = 2$ so that $g^T(D_A - P_{A,A})g = 1$; other values of ϵ will just lead to scaling the function g.
3. Player chooses the best $\hat{g} \in \mathcal{F}$ to minimize $\mathbb{E}[(\hat{g}(a) - g(a))^2]$

We can analyze this game by working backward from the innermost step, step 3. Given the function class $\mathcal{F}$ and adversarially chosen target function g, choosing $\hat{g}$ to minimize the expected squared error is equivalent to finding the orthogonal projection of g into $\mathcal{F}$ with respect to the measure $p(A)$, e.g. with respect to the weighted L2 norm $\mathcal{L}(2; p(A))$. More precisely, we want

$$\begin{aligned}\hat{g} &= \underset{\hat{g} \in \mathcal{F}}{\operatorname{argmin}}\, \mathbb{E}[(\hat{g}(a) - g(a))^2] = \underset{\hat{g} \in \mathcal{F}}{\operatorname{argmin}} (\hat{\mathbf{g}} - \mathbf{g})^T D_A (\hat{\mathbf{g}} - \mathbf{g}) \\ &= \underset{\hat{g} \in \mathcal{F}}{\operatorname{argmin}} (D_A^{1/2}\hat{\mathbf{g}} - D_A^{1/2}\mathbf{g})^T (D_A^{1/2}\hat{\mathbf{g}} - D_A^{1/2}\mathbf{g})\end{aligned}$$

But this is just finding the $\hat{\mathbf{g}} \in \mathcal{F}$ which minimizes $\|D_A^{1/2}\hat{\mathbf{g}} - D_A^{1/2}\mathbf{g}\|_2$. This is given by the orthogonal projection of the vector $D_A^{1/2}\mathbf{g}$ into $D_A^{1/2}\mathcal{F}$, under the ordinary L2 norm.

We can now define R as the orthogonal projection operator on $D_A^{1/2}\mathcal{F}$, such that $R\mathbf{h} \in D_A^{1/2}\mathcal{F}$ (e.g. $D_A^{-1/2}R\mathbf{h} \in \mathcal{F}$), and for $D_A^{1/2}\mathbf{f} \in D_A^{1/2}\mathcal{F}$ (e.g. $\mathbf{f} \in \mathcal{F}$), we have $D_A^{1/2}\mathbf{f} = RD_A^{1/2}\mathbf{f}$ (e.g. $\mathbf{f} = D_A^{-1/2}RD_A^{1/2}\mathbf{f}$). Observe that R is real and symmetric, and has eigenvalue 1 with multiplicity d and all other eigenvalues are 0. Since R characterizes the subset, we will find it convenient to redefine our objective for the initial player as choosing R, and then letting $\mathcal{F} = \{D_A^{-1/2} R D_A^{1/2} g : g \in \mathcal{A} \to \mathbb{R}\}$.

We then have

$$\hat{\mathbf{g}} = \operatorname*{argmin}_{\hat{\mathbf{g}} \in \mathcal{F}} \mathbb{E}[(\hat{g}(v) - g(v))^2] = D_A^{-1/2} R D_A^{1/2} \mathbf{g}.$$

and the cost is

$$\begin{aligned}
\mathbb{E}[(\hat{g}(v) - g(v))^2] &= (D_A^{1/2} f - D_A^{1/2}\mathbf{g})^T (D_A^{1/2} f - D_A^{1/2}\mathbf{g}) \\
&= (R D_A^{1/2}\mathbf{g} - D_A^{1/2}\mathbf{g})^T (R D_A^{1/2}\mathbf{g} - D_A^{1/2}\mathbf{g}) \\
&= (D_A^{1/2}\mathbf{g})^T (R - I)^T (R - I)(D_A^{1/2}\mathbf{g}) \\
&= \mathbf{g}^T D_A^{1/2} (R^T R - 2R + I) D_A^{1/2} \mathbf{g} \\
&= \mathbf{g}^T D_A^{1/2} (I - R) D_A^{1/2} \mathbf{g}.
\end{aligned}$$

We next consider step 2. Given R, what g should the adversary pick? Letting $L = D_A - P_{A,A}$, the adversary is constrained to pick $\mathbf{g}$ such that $\mathbf{g}^T L \mathbf{g} = (L^{1/2}\mathbf{g})^T (L^{1/2}\mathbf{g}) = 1$. We note that L is not full rank: in particular, any eigenvector of M with eigenvalue 1 is an eigenvector of L of eigenvector zero. Any function $\mathbf{g}$ chosen by the adversary must then be the sum of two parts:

- a component in in the range of L, of the form $\left(L^\dagger\right)^{1/2} \mathbf{u}$ where $\|\mathbf{u}\|_2 = 1$ and $\dagger$ represents the Moore-Penrose pseudoinverse,
- and a component in the null space of L.

Overall, we can thus write $\mathbf{g} = \left(L^\dagger\right)^{1/2} \mathbf{u} + \mathbf{h}$ where $\|\mathbf{u}\|_2 = 1$ and $\mathbf{h}^\top L \mathbf{h} = 0$. Similarly, the response $\hat{\mathbf{g}}$ must also have two components, one in the range of L and one in the null space of L. There are then two cases. If $\mathcal{F}$ does not span the entire null space of L, the adversary can force an arbitrarily high approximation error by choosing $\mathbf{h}$ to be in the null space of L but not $\mathcal{F}$. On the other hand, if $\mathcal{F}$ spans the entire null space of L, the player can always perfectly approximate $\mathbf{h}$, and so the adversary is forced to maximize cost by using $\mathbf{u}$. In particular, they will pick

$$\begin{aligned}
\mathbf{u} &= \operatorname*{argmax}_{\|\mathbf{u}\|_2 = 1} (\left(L^\dagger\right)^{1/2} \mathbf{u})^T D_A^{1/2} (I - R) D_A^{1/2} (\left(L^\dagger\right)^{1/2} \mathbf{u}) \\
&= \operatorname*{argmax}_{\|\mathbf{u}\|_2 = 1} \mathbf{u}^T \left(L^\dagger\right)^{1/2} D_A^{1/2} (I - R) D_A^{1/2} \left(L^\dagger\right)^{1/2} \mathbf{u} \\
&= \operatorname*{argmax}_{\|\mathbf{u}\|_2 = 1} \mathbf{u}^T A \mathbf{u}
\end{aligned}$$

where A is the matrix $\left(L^\dagger\right)^{1/2} D_A^{1/2} (I - R) D_A^{1/2} \left(L^\dagger\right)^{1/2}$. The optimal choice for $\mathbf{u}$ is an eigenvector of A with maximal eigenvalue, and the cost is then that maximal eigenvalue. But observe that M is similar to the following:

$$\begin{aligned}
A &\sim (\left(L^\dagger\right)^{1/2} D_A^{1/2})^{-1} M (\left(L^\dagger\right)^{1/2} D_A^{1/2}) \\
&= (I - R) D_A^{1/2} L^\dagger D_A^{1/2} \\
&= (I - R) \left(D_A^{-1/2} L D_A^{-1/2}\right)^\dagger := A'
\end{aligned}$$

Similar matrices have the same eigenvalues, so the maximum cost attainable by the adversary is the maximal eigenvalue of A'.

Finally, we consider step 1. Which R should our player choose to minimize this maximum cost? They should first ensure the cost is finite, by choosing R to span the null space of $D_A^{-1/2} L D_A^{-1/2}$. (Note that if d is less than the dimension of this null space, there is no choice that ensures a finite cost; in this case every representation has unbounded worst-case approximation error.) Afterward, they should ensure that A' has the smallest maximum eigenvalue. The sorted vector of eigenvalues of A' is bounded below by the vector obtained by matching the largest eigenvalues of $\left(D_A^{-1/2} L D_A^{-1/2}\right)^\dagger$ with the smallest of $(I - R)$ (e.g. the largest of R) (Bhatia, 2013, exercise III.6.14)[4]. Let d^* be the dimension of the null space of $D_A^{-1/2} L D_A^{-1/2}$. Then $I - R$ has d zero eigenvalues, of which d^* are used to span this null space, and the remaining $d - d^*$ (if any) are used to reduce the eigenvalues of A'. Thus, the largest eigenvalue of A' is always at least as big as the $(d - d^* + 1)$-th largest eigenvalue of $\left(D_A^{-1/2} L D_A^{-1/2}\right)^\dagger$. We can attain this bound by setting R to exactly capture the $(d - d^*)$-dimensional subspace of $\left(D_A^{-1/2} L D_A^{-1/2}\right)^\dagger$ spanned by its top eigenspaces, along with the d^*-dimensional null space of $D_A^{-1/2} L D_A^{-1/2}$.

But the combination of the null space of $D_A^{-1/2} L D_A^{-1/2}$ and the top eigenspace of $\left(D_A^{-1/2} L D_A^{-1/2}\right)^\dagger$ is just the space spanned by the d eigenvectors of $D_A^{-1/2} L D_A^{-1/2}$ with the *smallest* eigenvalues. Furthermore,

$$\begin{aligned} D_A^{-1/2} L D_A^{-1/2} &= D_A^{-1/2}(D_A - P_{A,A}) D_A^{-1/2} \\ &= I - D_A^{-1/2} P_{A,A} D_A^{-1/2} = I - M, \end{aligned}$$

so we are looking for the eigenvectors of M with the *largest* eigenvalues, where M is the matrix described in Appendix B.

Thus, the player should choose $\mathcal{F}$ such that $D_A^{1/2}\mathcal{F}$ spans the top d-dimensional eigenspace of M, e.g. they should choose functions of the form $D_A^{-1/2}\mathbf{v}_i$ where the $\mathbf{v}_i$ are the eigenvectors of M with largest eigenvalue. But these are exactly the left eigenvectors $\mathbf{f}_i$ of the positive pair Markov chain, which is how r_*^d is defined. We conclude that $\mathcal{F}_{r_*^d}$ is the optimal choice for the player, and thus Equation 7 holds.

Indeed, we can conclude something further: if λ_{d+1} is the $(d+1)$th eigenvalue of the positive pair Markov chain (the variance along the $(d+1)$th principal component of the contrastive kernel), then as long as $d \geq d^*$, $(1 - \lambda_{d+1})^{-1}$ is the $(d - d^* + 1)$th eigenvalue of $\left(D_A^{-1/2} L D_A^{-1/2}\right)^\dagger$, which is exactly the worst-case approximation error for $\mathcal{F}_{r_*^d}$ against any function with $\mathbb{E}[(g(v_1) - g(v_2))^2] = 2g^T L g = 2$. Scaling by ε, if $\mathbb{E}[(g(v_1) - g(v_2))^2] = \varepsilon$ then the worst case error is $\frac{1}{2}\varepsilon/(1 - \lambda_{d+1})$. In other words,

$$\max_{g \in S_\epsilon} \min_{\hat{g} \in \mathcal{F}_{r_*^d}} \mathbb{E}_{p(a)}\left[\left(g(a) - \hat{g}(a)\right)^2\right] = \frac{\varepsilon}{2(1 - \lambda_{d+1})}.$$

We now return our attention to Equation 6. This equation can also be formulated as an adversarial game:

1. Player chooses a rank-d subspace $\mathcal{F} \subset \mathcal{A} \to \mathbb{R}$ of functions.
2. Adversary chooses a function $\hat{g} \in \mathcal{F}$ with unit norm $\mathbb{E}[\hat{g}(v)^2] = \hat{\mathbf{g}}^T D_A \hat{\mathbf{g}} = 1$ to maximize $\mathbb{E}[(\hat{g}(v_1) - \hat{g}(v_2))^2] = 2\hat{\mathbf{g}}^T L \hat{\mathbf{g}}$.

We can again identify the choice of $\mathcal{F}$ with the choice of the orthogonal projection matrix R on $D_A^{1/2}\mathcal{F}$. We know $\hat{g} \in \mathcal{F}$, so we can write $\hat{\mathbf{g}} = D_A^{-1/2} R D_A^{1/2} \hat{\mathbf{g}}$. Also note that for any $\mathbf{h}$ (not even necessarily in $\mathcal{F}$), $D_A^{-1/2} R D_A^{1/2}\mathbf{h} \in \mathcal{F}$. Now suppose we choose an $\mathbf{h}$ so that $\mathbb{E}[h(a)^2] = \mathbf{h}^T D_A \mathbf{h} = 1$, and define $\hat{\mathbf{g}} = D_A^{-1/2} R D_A^{1/2}\mathbf{h}$. Then

$$\begin{aligned} \hat{\mathbf{g}}^T D_A \hat{\mathbf{g}} &= \mathbf{h}^T D_A^{1/2} R D_A^{-1/2} D_A (D_A^{-1/2} R D_A^{1/2} \mathbf{h}) \\ &= \mathbf{h}^T D_A^{1/2} R^2 D_A^{1/2} \mathbf{h} \\ &= \mathbf{h}^T D_A^{1/2} R D_A^{1/2} \mathbf{h} \\ &\leq \mathbf{h}^T D_A^{1/2} I D_A^{1/2} \mathbf{h} = 1 \end{aligned}$$

because R has eigenvalues at most 1. So, the following are equivalent:

[4] See also https://math.stackexchange.com/questions/573583/eigenvalues-of-the-product-of-two-symmetric-matrices

- choosing $\hat{\mathbf{g}} \in \mathcal{F}$ with $\mathbb{E}[\hat{g}(v)^2] \leq 1$
- choosing $\mathbf{h} \in \mathbb{R}^{\mathcal{A}}$ with $\mathbb{E}[h(v)^2] \leq 1$ and letting $\hat{\mathbf{g}} = D_A^{-1/2} R D_A^{1/2} \mathbf{h}$

We also note that there is no advantage to the adversary from picking a function such that $\mathbb{E}[\hat{g}(v)^2] < 1$. So we can reframe step 2 as choosing $\mathbf{h}$ so that $\mathbb{E}[h(v)^2] = 1$, to maximize

$$2\left(D_A^{-1/2} R D_A^{1/2} \mathbf{h}\right)^T L \left(D_A^{-1/2} R D_A^{1/2} \mathbf{h}\right)$$

We can further reparameterize by letting $\mathbf{h} = D_A^{-1/2} \mathbf{u}$, so that $\mathbb{E}[h(v)^2] = 1$ is equivalent to $\|\mathbf{u}\|_2^2 = 1$. We then have cost

$$\begin{aligned} C = 2\hat{\mathbf{g}}^T L \hat{\mathbf{g}} &= 2(\mathbf{h}^T D_A^{1/2} R D_A^{-1/2}) L (D_A^{-1/2} R D_A^{1/2} \mathbf{h}) \\ &= 2(\mathbf{u}^T D_A^{-1/2}) D_A^{1/2} R D_A^{-1/2} L D_A^{-1/2} R D_A^{1/2} (D_A^{-1/2} \mathbf{u}) \\ &= 2\mathbf{u}^T R D_A^{-1/2} L D_A^{-1/2} R \mathbf{u} \end{aligned}$$

The choice that maximizes the cost is then an eigenvector of

$$B = 2 R D_A^{-1/2} L D_A^{-1/2} R = \left(L^{1/2} D_A^{-1/2} R\right)^\top \left(L^{1/2} D_A^{-1/2} R\right)$$

with maximal eigenvalue, and the cost is 2 times the maximum eigenvalue of B. But note that B has the same eigenvalues as

$$B' = \left(L^{1/2} D_A^{-1/2} R\right) \left(L^{1/2} D_A^{-1/2} R\right)^\top = L^{1/2} D_A^{-1/2} R D_A^{-1/2} L^{1/2}$$

since $R^2 = R$. And B' is similar to

$$B'' = D_A^{-1/2} L D_A^{-1/2} R$$

so B and B'' have the same eigenvalues.

We now consider step 1. What should the player choose for R? By a similar eigenvalue-of-product argument as used for Equation 7, regardless of the choice of R the largest eigenvalue of B'' must always be at least as big as the dth smallest eigenvalue of $D_A^{-1/2} L D_A^{-1/2}$, because R has eigenvalue 1 with multiplicity d. We can attain this minimum cost by choosing R to project into the eigenspace spanned by the d eigenvectors of $D_A^{-1/2} L D_A^{-1/2}$ with the *smallest* eigenvalues.

But observe that the smallest eigenvalues and corresponding eigenvectors of $D_A^{-1/2} L D_A^{-1/2} = I - M$ are exactly the largest eigenvalues and corresponding eigenvectors of $M = D_A^{-1/2} P_{A,A} D_A^{-1/2} = I - L$, which as we argued above, is exactly the set of eigenfunctions f_i used to construct r_*^d.

In this case, the optimal cost itself is determined by the largest eigenvalue of $D_A^{-1/2} L D_A^{-1/2}$ (times two), so we obtain

$$\max_{\substack{\hat{g} \in \mathcal{F}_{r_*^d}, \\ \mathbb{E}[\hat{g}(a)^2] = 1}} \mathbb{E}_{p_+}\left[\left(\hat{g}(a_1) - \hat{g}(a_2)\right)^2\right] = 2(1 - \lambda_d).$$

□

C.2. Generalization bound for linear prediction with the eigenfunction representation

Proposition 4.2. *Let $(A_i, Y_i)_{i=1}^n$ be i.i.d. samples, choose $R \geq 0$, and consider the constrained empirical risk minimizer $\hat{\beta}_R \in \operatorname{argmin}_{\|\beta\|_2 \leq R} n^{-1} \sum_{i=1}^n |\langle \beta, r_d^*(A_i) \rangle - Y_i|$. Then the expected excess risk of $\hat{\beta}_R$ is bounded by:*

$$\mathbb{E}\left[\mathcal{E}(\hat{\beta}_R)\right] \leq \frac{2dR}{\sqrt{n}} + \sqrt{d}(\|\beta^*\|_2 - R)_+ + \sqrt{\frac{\varepsilon}{2(1 - \lambda_{d+1})}}$$

where $\mathcal{E}(\beta) := \mathcal{R}(\beta) - \mathcal{R}^*$ is the excess risk and $(x)_+ := \max\{x, 0\}$.

Proof. We start by decomposing the excess risk as:

$$\mathcal{R}(\hat{\beta}_R) - \mathcal{R}^* = \left(\mathcal{R}(\hat{\beta}_R) - \inf_{\|\beta\|_2 \le R} \mathcal{R}(\beta)\right) + \left(\inf_{\|\beta\|_2 \le R} \mathcal{R}(\beta) - \mathcal{R}(\beta^*)\right) + \left(\mathcal{R}(\beta^*) - \mathcal{R}^*\right)$$

The first term is the estimation error, which we can readily bound by first noting that:

$$\mathbb{E}\big[\|r_*^d(A)\|_2^2\big] = \mathbb{E}\left[\sum_{i=1}^d f_i(A)^2\right] = \sum_{i=1}^d \mathbb{E}\big[f_i(A)^2\big] = d$$

then noticing that our loss is 1-Lipschitz, and finishing with the standard Rademacher complexity argument for constrained linear classes (Kakade et al., 2008) to get:

$$\left(\mathcal{R}(\hat{\beta}_D) - \inf_{\|\beta\|_2 \le D} \mathcal{R}(\beta)\right) \le \frac{2dR}{\sqrt{n}} \tag{8}$$

The second term is an approximation error term due to the use of a constrained linear class instead of the full linear class. Define $\tilde{\beta}^* := \frac{\beta^*}{\max\{\|\beta^*\|_2/R, 1\}}$. Then we can bound this second term by:

$$\inf_{\|\beta\|_2 \le R} \mathcal{R}(\beta) - \mathcal{R}(\beta^*) \le \mathcal{R}(\tilde{\beta}^*) - \mathcal{R}(\beta^*) \le \mathbb{E}\left[\left|\left\langle r_*^d(A), \tilde{\beta}^* - \beta^*\right\rangle\right|\right] \le \mathbb{E}\big[\|r_*^d(A)\|_2\big]\,\|\tilde{\beta}^* - \beta^*\|_2 \le \sqrt{d}(\|\beta^*\|_2 - R)_+ \tag{9}$$

where the second inequality follows from the 1-Lipschitznes of the loss, the third by Cauchy-Schwartz inequality, and the last by Jensen's inequality.

The third and last term is an approximation error term due to the use of a function class which is contained in the span of the first d eigenfunctions $(f_i)_{i=1}^d$. We can bound it as follows:

$$\mathcal{R}(\beta^*) - \mathcal{R}^* \le \mathbb{E}\big[\big|\langle r_*^d(A), \beta^*\rangle - g^*(A)\big|\big] \le \sqrt{\mathbb{E}[(\langle r_*^d(A), \beta^*\rangle - g^*(A))^2]} \le \sqrt{\frac{\varepsilon}{2(1-\lambda_{d+1})}} \tag{10}$$

where the first inequality follows from the 1-Lipschitznes of the loss, the second from Jensen's inequality, and the last by first noticing that the function $h(a) := \langle r_*^d(a), \beta^*\rangle$ satisfies $h = \operatorname{argmin}_{g \in \mathcal{F}_{r_*^d}} \mathbb{E}\big[(g(A) - g^*(A))^2\big]$ (since it is the projection of g^* onto $\mathcal{F}_{r_*^d}$ under the norm $\|x\|^2 := \mathbb{E}\big[x^2(A)\big]$), then appealing to the proof of Proposition 4.1. Combining the bounds of equations (8), (9), and (10), we obtain the stated generalization bound. □

D. Experimental details

D.1. Task: MNIST with Multinomial Augmentations

Our goal in designing our task was to construct distributions $p(X)$ and $p(A|X)$ such that the exact contrastive kernel could be computed, and so that the Markov chain would mix between different unperturbed dataset examples $x \in \mathcal{X}$ without changing the label too frequently.

To this end, we constructed our task as follows:

- Define $\mathcal{X}$ to be a particular subset of the MNIST dataset, and choose $p(X)$ to be the uniform distribution over $\mathcal{X}$. We consider two choices for $\mathcal{X}$: randomly selecting 512 images from each of the ten digit classes (used for comparisions between models), and randomly selecting 1024 images from the digits 0, 1, and 2 (used for visualizations of the eigenfunctions).
- For each image, generate 64 pertubed copies, by randomly blurring, translating, and rotating digits by a small amount. Add a small constant to each pixel so that no pixel has value zero, then normalize each such copy so that its pixel values sum to 1.
- To generate an augmentation of an image $x \in \mathcal{X}$ according to $p(A|X = x)$, first choose one of the 64 copies of x, then sample a set of k pixel locations with replacement from the distribution represented by that copy, where k determines the augmentation strength. This means we are more likely to sample pixel locations which were within the original digit, although due to the perturbations described above the pixels may be scattered around the digit.

Our set $\mathcal{A}$ is thus the set of all 28×28 images for which all pixels have a nonnegative integer value, and the total of all pixels is k. (Due to the low pixel density, to improve visibility in our figures we render each pixel as a box 5x its original size, with shading indicating overlap of these boxes. However, when giving input to the model, we directly input this sparse pixel reprsentation, without the 5x multiplier.)

Given a particular augmented example a, we can compute $p(a|x)$ for any $x \in \mathcal{X}$ by summing over each of the 64 copies of x and using the closed-form PDF of a multinomial distribution. We can then compute $p(x|a)$ by normalizing over all possible x, and use this to exactly compute the contrastive kernel feature map ϕ_{Ctr}. We selected the perturbation parameters such that there was nontrivial uncertainty in x given each a; in other words, we made sure the positive pair Markov chain mixed well between examples.

D.2. Ten-class MNIST models

For our main experimental results on all ten MNIST digit classes, we used three-block variants of a ResNet-18 model followed by a two 128-dimensional fully-connected layers and a final output layer.

- Linear kernel: We used kernel parameterization $\widehat{K}_\theta(a_1, a_2) = h_\theta(a_1)^T h_\theta(a_2)$ and the spectral loss, with h_θ as the output of the final layer. We set the dimension of the final layer to 512.
- Temperature-normalized exponential kernel: We used kernel parameterization $\exp(h_\theta(a_1)^T h_\theta(a_2)/\tau)$, where h_θ is computed by normalizing the output of the final layer to lie on the unit hypersphere. We set the dimension of the final layer to 32. For the loss function, we used a weighted combination of 0.9 times the NT-XEnt loss and 0.1 times the NT-Logistic loss. (Including the NT-Logistic loss ensures that there is a unique function that minimizes the total loss, without requiring us to estimate proportionality constants.) We optimized the temperature τ jointly with the model parameters.
- Temperature-normalized exponential kernel with scale: We used kernel parameterization $\exp(h_\theta(a_1)^T h_\theta(a_2)/\tau) \cdot s_\theta(a_1)s_\theta(a_2)$. We set the dimension of the final layer to 33, and defined h_θ by taking the first 32 entries and normalizing them to lie on the unit hypersphere. We then defined s_θ to be $\exp(5 \times \tanh(x))$ where x is the 33rd entry of the final layer. We again optimized the temperature τjointly with the model parameters, using a weighted combination of 0.9 times the NT-XEnt loss and 0.1 times the NT-Logistic loss. The scale parameter allows the model to adjust the magnitude of the kernel, which can be used to scale the eigenvalues of the principal components or to correct for differences in likelihood between points.

We trained our models using the Adam optimizer over 50,000 training iterations and a batch size of 4096 positive pairs per iteration, implemented using the JAX and FLAX libraries (Bradbury et al., 2018; Heek et al., 2020). We used batch normalization for the first 35,000 iterations, then interpolated between the current batch statistics and the average from previous batches for 2,000 more iterations, and finally trained for 13,000 iterations using frozen batch norm statistics alone (e.g. in "inference" mode); we found that this increased the quality of the extracted principal components.

D.3. Three-class MNIST rational quadratic model

Figures 1 and 3, we trained a ResNet-18 model on only the MNIST digits 0, 1, and 2, using a scaled two-dimensional rational quadratic kernel:

$$\widehat{K}_\theta(a_1, a_2) = s_\theta(a_1)s_\theta(a_2)\left(1 - \frac{\|f_\theta(a_1) - f_\theta(a_2)\|^2}{2\alpha}\right)^{-\alpha}. \tag{11}$$

Here $f_\theta : \mathcal{A} \to \mathbb{R}^2$ embeds inputs into the plane, $s_\theta : \mathcal{A} \to \mathbb{R}^+$ is a learned scale function, and α is a learned shape parameter. We set the output dimension of the ResNet-18 model to 3, and took the first two elements as f_θ; s_θ was defined as $\exp(5 \times \tanh(x))$ where x is the third element. We also parameterize $\alpha = \exp(\gamma)$ and learn the scalar parameter γ. The model has a base hidden dimension of 128. The model was trained for 50,000 training iterations. We used the cross entropy InfoNCE loss (as described in Appendix A.1) and the Adam optimizer, with a batch size of 4096 positive pairs per iteration.

D.4. Extraction and analysis of principal components

To estimate the eigenfunctions of the true kernel, we performed PCA using the explicit form ϕ_{Ctr} of the contrastive kernel feature map, where the population covariance was approximated by averaging over 256 augmentations for each of the images in $\mathcal{X}$. We then constructed the principal component projection functions using that covariance.

For our approximate kernels $\widehat{K}_\theta$, we first constructed an approximation of the feature map for $\widehat{K}_\theta$ using the Nyström method (Williams and Seeger, 2000): we sampled a subset S of augmentations by randomly selecting 25% of $\mathcal{X}$ and sampling one augmentation for each image, computed the Gram matrix $\widehat{K}_\theta(S, S)$ for that subset, then set

$$\hat{\phi}(a) = \widehat{K}_\theta(S, S)^{-1/2}\widehat{K}_\theta(S, a)$$

where $\widehat{K}_\theta(S, a)$ is the vector of similarities of a to each reference augmentation in S. The result is a feature map such that $\hat{\phi}(a_1)^\top\hat{\phi}(a_2) \approx \widehat{K}_\theta(a_1, a_2)$. We then again performed PCA using this feature map, using 256 samples per example in $\mathcal{X}$ to compute the covariance.

We normalized all principal component projection functions to have unit uncentered variance, e.g. $\mathbb{E}[f_i(a)^2] = 1$ and $\mathbb{E}[\hat{f}_i(a)^2] = 1$. We then computed alignments by taking the squared covariance $\mathbb{E}[f_i(a)\hat{f}_j(a)]^2$. Note that by Theorem 3.1 this is equivalent to the square of the coefficient c_i for the function $\hat{f}_j$ expanded in terms of the basis of eigenfunctions f_i; consequently we have $\sum_i \mathbb{E}[f_i(a)\hat{f}_j(a)]^2 = \mathbb{E}[\hat{f}_j(a)^2] = 1$. (Note that the sign of a principal component is not uniquely determined, so squaring the covariance ensures that f and $-f$ are considered to be aligned with each other.)

We estimated the positive-pair discrepancy for each principal component function by taking the sample average of $\big(f_i(a_1) - f_i(a_2)\big)^2$ over 16 randomly sampled augmentation pairs for each image in $\mathcal{X}$.

D.5. Downstream supervised learning

To generate our supervised training and validation sets, we sampled one augmentation of 16 random images from each digit class, labeled with the original label, a total of 160 labeled augmentations in each set. For the test set, we took 170 distinct images from each digit class and generated one augmentation from each image, without overlap with the training or validation sets, for a total of 1700 labeled augmentations.

For the classification task, we fit a logistic regression classifier on the training set using SciKit Learn. For PCA representations, we swept over 40 logarithmically-spaced L2 regularization strengths from 10^{-4} to 10^{1}, and also swept over representation dimension d, taking the first d principal components for d between 1 and 256. For ResNet embedding representations, we swept over 150 logarithmically-spaced L2 regularization strengths from 10^{-4} to 10^{10}; we found that higher regularization strengths were necessary to attain a good solution. We selected the hyperparameters based on which setting gave the highest top-1 accuracy on the validation set.

For the regression task, we used a centered version of the one-hot indicator vector, e.g. the target vector for an example from digit 2 was

$$[-0.1, -0.1, 0.9, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1].$$

The purpose of this centering was to ensure that the expected value of the label vector was the zero vector. We then fit a predictor using ridge regression (L2-regularized least-squares regression) in Numpy. As for the classification task, for PCA representations we swept over 40 logarithmically-spaced L2 regularization strengths from 10^{-4} to 10^{1} and over each representation dimension d between 1 and 256, and for ResNet embedding representations we swept over 150 L2 regularization strengths from 10^{-4} to 10^{10}. We selected the hyperparameters based on which setting gave the lowest squared error on the validation set.

D.6. Alignment and discrepancy relationships across regularization strengths

Figure 4 in the main paper shows the alignment and discrepancy relationships for the ten-digit MNIST task for $k = 20$ sampled pixels per digit. In this section we show the corresponding plots for $k = 10$ (stronger augmentations) and $k = 50$ (weaker augmentations.)

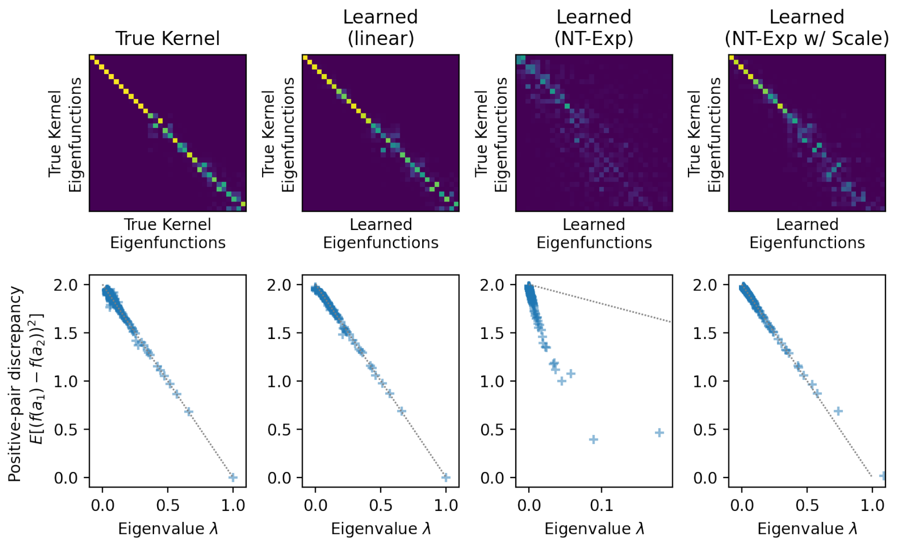

Figure 5. Alignments and discrepancy relationships for the MNIST task with $k = 10$. Stronger augmentations lead to larger gaps between eigenvalues, and the corresponding principal component functions can be estimated quite accurately from learned kernels.

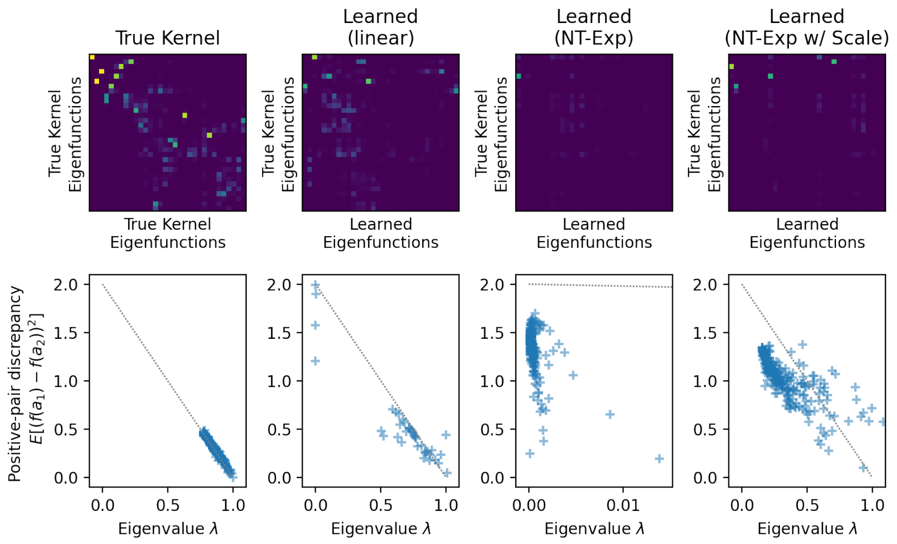

Figure 6. Alignments and discrepancy relationships for the MNIST task with $k = 50$. Weaker augmentations lead to more eigenvalues close to 1, and make it harder to estimate the principal components, even between two runs of Kernel PCA for the true contrastive kernel. The linear kernel model converges to a low-rank solution, leading to a lower eigenvalue density that still obeys the predicted discrepancy relationship, whereas the NT-Exp model deviates from that relationship without showing a clear pattern.

E. Additional visualizations of three-digit MNIST task

In this section, we provide a few additional visualizations of the three-digit variant of the MNIST task, the contrastive kernel for that task, and our learned rational quadratic model.

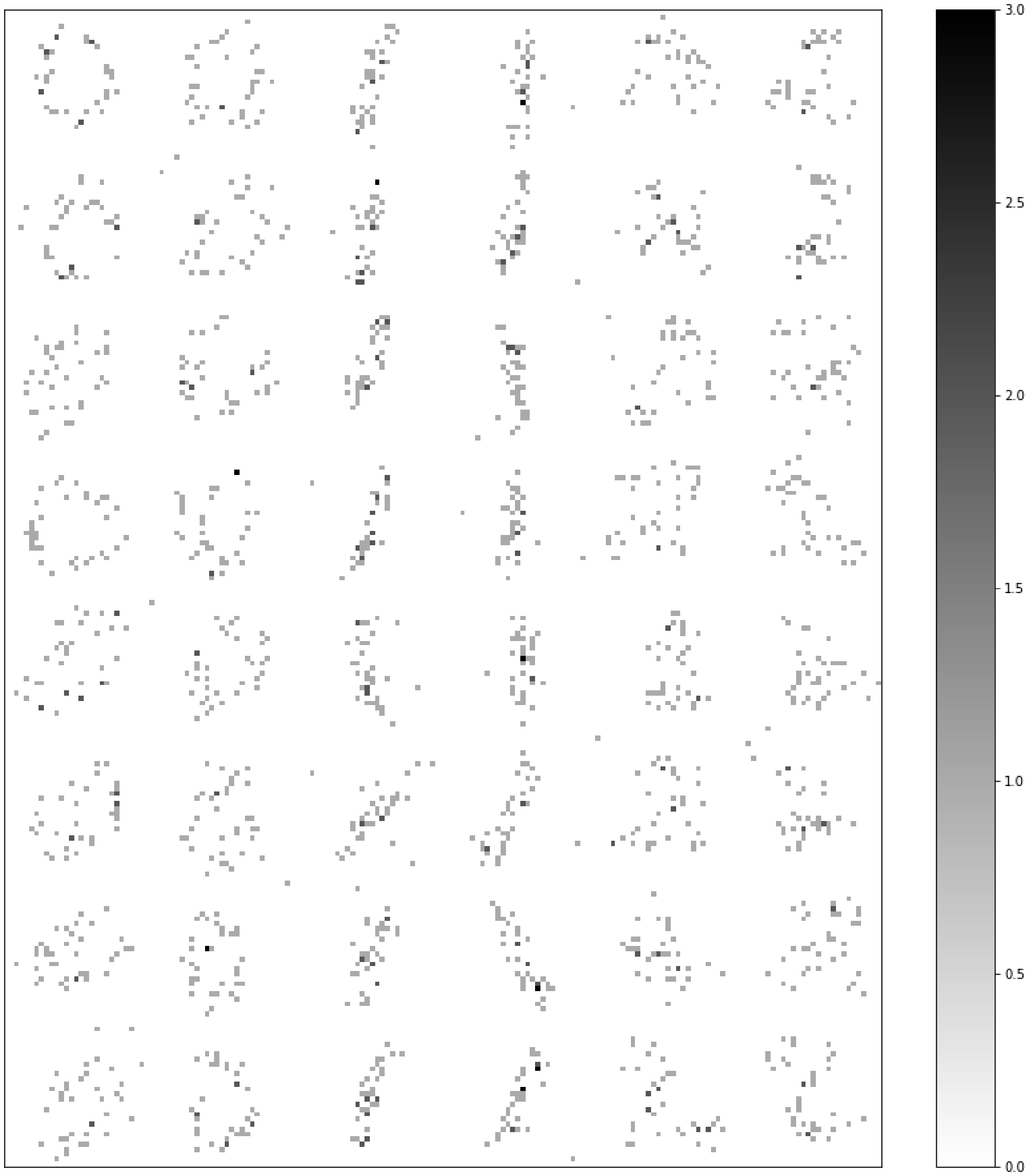

Figure 7. Visualization of the training set for the supervised learning evaluation. The examples (augmented versions of original digits) are shown in their original form, without the upscaling performed in other figures.

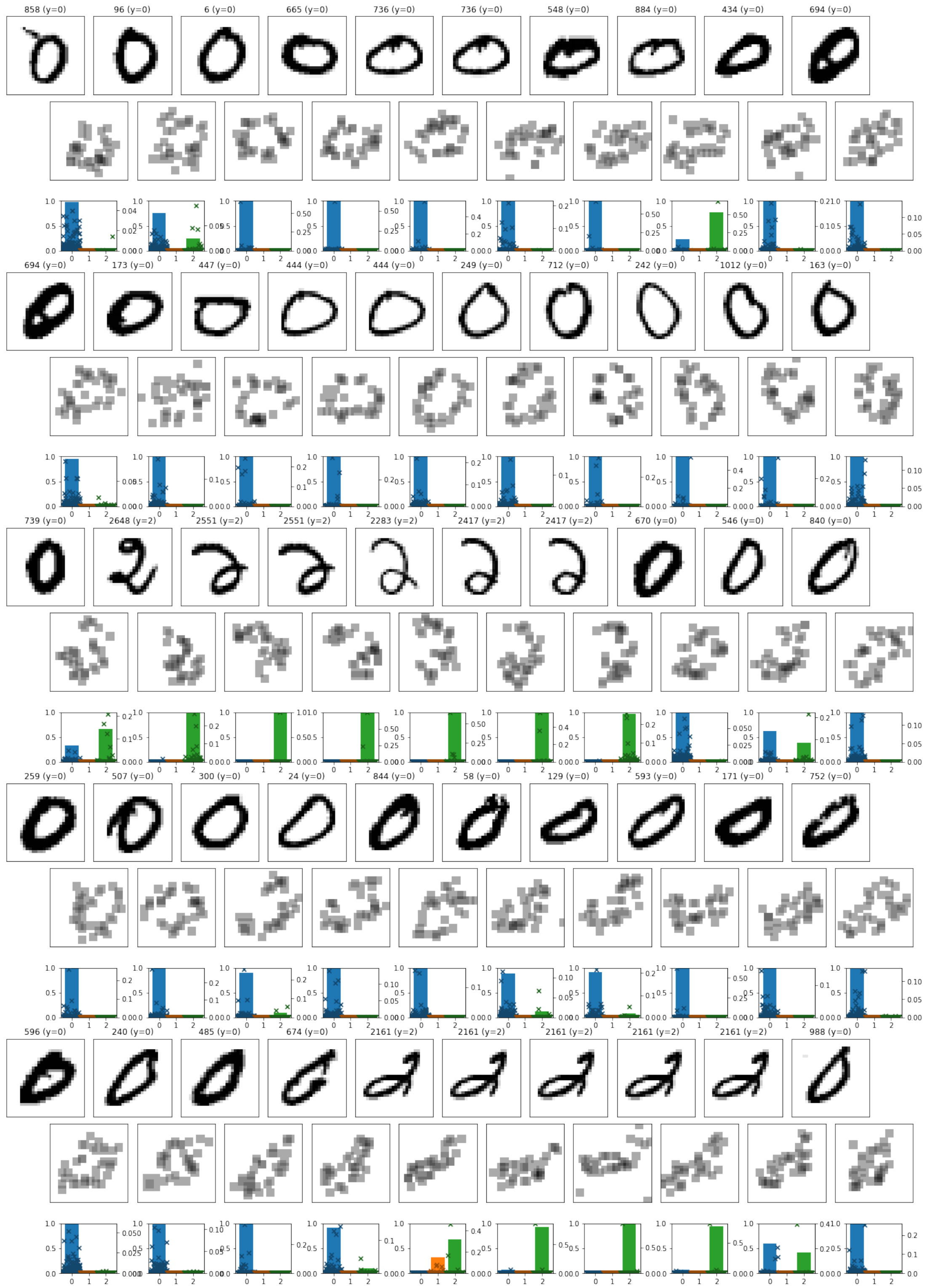

Figure 8. An extended version of the Markov chain trajectory shown in Figure 2. Below each digit, we show a bar plot giving the conditional probability of each original digit label (scale on left y-axis), along with a scatter plot giving the conditional probability of each individual dataset example in $\mathcal{X}$ (scale on right y-axis), sorted by digit type.

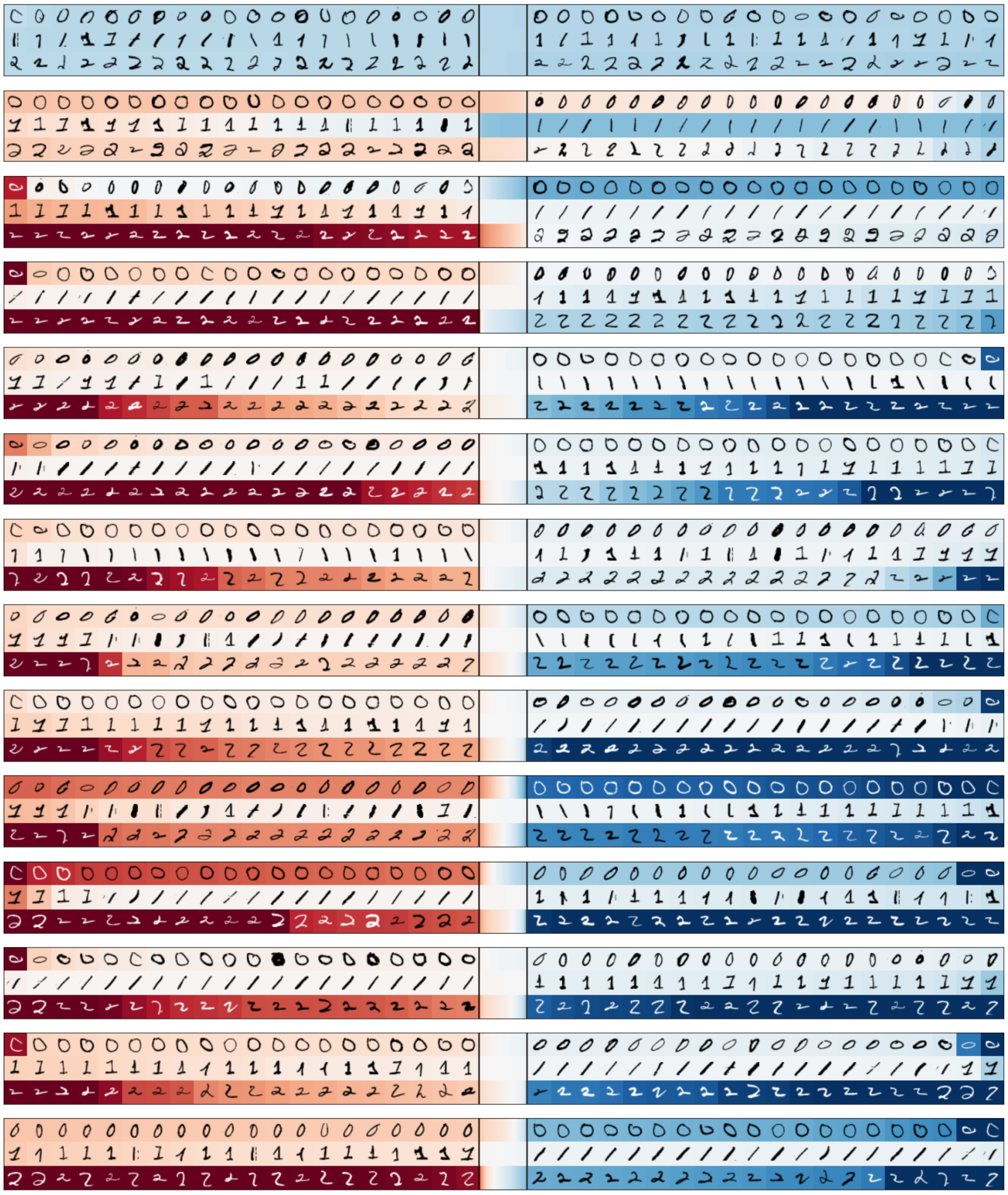

Figure 9. A visualization of the largest principal component directions for the contrastive kernel feature map ϕ_{Ctr}. For each row, we plot the dataset examples corresponding to the largest and smallest elements of those principal component directions, and also the overall distribution of values over the elements not visualized, shown with a gradient in the center. Note that this figure directly visualizes the directions in the space $\mathbb{R}^{\mathcal{X}}$; to compute the principal component projections for an augmentation $ain\mathcal{A}$, we first compute $\phi_{\mathrm{Ctr}}(a)$, which puts higher weight on xs that could have generated a, then compare the resulting scores to each of these rows.

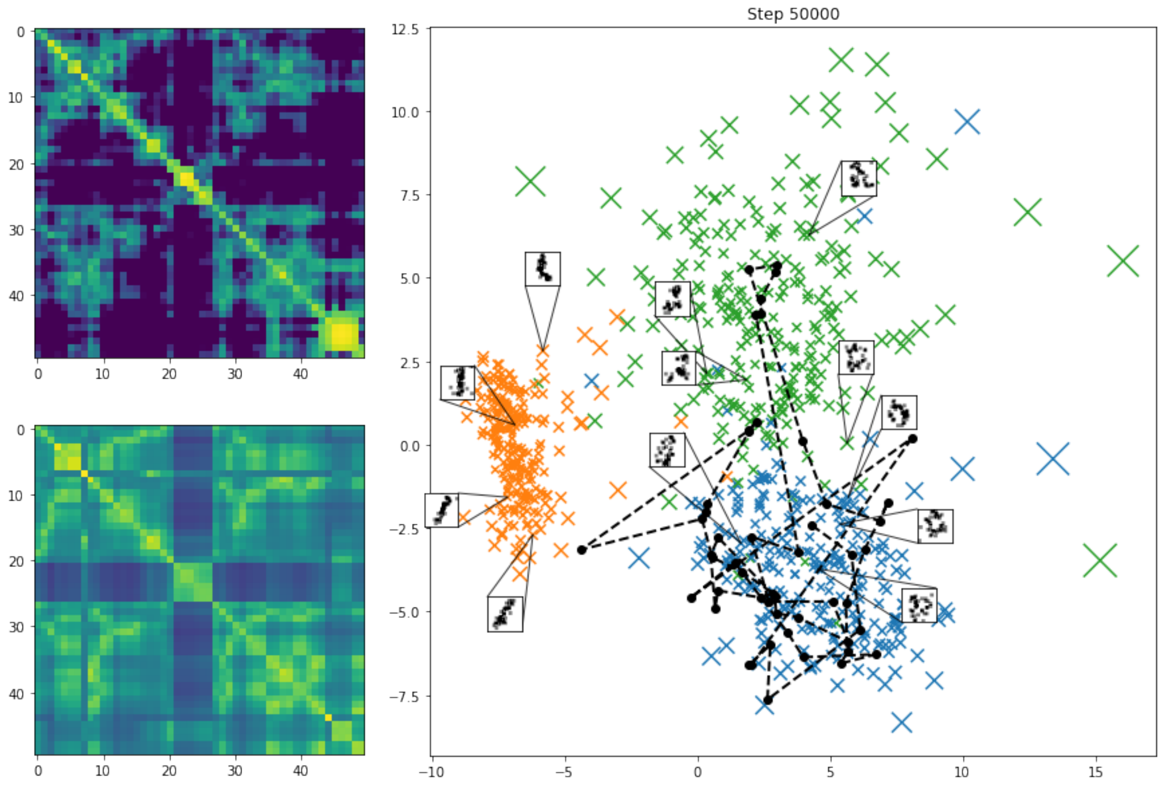

Figure 10. Left: Comparison of the Gram matrices for the true contrastive kernel K_{Ctr} and our approximation $\widehat{K}_\theta$ for the set of augmentations shown in the Markov chain in Figure 8. Right: Visualization of the learned embedding space of our ResNet-18 model; magnitude of the scale multiplier $g_\theta(a)$ is shown as marker size. Note that the model learns to prevent overcrowding by reducing the scale near cluster centers. Dashed line shows how the embeddings change as we move along the Markov chain trajectory in Figure 8; most steps move only a small distance.