On the Dynamics of Gradient Descent for Autoencoders

Thanh V. Nguyen[⋆], **Raymond K. W. Wong**[†], **Chinmay Hegde**[⋆]
[⋆]Iowa State University, [†]Texas A&M University

Abstract

We provide a series of results for unsupervised learning with autoencoders. Specifically, we study shallow two-layer autoencoder architectures with shared weights. We focus on three generative models for data that are common in statistical machine learning: (i) the mixture-of-gaussians model, (ii) the sparse coding model, and (iii) the sparsity model with non-negative coefficients. For each of these models, we prove that under suitable choices of hyperparameters, architectures, and initialization, autoencoders learned by gradient descent can successfully recover the parameters of the corresponding model. To our knowledge, this is the first result that rigorously studies the dynamics of gradient descent for weight-sharing autoencoders. Our analysis can be viewed as theoretical evidence that shallow autoencoder modules indeed can be used as feature learning mechanisms for a variety of data models, and may shed insight on how to train larger stacked architectures with autoencoders as basic building blocks.

1 Introduction

1.1 Motivation

Due to the resurgence of neural networks and deep learning, there has been growing interest in the community towards a thorough and principled understanding of training neural networks in both theoretical and algorithmic aspects. This has led to several important breakthroughs recently, including provable algorithms for learning shallow (1-hidden layer) networks with nonlinear activations [1, 2, 3, 4], deep networks with linear activations [5], and residual networks [6, 7].

Proceedings of the 22[nd] International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

A typical approach adopted by this line of work is as follows: assume that the data obeys a ground truth *generative model* (induced by simple but reasonably expressive data-generating distributions), and prove that the weights learned by the proposed algorithms (either exactly or approximately) recover the parameters of the generative model. Indeed, such distributional assumptions are necessary to overcome known NP-hardness barriers for learning neural networks [8]. Nevertheless, the majority of these approaches have focused on neural network architectures for supervised learning, barring a few exceptions which we detail below.

1.2 Our contributions

In this paper, we complement this line of work by providing new theoretical results for *unsupervised* learning using neural networks. Our focus here is on shallow two-layer autoencoder architectures with shared weights. Conceptually, we build upon previous theoretical results on learning autoencoder networks [9, 10, 11], and we elaborate on the novelty of our work in the discussion on prior work below.

Our setting is standard: we assume that the training data consists of i.i.d. samples from a high-dimensional distribution parameterized by a generative model, and we train the weights of the autoencoder using ordinary (batch) gradient descent. We consider three families of generative models that are commonly adopted in machine learning: (i) the Gaussian mixture model with well-separated centers [12]; (ii) the k-sparse model, specified by sparse linear combination of atoms [13]; and (iii) the non-negative k-sparse model [11]. While these models are traditionally studied separately depending on the application, all of these model families can be expressed via a unified, generic form:

$$y = Ax^* + \eta, \tag{1}$$

which we (loosely) dub as the *generative bilinear model*. In this form, A is a groundtruth $n \times m$-matrix, x^* is an m-dimensional latent code vector and η is an independent n-dimensional random noise vector. Samples y's are what we observe. Different choices of n and m,

as well as different assumptions on A and x^* lead to the three aforementioned generative models.

Under these three generative models, and with suitable choice of hyper-parameters, initial estimates, and autoencoder architectures, we rigorously prove that:

> *Two-layer autoencoders, trained with (normalized) gradient descent over the reconstruction loss, provably learn the parameters of the underlying generative bilinear model.*

To the best of our knowledge, our work is the first to analytically characterize the dynamics of gradient descent for training two-layer autoencoders. Our analysis can be viewed as theoretical evidence that shallow autoencoders can be used as feature learning mechanisms (provided the generative modeling assumptions hold), a view that seems to be widely adopted in practice. Our analysis highlights the following interesting conclusions: (i) the activation function of the hidden (encoder) layer influences the choice of bias; (ii) the bias of each hidden neuron in the encoder plays an important role in achieving the convergence of the gradient descent; and (iii) the gradient dynamics depends on the complexity of the generative model. Further, we speculate that our analysis may shed insight on practical considerations for training deeper networks with stacked autoencoder layers as building blocks [9].

1.3 Techniques

Our analysis is built upon recent algorithmic developments in the sparse coding literature [14, 15, 16]. Sparse coding corresponds to the setting where the synthesis coefficient vector $x^{*(i)}$ in (1) for each data sample $y^{(i)}$ is assumed to be k-sparse, i.e., $x^{*(i)}$ only has at most $k \ll m$ non-zero elements. The exact algorithms proposed in these papers are all quite different, but at a high level, all these methods involve establishing a notion that we dub as "support consistency". Broadly speaking, for a given data sample $y^{(i)} = Ax^{*(i)} + \eta^{(i)}$, the idea is that when the parameter estimates are close to the ground truth, it is possible to accurately estimate the true support of the synthesis vector $x^{*(i)}$ for *each* data sample $y^{(i)}$.

We extend this to a broader family of generative models to form a notion that we call "code consistency". We prove that if initialized appropriately, the weights of the hidden (encoder) layer of the autoencoder provides useful information about the *sign* pattern of the corresponding synthesis vectors for every data sample. Somewhat surprisingly, the choice of activation function of each neuron in the hidden layer plays an important role in establishing code consistency and affects the possible choices of bias.

The code consistency property is crucial for establishing the correctness of gradient descent over the reconstruction loss. This turns out to be rather tedious due to the weight sharing — a complication which requires a substantial departure from the existing machinery for analysis of sparse coding algorithms — and indeed forms the bulk of the technical difficulty in our proofs. Nevertheless, we are able to derive explicit *linear* convergence rates for all the generative models listed above. We do not attempt to analyze other training schemes (such as stochastic gradient descent or dropout) but anticipate that our analysis may lead to further work along those directions.

1.4 Comparison with prior work

Recent advances in algorithmic learning theory has led to numerous provably efficient algorithms for learning Gaussian mixture models, sparse codes, topic models, and ICA (see [12, 13, 14, 15, 16, 17, 18, 19] and references therein). We omit a complete treatment of prior work due to space constraints.

We would like to emphasize that we do *not* propose a new algorithm or autoencoder architecture, nor are we the first to highlight the applicability of autoencoders with the aforementioned generative models. Indeed, generative models such as k-sparsity models have served as the motivation for the development of deep stacked (denoising) autoencoders dating back to the work of [20]. The paper [9] proves that stacked weight-sharing autoencoders can recover the parameters of sparsity-based generative models, but their analysis succeeds only for certain generative models whose parameters are themselves randomly sampled from certain distributions. In contrast, our analysis holds for a broader class of networks; we make no randomness assumptions on the parameters of the generative models themselves.

More recently, autoencoders have been shown to learn sparse representations [21]. The recent paper [11] demonstrates that under the sparse generative model, the standard squared-error reconstruction loss of ReLU autoencoders exhibits (with asymptotically many samples) critical points in a neighborhood of the ground truth dictionary. However, they do not analyze gradient dynamics, nor do they establish convergence rates. We complete this line of work by proving explicitly that gradient descent (with column-wise normalization) in the asymptotic limit exhibits linear convergence up to a radius around the ground truth parameters.

2 Preliminaries

Notation Denote by x_S the sub-vector of $x \in \mathbb{R}^m$ indexed by the elements of $S \subseteq [m]$. Similarly, let W_S

be the sub-matrix of $W \in \mathbb{R}^{n\times m}$ with columns indexed by elements in S. Also, define $\mathrm{supp}(x) \triangleq \{i \in [m] : x_i \neq 0\}$ as the support of x, $\mathrm{sgn}(x)$ as the element-wise sign of x and $\mathbf{1}_E$ as the indicator of an event E.

We adopt standard asymptotic notations: let $f(n) = O(g(n))$ (or $f(n) = \Omega(g(n))$) if there exists some constant $C > 0$ such that $|f(n)| \leq C|g(n)|$ (respectively, $|f(n)| \geq C|g(n)|$). Next, $f(n) = \Theta(g(n))$ is equivalent to that $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$. Also, $f(n) = \omega(g(n))$ if $\lim_{n\to\infty} |f(n)/g(n)| = \infty$. In addition, $g(n) = O^*(f(n))$ indicates $|g(n)| \leq K|f(n)|$ for some small enough constant K. Throughout, we use the phrase "with high probability" (abbreviated to w.h.p.) to describe any event with failure probability at most $n^{-\omega(1)}$.

2.1 Two-Layer Autoencoders

We focus on shallow autoencoders with a single hidden layer, n neurons in the input/output layer and m hidden neurons. We consider the *weight-sharing* architecture in which the encoder has weights $W^T \in \mathbb{R}^{m\times n}$ and the decoder uses the shared weight $W \in \mathbb{R}^{n\times m}$. The architecture of the autoencoder is shown in Fig. 1. Denote $b \in \mathbb{R}^m$ as the vector of biases for the encoder (we do not consider decoder bias.) As such, for a given data sample $y \in \mathbb{R}^n$, the encoding and decoding respectively can be modeled as:

$$x = \sigma(W^T y + b) \quad \text{and} \quad \hat{y} = Wx, \tag{2}$$

where $\sigma(\cdot)$ denotes the activation function in the encoder neurons. We consider two types of activation functions: (i) the rectified linear unit:

$$\mathrm{ReLU}(z) = \max(z, 0),$$

and (ii) the hard thresholding operator:

$$\mathrm{threshold}_\lambda(z) = z\mathbf{1}_{|z|\geq\lambda}.$$

When applied to a vector (or matrix), these functions are operated on each element and return a vector (respectively, matrix) of same size. Our choice of the activation $\sigma(\cdot)$ function varies with different data generative models, and will be clear by context.

Herein, the loss function is the (squared) reconstruction error:

$$L = \frac{1}{2}\|y - \hat{y}\|^2 = \frac{1}{2}\|y - W\sigma(W^T y + b)\|^2,$$

and we analyze the expected loss where the expectation is taken over the data distribution (specified below). Inspired by the literature of analysis of sparse coding [11, 14, 22], we investigate the landscape of the expected loss so as to shed light on dynamics of gradient descent for training the above autoencoder architectures. Indeed, we show that for a variety of data distributions, such autoencoders can recover the distribution parameters via suitably initialized gradient descent.

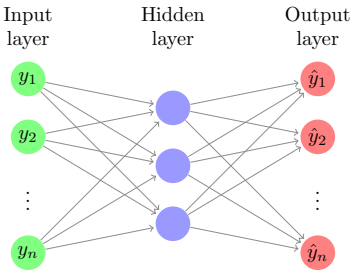

Figure 1: *Architecture of a shallow 2-layer autoencoder network. The encoder and the decoder share the weights.*

2.2 Generative Bilinear Model

We now describe an overarching generative model for the data samples. Specifically, we posit that the data samples $\{y^{(i)}\}_{i=1}^N \in \mathbb{R}^n$ are drawn according to the following "bilinear" model:

$$y = Ax^* + \eta, \tag{3}$$

where $A \in \mathbb{R}^{n\times m}$ is a ground truth set of parameters, $x^* \in \mathbb{R}^m$ is a latent code vector, and $\eta \in \mathbb{R}^n$ represents noise. Depending on different assumptions made on A and x^*, this model generalizes various popular cases, such as mixture of spherical Gaussians, sparse coding, nonnegative sparse coding, and independent component analysis (ICA). We will elaborate further on specific cases, but in general our generative model satisfies the following generic assumptions:

A1. The code x^* is supported on set S of size at most k, such that $p_i = \mathbb{P}[i \in S] = \Theta(k/m)$, $p_{ij} = \mathbb{P}[i, j \in S] = \Theta(k^2/m^2)$ and $p_{ijl} = \mathbb{P}[i, j, l \in S] = \Theta(k^3/m^3)$;

A2. Nonzero entries are independent; moreover, $\mathbb{E}[x_i^*|i \in S] = \kappa_1$ and $\mathbb{E}[x_i^{*2}|i \in S] = \kappa_2 < \infty$;

A3. For $i \in S$, $|x_i^*| \in [a_1, a_2]$ with $0 \leq a_1 \leq a_2 \leq \infty$;

A4. The noise term η is distributed according to $\mathcal{N}(0, \sigma_\eta^2 I)$ and is independent of x^*.

As special cases of the above model, we consider the following variants.

Mixture of spherical Gaussians: We consider the standard Gaussian mixture model with m centers,

which is one of the most popular generative models encountered in machine learning applications. We model the means of the Gaussians as columns of the matrix A. To draw a data sample y, we sample x^* uniformly from the canonical basis $\{e_i\}_{i=1}^m \in \mathbb{R}^n$ with probability $p_i = \Theta(1/m)$. As such, x^* has sparsity parameter $k = 1$ with only one nonzero element being 1. That means, $\kappa_1 = \kappa_2 = a_1 = a_2 = 1$.

Sparse coding: This is a well-known instance of the above structured linear model, where the goal is basically to learn an *overcomplete* dictionary A that sparsely represents the input y. It has a rich history in various fields of signal processing, machine learning and neuroscience [23]. The generative model described above has successfully enabled recent theoretical advances in sparse coding [13, 14, 15, 16, 24]. The latent code vector x^* is assumed to be k-sparse, whose nonzero entries are sub-Gaussian and bounded away from zero. Therefore, $a_1 > 0$ and $a_2 = \infty$. We assume that the distribution of nonzero entries are standardized such that $\kappa_1 = 0$, $\kappa_2 = 1$. Note that the condition of κ_2 further implies that $a_1 \leq 1$.

Non-negative sparse coding: This is another variant of the above sparse coding model where the elements of the latent code x^* are additionally required to be non-negative [11]. In some sense this is a generalization of the Gaussian mixture model described above. Since the code vector is non-negative, we *do not* impose the standardization as in the previous case of general sparse coding ($\kappa_1 = 0$ and $\kappa_2 = 1$); instead, we assume a compact interval of the nonzero entries; that is, a_1 and a_2 are *positive* and *bounded*.

Having established probabilistic settings for these models, we now establish certain deterministic conditions on the true parameters A to enable analysis. First, we require each column A_i to be normalized to unit norm in order to avoid the scaling ambiguity between A and x^*. (Technically, this condition is not required for the mixture of Gaussian model case since x^* is binary; however we make this assumption anyway to keep the treatment generic.) Second, we require columns of A to be "sufficiently distinct"; this is formalized by adopting the notion of pairwise incoherence.

Definition 1. *Suppose that $A \in \mathbb{R}^{n\times m}$ has unit-norm columns. A is said to be μ-incoherent if for every pair of column indices (i, j), $i \neq j$ we have $|\langle A_i, A_j\rangle| \leq \frac{\mu}{\sqrt{n}}$.*

Though this definition is motivated from the sparse coding literature, pairwise incoherence is sufficiently general to enable identifiability of all aforementioned models. For the mixture of Gaussians with unit-norm means, pairwise incoherence states that the means are well-separated, which is a standard assumption. In the case of Gaussian mixtures, we assume that $m = O(1) \ll n$. For sparse coding, we focus on learning overcomplete dictionaries where $n \leq m = O(n)$. For the sparse coding case, we further require the spectral norm bound on A, i.e., $\|A\| \leq O(\sqrt{m/n})$. (In other words, A is well-conditioned.)

Our eventual goal is to show that training autoencoder via gradient descent can effectively recover the generative model parameter A. To this end, we need a measure of goodness in recovery. Noting that any recovery method can only recover A up to a permutation ambiguity in the columns (and a sign-flip ambiguity in the case of sparse coding), we first define an operator π that permutes the columns of the matrix (and multiplies by $+1$ or -1 individually to each column in the case of sparse coding.) Then, we define our measure of goodness:

Definition 2 (δ-closeness and (δ, ξ)-nearness)**.** *A matrix W is said to be δ-close to A if there exists an operator $\pi(\cdot)$ defined above such that $\|\pi(W)_i - A_i\| \leq \delta$ for all i. We say W is (δ, ξ)-near to A if in addition $\|\pi(W) - A\| \leq \xi\|A\|$.*

To simplify notation, we simply replace π by the identity operator while keeping in mind that we are only recovering an element from the equivalence class of all permutations and sign-flips of A.

Armed with the above definitions and assumptions, we are now ready to state our results. Since the actual mathematical guarantees are somewhat tedious and technical, we summarize our results in terms of informal theorem statements, and elaborate more precisely in the following sections.

Our first main result establishes the code consistency of weight-sharing autoencoders under all the generative linear models described above, provided that the weights are suitably initialized.

Theorem 1 (informal)**.** *Consider a sample $y = Ax^* + \eta$. Let $x = \sigma(W^T y + b)$ be the output of the encoder part of the autoencoder. Suppose that W is δ-close to A with $\delta = O^*(1/\log n)$.*

(i) If $\sigma(\cdot)$ is either the ReLU or the hard thresholding activation, then the support of the true code vector x^* matches that of x for the mixture-of-Gaussians and non-negative sparse coding generative models.

(ii) If $\sigma(\cdot)$ is the hard thresholding activation, then the support of x^* matches that of x for the sparse coding generative model.

Our second main result leverages the above property. We show that iterative gradient descent over the weights W *linearly* converges to a small neighborhood of the ground truth.

Theorem 2 (informal)**.** *Provided that the initial weight W^0 such that W^0 is $(\delta, 2)$-near to A. Given asymptotically many samples drawn from the above models, an iterative gradient update of W can linearly converge to a small neighborhood of the ground truth A.*

We formally present these technical results in the next sections. Note that we analyze the encoding and the gradient given W^s at iteration s; however we often skip the superscript for clarity.

3 Initialization

Our main result is a local analysis of the learning dynamics for two-layer autoencoders. More specifically, we prove that the (batch) gradient descent linearly converges to the ground truth parameter A given an initialization W^0 that is $O^*(1/\log n)$ column-wise close to the ground truth. Despite the fact that the recovery error at the convergence is exponentially better than the initial $1/\log n$ order, a natural question is how to achieve this initialization requirement. In practice, random initialization for autoencoders is a common strategy and it often leads to surprisingly good results [25, 26]. In theory, however, the validity of the random initialization is still an open problem. For the k-sparse model, the authors in [16] introduce an algorithm that provably produces such a coarse estimate of A using spectral methods. This algorithm applies perfectly to this context of the autoencoder architecture. We conjecture that this spectral algorithm still works for non-negative sparse case (including the special mixture of Gaussian model) although, due to non-negativity, more complicated treatments including concentration arguments and sign flips of the columns are involved. We leave this to our future work.

4 Encoding Stage

Our technical results start with the analysis of the encoding stage in the forward pass. We rigorously prove that the encoding performed by the autoencoder is sufficiently good in the sense that it recovers part of the information in the latent code x^* (specifically, the signed support of x^*.) This is achieved based on appropriate choices of activation function, biases, and a good W within close neighborhood of the true parameters A. We call this property code consistency:

Theorem 3 (Code consistency)**.** *Let $x = \sigma(W^T y + b)$. Suppose W is δ-close to A with $\delta = O^*(1/\log n)$ and the noise satisfies $\sigma_\eta = O(1/\sqrt{n})$. Then the following results hold:*

(i) General k-sparse code with thresholding activation: Suppose $\mu \le \sqrt{n}/\log^2 n$ and $k \le n/\log n$. If $x = \text{threshold}_\lambda(W^T y + b)$ with $\lambda = a_1/2$ and $b = 0$, then with high probability

$$\text{sgn}(x) = \text{sgn}(x^*).$$

(ii) Non-negative k-sparse code with ReLU activation: Suppose $\mu \le \delta\sqrt{n}/k$ and $k = O(1/\delta^2)$. If $x = \text{ReLU}(W^T y + b)$, and $b_i \in [-(1-\delta)a_1 + a_2\delta\sqrt{k},\ -a_2\delta\sqrt{k}]$ for all i, then with high probability,

$$\text{supp}(x) = \text{supp}(x^*).$$

(iii) Non-negative k-sparse code with thresholding activation: Suppose $\mu \le \delta\sqrt{n}/k$ and $k = O(1/\delta^2)$. If $x = \text{threshold}_\lambda(W^T y + b)$ with $\lambda = a_1/2$ and $b = 0$, then with high probability,

$$\text{supp}(x) = \text{supp}(x^*).$$

The full proof for Theorem 3 is relegated to Appendix A. Here, we provide a short proof for the mixture-of-Gaussians generative model, which is really a special case of *(ii)* and *(iii)* above, where $k = 1$ and the nonzero component of x^* is equal to 1 (i.e., $\kappa_1 = \kappa_2 = a_1 = a_2 = 1$.)

Proof. Denote $z = W^T y + b$ and $S = \text{supp}(x^*) = \{j\}$. Let i be fixed and consider two cases: if $i = j$, then

$$z_i = \langle W_i, A_i\rangle + \langle W_i, \eta\rangle + b_i \ge (1 - \delta^2/2) - \sigma_\eta \log n + b_i > 0,$$

w.h.p. due to the fact that $\langle W_i, A_i\rangle \ge 1 - \delta^2/2$ (Claim 1), and the conditions $\sigma_\eta = O(1/\sqrt{n})$ and $b_i > -1 + \delta$.

On the other hand, if $i \ne j$, then using Claims 1 and 2 in Appendix A, we have w.h.p.

$$z_i = \langle W_i, A_j\rangle + \langle W_i, \eta\rangle + b_i \le \mu/\sqrt{n} + \delta + \sigma_\eta \log n + b_i < 0,$$

for $b_i \le -2\delta, \mu \le \delta\sqrt{n}/k$ and $\sigma_\eta = O(1/\sqrt{n})$. Due to Claim 2, these results hold w.h.p. uniformly for all i, and hence $x = \text{ReLU}(z)$ has the same support as x^* w.h.p..

Moreover, one can also see that when $b_i = 0$, then w.h.p., $z_i > 1/2$ if $i = j$ and $z_i < 1/4$ otherwise. This result holds w.h.p. uniformly for all i, and therefore, $x = \text{threshold}_{1/2}(z)$ has the same support as x^* w.h.p. ■

Note that for the non-negative case, both ReLU and threshold activation would lead to a correct support of the code, but this requires $k = O(1/\delta^2)$, which is rather restrictive and might be a limitation of the current analysis. Also, in Theorem 3, b is required to be negative for ReLU activation for any $\delta > 0$ due to the error of the current estimate W. However, this result is

consistent with the conclusion of [27] that negative bias is desirable for ReLU activation to produce sparse code. Note that such choices of b also lead to statistical bias (error) in nonzero code and make it difficult to construct a provably correct learning procedure (Section 5) for ReLU activation.

Part (i) of Theorem 3 mirrors the consistency result established for sparse coding in [16].

Next, we apply the above result to show that provided the consistency result a (batch) gradient update of the weights W (and bias in certain cases) converges to the true model parameters.

5 Learning Stage

In this section, we show that a gradient descent update for W of the autoencoder (followed by a normalization in the Euclidean column norm of the updated weights) leads to a linear convergence to a small neighborhood of the ground truth A under the aforementioned generative models. For this purpose, we analyze the gradient of the expected loss with respect to W. Our analysis involves calculating the expected value of the gradient as if we were given infinitely many samples. (The finite sample analysis is left as future work.)

Since both ReLU and hard thresholding activation functions are non-differentiable at some values, we will formulate an approximate gradient. Whenever differentiable, the gradient of the loss L with respect to the *column* $W_i \in \mathbb{R}^n$ of the weight matrix W is given by:

$$\nabla_{W_i} L = -\sigma'(W_i^T y + b_i)\big[(W_i^T y + b_i)I + y W_i^T\big]\big[y - Wx\big], \tag{4}$$

where $x = \sigma(W^T y + b)$ and $\sigma'(z_i)$ is the gradient of $\sigma(z_i)$ at z_i where σ is differentiable. For the rectified linear unit $\mathrm{ReLU}(z_i) = \max(z_i, 0)$, its gradient is

$$\sigma'(z_i) = \begin{cases} 1 & \text{if } z_i > 0, \\ 0 & \text{if } z_i < 0. \end{cases}$$

On the other hand, for the hard thresholding activation $\mathrm{threshold}_\lambda(z_i) = z_i \mathbf{1}_{|z_i| \geq \lambda}$, the gradient is

$$\sigma'(z_i) = \begin{cases} 1 & \text{if } |z_i| > \lambda, \\ 0 & \text{if } |z_i| < \lambda. \end{cases}$$

One can see that in both cases, the gradient $\sigma'(\cdot)$ at $z_i = W_i^T y + b_i$ resembles an indicator function $\mathbf{1}_{x_i \neq 0} = \mathbf{1}_{\sigma(z_i) \neq 0}$ except where it is not defined. The observation motivates us to approximate the $\nabla_{W_i} L$ with a simpler rule by replacing $\sigma'(W_i^T y + b_i)$ with $\mathbf{1}_{x_i \neq 0}$:

$$\widetilde{\nabla_i L} = -\mathbf{1}_{x_i \neq 0}(W_i^T y I + b_i I + y W_i^T)(y - Wx).$$

In fact, [11] (Lemma 5.1) shows that this approximate gradient $\widetilde{\nabla_i L}$ is a good approximation of the true gradient (4) in expectation. Since A is assumed to have normalized columns (with $\|A_i\| = 1$), we can enforce this property to the update by a simple column normalization after every update; to denote this, we use the operator normalize$(\cdot)$ that returns a matrix normalize(B) with unit columns, i.e.:

$$\mathrm{normalize}(B)_i = B_i / \|B_i\|,$$

for any matrix B that has no all-zero columns.

Our convergence result leverages the code consistency property in Theorem 3, but in turn succeeds under constraints on the biases of the hidden neurons b. For thresholding activation, we can show that the simple choice of setting all biases to zero leads to both code consistency and linear convergence. However, for ReLU activation, the range of bias specified in Theorem 3 (ii) has a profound effect on the descent procedure. Roughly speaking, we need non-zero bias in order to ensure code consistency, but high values of bias can adversely impact gradient descent. Indeed, our current analysis does not succeed for any *constant* choice of bias (i.e., we do not find a constant bias that leads to both support consistency and linear convergence.) To resolve this issue, we propose to use a simple *diminishing* (in magnitude) sequence of biases b along different iterations of the algorithm. Overall, this combination of approximate gradient and normalization lead to an update rule that certifies the existence of a linear convergent algorithm (up to a neighborhood of A.) The results are formally stated as follows:

Theorem 4 (Descent property)**.** *Suppose that at step s the weight W^s is $(\delta^s, 2)$-near to A. There exists an iterative update rule using an approximate gradient g^s: $W^{s+1} = \mathrm{normalize}(W^s - \zeta g^s)$ that linearly converges to A when given infinitely many* fresh *samples. More precisely, there exists some $\tau \in (1/2, 1)$ such that:*

(i) Mixture of Gaussians: Suppose the conditions in either (ii) or (iii) of Theorem 3 hold. Suppose that the learning rate $\zeta = \Theta(m)$, and that the bias vector b satisfies:

(i.1) $b = 0$ if $x = \mathrm{threshold}_{1/2}(W^T y + b)$; or

(i.2) $b^{s+1} = b^s / C$ if $x = \mathrm{ReLU}(W^T y + b)$ for some constant $C > 1$.

Then, $\|W^{s+1} - A\|_F^2 \leq (1 - \tau)\|W^s - A\|_F^2 + O(mn^{-O(1)})$.

(ii) General k-sparse code: Provided the conditions in Theorem 3 (i) hold and the learning rate $\zeta = \Theta(m/k)$.

Then, $\|W^{s+1} - A\|_F^2 \leq (1 - \tau)\|W^s - A\|_F^2 + O(mk^2/n^2)$.

(iii) Non-negative k-sparse code: Suppose the conditions in either (ii) or (iii) of Theorem 3 hold. Suppose that the learning rate $\zeta = \Theta(m/k)$ and the bias b satisfies:

(iii.1) $b = 0$ *if* $x = \text{threshold}_{a_1/2}(W^T y + b)$*; or*

(iii.2) $b^{s+1} = b^s/C$ *if* $x = \text{ReLU}(W^T y + b)$ *for some constant* $C > 1$.

Then, $\|W^{s+1} - A\|_F^2 \leq (1-\tau)\|W^s - A\|_F^2 + O(k^3/m)$.

Recall the approximate gradient of the squared loss:

$$\widetilde{\nabla_i L} = -\mathbf{1}_{x_i \neq 0}(W_i^T y I + b_i I + y W_i^T)(y - Wx).$$

We will use this form to construct a desired update rule with linear convergence. Let us consider an update step g^s in expectation over the code x^* and and the noise η:

$$g_i = -\mathbb{E}[\mathbf{1}_{x_i \neq 0}(W_i^T y I + b_i I + y W_i^T)(y - Wx)]. \quad (5)$$

To prove Theorem 4, we compute g_i according to the generative models described in (3) and then argue the descent. Here, we provide a proof sketch for (again) the simplest case of mixture-of-Gaussians; the full proof is deferred to Appendix B.

Proof of Theorem 4 (i). Based on Theorem 3, one can explicitly compute the expectation expressed in (5). Specifically, the expected gradient g_i is of the form:

$$g_i = -p_i \lambda_i A + p_i(\lambda_i^2 + 2b_i\lambda_i + b_i^2)W_i + \gamma$$

where $\lambda_i = \langle W_i^s, A_i \rangle$ and $\|\gamma\| = O(n^{-w(1)})$. If we can find b_i such that $\lambda_i^2 + 2b_i\lambda_i + b_i^2 \approx \lambda_i$ for all i, g_i roughly points in the same desired direction to A_i, and therefore, a descent property can be established via the following result:

Lemma 1. *Suppose W is δ-close to A and the bias satisfies $|(b_i + \lambda_i)^2 - \lambda_i| \leq 2(1-\lambda_i)$. Then:*

$$2\langle g_i, W_i - A_i \rangle \geq p_i(\lambda_i - 2\delta^2)\|W_i - A_i\|^2 + \frac{1}{p_i\lambda_i}\|g_i\|^2 - \frac{2}{p_i\lambda_i}\|\gamma\|^2$$

From Lemma 1, one can easily prove the descent property using [16] (Theorem 6). We apply this lemma with learning rate $\zeta = \max_i(1/p_i\lambda_i)$ and $\tau = \zeta p_i(\lambda_i - 2\delta^2) \in (0,1)$ to achieve the descent as follows:

$$\|\widetilde{W}_i^{s+1} - A_i\|^2 \leq (1-\tau)\|W_i^s - A_i\|^2 + O(n^{-K}),$$

where $\widetilde{W}^{s+1} = W^s - \zeta g_s$ and K is some constant greater than 1. Finally, we use Lemma 5 to obtain the descent property for the normalized W_i^{s+1}.

Now, we determine when the bias conditions in Theorem 3 and Lemma 1 simultaneously hold for different choices of activation function. For the hard-thresholding function, since we do not need bias (i.e. $b_i = 0$ for every i), then $\lambda_i(1-\lambda_i) \leq 2(1-\lambda_i)$ and this lemma clearly follows.

On the other hand, if we encode $x = \text{ReLU}(W^T y + b)$, then we need every bias b_i to satisfy $b_i \in [-1 + 2\delta^s\sqrt{k}, -\delta^s]$ and $|(b_i + \lambda_i)^2 - \lambda_i| \leq 2(1-\lambda_i)$. Since $\lambda_i = \langle W_i^s, A_i \rangle \to 1$ and $\delta^s \to 0$, for the conditions of b_i to hold, we require $b_i \to 0$ as s increases. Hence, a fixed bias for the rectified linear unit would not work. Instead, we design a simple update for the bias (and this is enough to prove convergence in the ReLU case).

Here is our intuition. The gradient of L with respect to b_i is given by:

$$\nabla_{b_i} L = -\sigma'(W_i^T y + b_i)W_i^T(y - Wx)$$

Similarly to the update for the weight matrix, we approximate this gradient with by replacing $\sigma'(W_i^T y + b_i)$ with $\mathbf{1}_{x_i \neq 0}$, calculate the expected gradient and obtain:

$$\begin{aligned}(g_b)_i &= -\mathbb{E}[W_i^T(y - Wx)\mathbf{1}_{x_i^* \neq 0}] + \gamma \\ &= -\mathbb{E}[W_i^T(y - W_i(W_i^T y + b_i)\mathbf{1}_{x_i^* \neq 0}] + \gamma \\ &= -\mathbb{E}[(W_i^T - \|W_i\|^2 W_i^T)y + \|W_i\|^2 b_i \mathbf{1}_{x_i^* \neq 0}] + \gamma \\ &= -p_i b_i + \gamma\end{aligned}$$

From the expected gradient formula, we design a very simple update for the bias: $b^{s+1} = \sqrt{1-\tau}b^s$ where $b^0 = -1/\log n$, and show by induction that this choice of bias is sufficiently negative to make the consistency result 3 (ii) and (iii) hold at each step. At the first step, we have $\delta^0 \leq O^*(1/\log n)$, then

$$b_i^0 = -1/\log n \leq -\|W_i^0 - A_i\|.$$

Now, assuming $b_i^s \leq -\|W_i^s - A_i\|$, we need to prove that $b^{s+1} \leq -\|W_i^{s+1} - A_i\|$.

From the descent property at the step s, we have

$$\|W_i^{s+1} - A_i\| \leq \sqrt{1-\tau}\|W_i^s - A_i\| + o(\delta^s).$$

Therefore, $b_i^{s+1} = \sqrt{1-\tau}b_i^s \leq -\sqrt{1-\tau}\|W_i^s - A_i\| \leq -\|W_i^{s+1} - A_i\| - o(\delta_s)$. As a result, $|(b_i + \lambda_i)^2 - \lambda_i| \approx \lambda_i(1-\lambda_i) \leq 2(1-\lambda_i)$. In addition, the condition of bias in the support consistency holds. By induction, we can guarantee the consistency at all the update steps. Lemma 1 and hence the descent results stated in (i.2) and (iii.2) hold for the special case of the Gaussian mixture model. ■

6 Experiments

We support our theoretical results with some experiments on synthetic data sets under on the mixture-of-Gaussians model. We stress that these experimental

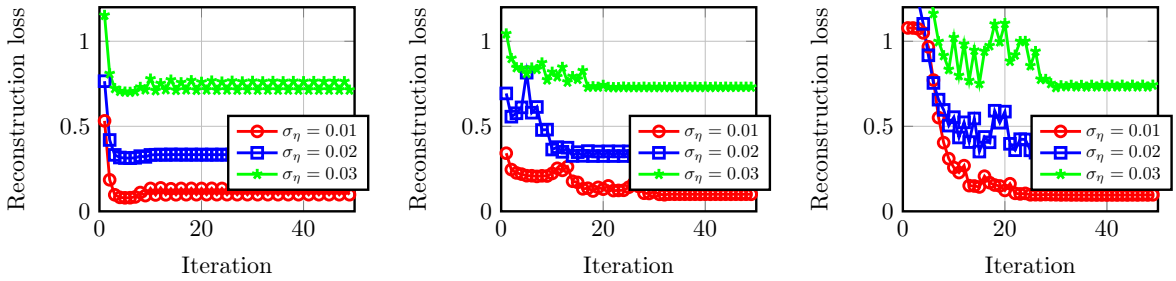

Figure 2: *The learning curve in training step using different initial estimate W^0. From left to right, the autoencoder is initialized by (i) some perturbation of the ground truth, (ii) PCA and (iii) random guess.*

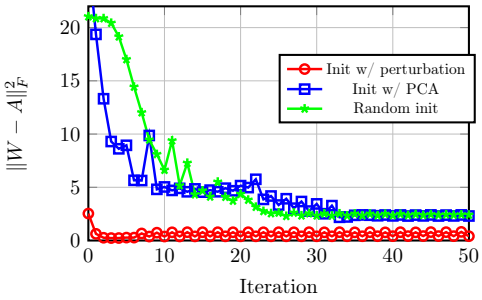

Figure 3: *Frobenius norm difference between the learned W and the ground truth A by three initialization schemes.*

results are *not* intended to be exhaustive or of practical relevance, but rather only to confirm some aspects of our theoretical results, and shed light on where the theory falls short.

We generate samples from a mixture of $m = 10$ Gaussians with dimension $n = 784$ using the model $y = Ax^* + \eta$. The means are the columns of A, randomly generated according to $A_i \sim \mathcal{N}(0, \frac{1}{\sqrt{n}} I_n)$. To synthesize each sample y, we choose x^* uniformly from the canonical bases $\{e_i\}_{i=1}^m$ and generate a Gaussian noise vector η with independent entries and entry-wise standard deviation σ_η. We create a data set of $10,000$ samples in total for each Monte Carlo trial.

We consider a two-layer autoencoder with shared weights as described in Section 2.1, such that the hidden layer has 10 units with ReLU activation. Then, we observe its gradient dynamics on the above data using three different initializations: (i) we initialize W by adding small random perturbation to the groundtruth A such that $W^0 = A + \delta E$ for $\delta = 0.5$ with the perturbation $E \in \mathbb{R}^{784 \times 10}$ generated according to $E_{ij} \sim \mathcal{N}(0, 1/\sqrt{n})$; (ii) we perform principal component analysis of the data samples and choose the top 10 singular vectors as W^0; (iii) we randomly generate W with $W_i \sim \mathcal{N}(0, \frac{1}{\sqrt{n}} I_n)$.

For all three initializations, the bias b of the encoder are initially set to $b = -2.5\delta$. We train the weights W with the batch gradient descent and update the bias using a fixed update rule $b^{s+1} = b^s/2$.

The learning rate for gradient descent is set fixed to $\zeta = m$. The number of descent steps is $T = 50$. We run the batch descent algorithm at each initialization with different levels of noise ($\sigma_\eta = 0.01, 0.02, 0.03$), then we observe the reconstruction loss over the data samples.

Figure 2 shows the learning curve in the number of iterations. From the left, the first plot is the loss with the initial point 0.5-close to A. The next two plots represent the learning using the PCA and random initializations. The gradient descent also converges when using the same step size and bias as described above. The convergence behavior is somewhat unexpected; *even* with random initialization the reconstruction loss decreases to low levels when the noise parameter σ_η is small. This suggests that the loss surface is perhaps amenable to optimization even for radius bigger than $O(\delta)$-away from the ground truth parameters, although our theory does not account for this.

In Figure 3 we show the Frobenius norm difference between the ground truth A and final solution W using three initialization schemes on a data set with noise $\sigma_\eta = 0.01$. Interestingly, despite the convergence, neither PCA nor random initialization leads to the recovery of the ground truth A. Note that since we can only estimate W up to some column permutation, we use the Hungarian algorithm to compute matching between W and A and then calculate the norm.

Conclusions To our knowledge, the above analysis is the first to prove rigorous convergence of gradient dynamics for autoencoder architectures for a wide variety of (bilinear) generative models. Numerous avenues for future work remain — finite sample complexity analysis; extension to more general architectures; and extension to richer classes of generative models.

7 Acknowledgements

This work was supported in part by the National Science Foundation under grants CCF-1566281, CAREER CCF-1750920 and DMS-1612985/1806063, and in part by a Faculty Fellowship from the Black and Veatch Foundation.

References

[1] Yuandong Tian. Symmetry-breaking convergence analysis of certain two-layered neural networks with relu nonlinearity. 2017.

[2] Rong Ge, Jason D Lee, and Tengyu Ma. Learning one-hidden-layer neural networks with landscape design. *arXiv preprint arXiv:1711.00501*, 2017.

[3] Alon Brutzkus and Amir Globerson. Globally optimal gradient descent for a convnet with gaussian inputs. In *International Conference on Machine Learning*, pages 605–614, 2017.

[4] Kai Zhong, Zhao Song, Prateek Jain, Peter L Bartlett, and Inderjit S Dhillon. Recovery guarantees for one-hidden-layer neural networks. In *International Conference on Machine Learning*, pages 4140–4149, 2017.

[5] Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, pages 586–594, 2016.

[6] Yuanzhi Li and Yang Yuan. Convergence analysis of two-layer neural networks with relu activation. In *Advances in Neural Information Processing Systems*, pages 597–607, 2017.

[7] Moritz Hardt and Tengyu Ma. Identity matters in deep learning. 2017.

[8] Avrim Blum and Ronald L Rivest. Training a 3-node neural network is np-complete. In *Advances in Neural Information Processing Systems*, pages 494–501, 1989.

[9] Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. Provable bounds for learning some deep representations. In *International Conference on Machine Learning*, pages 584–592, 2014.

[10] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. Why are deep nets reversible: A simple theory, with implications for training. *arXiv preprint arXiv:1511.05653*, 2015.

[11] Akshay Rangamani, Anirbit Mukherjee, Ashish Arora, Tejaswini Ganapathy, Amitabh Basu, Sang Chin, and Trac D Tran. Sparse coding and autoencoders. *arXiv preprint arXiv:1708.03735*, 2017.

[12] Sanjeev Arora and Ravi Kannan. Learning mixtures of separated nonspherical gaussians. *The Annals of Applied Probability*, 15(1A):69–92, 2005.

[13] Daniel A Spielman, Huan Wang, and John Wright. Exact recovery of sparsely-used dictionaries. In *Conference on Learning Theory*, pages 37–1, 2012.

[14] Alekh Agarwal, Animashree Anandkumar, Prateek Jain, Praneeth Netrapalli, and Rashish Tandon. Learning sparsely used overcomplete dictionaries. In *Conference on Learning Theory*, pages 123–137, 2014.

[15] Rémi Gribonval, Rodolphe Jenatton, Francis Bach, Martin Kleinsteuber, and Matthias Seibert. Sample complexity of dictionary learning and other matrix factorizations. *IEEE Transactions on Information Theory*, 61(6):3469–3486, 2015.

[16] Sanjeev Arora, Rong Ge, Tengyu Ma, and Ankur Moitra. Simple, efficient, and neural algorithms for sparse coding. In *Conference on Learning Theory*, pages 113–149, 2015.

[17] Ankur Moitra and Gregory Valiant. Settling the polynomial learnability of mixtures of gaussians. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 93–102. IEEE, 2010.

[18] Sanjeev Arora, Rong Ge, Ankur Moitra, and Sushant Sachdeva. Provable ica with unknown gaussian noise, with implications for gaussian mixtures and autoencoders. In *Advances in Neural Information Processing Systems*, pages 2375–2383, 2012.

[19] Navin Goyal, Santosh Vempala, and Ying Xiao. Fourier pca and robust tensor decomposition. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 584–593. ACM, 2014.

[20] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[21] Devansh Arpit, Yingbo Zhou, Hung Ngo, and Venu Govindaraju. Why regularized auto-encoders learn sparse representation? *arXiv preprint arXiv:1505.05561*, 2015.

[22] Alekh Agarwal, Animashree Anandkumar, and Praneeth Netrapalli. Exact recovery of sparsely used overcomplete dictionaries. *stat*, 1050:8, 2013.

[23] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.

[24] Sanjeev Arora, Rong Ge, and Ankur Moitra. New algorithms for learning incoherent and overcom-

plete dictionaries. In *Conference on Learning Theory*, pages 779–806, 2014.

[25] Adam Coates and Andrew Y Ng. The importance of encoding versus training with sparse coding and vector quantization. In *International Conference on Machine Learning*, pages 921–928, 2011.

[26] Andrew M Saxe, Pang Wei Koh, Zhenghao Chen, Maneesh Bhand, Bipin Suresh, and Andrew Y Ng. On random weights and unsupervised feature learning. In *International Conference on Machine Learning*, pages 1089–1096, 2011.

[27] Kishore Konda, Roland Memisevic, and David Krueger. Zero-bias autoencoders and the benefits of co-adapting features. *arXiv preprint arXiv:1402.3337, published in ICLR 2015*, 2015.

[28] Thanh V Nguyen, Raymond K W Wong, and Chinmay Hegde. A provable approach for double-sparse coding. In *Proc. Conf. American Assoc. Artificial Intelligence*, Feb. 2018.