Optimal Sparse Linear Encoders and Sparse PCA

Malik Magdon-Ismail
Rensselaer Polytechnic Institute, Troy, NY 12211
magdon@cs.rpi.edu

Christos Boutsidis
New York, NY
christos.boutsidis@gmail.com

Abstract

Principal components analysis (PCA) is the optimal linear encoder of data. Sparse linear encoders (e.g., sparse PCA) produce more interpretable features that can promote better generalization. (i) Given a level of sparsity, what is the best approximation to PCA? (ii) Are there efficient algorithms which can achieve this optimal combinatorial tradeoff? We answer both questions by providing the first polynomial-time algorithms to construct *optimal* sparse linear auto-encoders; additionally, we demonstrate the performance of our algorithms on real data.

1 Introduction

The data matrix is $\mathrm{X} \in \mathbb{R}^{n\times d}$ (a row $\mathbf{x}_i^{\mathrm{T}} \in \mathbb{R}^{1\times d}$ is a data point in d dimensions). auto-encoders transform (encode) the data into a low dimensional feature space and then lift (decode) it back to the original space, reconstructing the data through a bottleneck. If the reconstruction is close to the original, then the encoder preserved most of the information using just a small number of features. Auto-encoders are important in machine learning because they perform information preserving dimension reduction. Our focus is the *linear* auto-encoder, which, for $k < d$, is a pair of linear mappings $h : \mathbb{R}^d \mapsto \mathbb{R}^k$ and $g : \mathbb{R}^k \mapsto \mathbb{R}^d$, specified by an encoder matrix $\mathrm{H} \in \mathbb{R}^{d\times k}$ and a decoder matrix $\mathrm{G} \in \mathbb{R}^{k\times d}$. For data point $\mathbf{x} \in \mathbb{R}^d$, the encoded feature is $\mathbf{z} = h(\mathbf{x}) = \mathrm{H}^{\mathrm{T}}\mathbf{x} \in \mathbb{R}^k$ and the reconstructed datum is $\hat{\mathbf{x}} = g(\mathbf{z}) = \mathrm{G}^{\mathrm{T}}\mathbf{z} \in \mathbb{R}^d$. The reconstructed data matrix is $\hat{\mathrm{X}} = \mathrm{XHG}$. The pair (H, G) is a good auto-encoder if $\hat{\mathrm{X}} \approx \mathrm{X}$ under some loss metric (we use squared loss):

Definition 1 (Loss $\ell(\mathrm{H}, \mathrm{X})$)**.** *The loss of encoder* H *is the minimum possible Frobenius reconstruction error (over all linear decoders* G*) when using* H *as encoder for* X*:*

$$\ell(\mathrm{H}, \mathrm{X}) = \min_{\mathrm{G}\in\mathbb{R}^{k\times d}} \|\mathrm{X} - \mathrm{XHG}\|_{\mathrm{F}}^2 = \|\mathrm{X} - \mathrm{XH}(\mathrm{XH})^{\dagger}\mathrm{X}\|_{\mathrm{F}}^2.$$

The loss is defined for an encoder H alone, by choosing the decoder optimally. The literature considers primarily the symmetric auto-encoder which places the additional restriction that $\mathrm{G} = \mathrm{H}^{\dagger}$ [18]. *To get the most useful features, one should not place unnecessary constraints on the decoder.* Principal Component Analysis (PCA) is the most famous linear auto-encoder, because it is optimal (and symmetric). Since $\mathrm{rank}(\mathrm{XHG}) \leq k$, the loss is bounded by $\ell(\mathrm{H}, \mathrm{X}) \geq \|\mathrm{X} - \mathrm{X}_k\|_{\mathrm{F}}^2$ (X_k is its best rank-k approximation to X). By the Eckart-Young theorem $\mathrm{X}_k = \mathrm{XV}_k\mathrm{V}_k^{\mathrm{T}}$, where $\mathrm{V}_k \in \mathbb{R}^{d\times k}$ is the matrix whose columns are the top-k right singular vectors of X (see, for example, e-Chapter 9 of [14]). Thus, the *optimal* linear encoder is $\mathrm{H}_{opt} = \mathrm{V}_k$, and the top-$k$ PCA-features are $\mathrm{Z}_{\mathrm{pca}} = \mathrm{XV}_k$. Since its early beginings, [19], PCA has evolved into a classic tool for data analysis. While PCA simplifies the data by concentrating the maximum information into a few components, those components may be hard to interpret. In many applications, it is desirable to "explain" the features using a few original variables (for example, genes in a biological application or assets in a financial application). One trades off the *fidelity* of the features (their ability to reconstruct the data) with the *interpretability* of the features using a few original features (sparsity of the encoder H). We introduce a sparsity parameter r and require that every column of H have at most r non-zero elements. *Every* feature in an r-sparse encoding can be "explained" using at most r original features. We now formally state the *sparse linear encoder problem*:

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

Problem 1: Optimal r-sparse encoder (Sparse PCA)

Given $\mathrm{X} \in \mathbb{R}^{n\times d}$, $\varepsilon > 0$ and $k < \mathrm{rank}(\mathrm{A})$, find an r-sparse encoder H with minimum r, for which the loss is a $(1+\epsilon)$ *relative approximation* to the optimal loss,

$$\ell(\mathrm{H},\mathrm{X}) = \|\mathrm{X} - \mathrm{XH}(\mathrm{XH})^{\dagger}\mathrm{X}\|_{\mathrm{F}}^2 \le (1+\varepsilon)\|\mathrm{X}-\mathrm{X}_k\|_{\mathrm{F}}^2.$$

Our Contributions. First, we are proposing the "sparse-PCA" problem defined above in lieu of traditional sparse-PCA which is based on maximizing variance, not minimizing loss. With no sparsity constraint, variance maximization and loss minimization are equivalent, both being solved by PCA. Historically, variance maximization became the norm for sparse-PCA. However, minimizing loss better achieves the machine learning goal of preserving information. The table below compares the 10-fold cross-validation error E_{out} for an SVM classifier using features from popular variance maximizing sparse-PCA encoders and our loss minimizing sparse-encoder ($k = 6$ and $r = 7$),

	d	SVM	T-Power [23]			G-Power-ℓ_0 [10]			G-Power-ℓ_1 [10]			Our Algorithm		
		E_{out}	E_{out}	Loss	Var	E_{out}	Loss	Var	E_{out}	Loss	Var	E_{out}	Loss	Var
ARCENE	10^4	0.44	matrix to large			0.325	2.5	0.01	0.35	2.5	0.01	**0.29**	**1.4**	0.005
Gisette	5000	0.44	0.45	1.17	0.1	0.49	1.2	0.02	0.50	1.2	0.02	**0.31**	**1.1**	0.02
Madelon	500	0.51	0.46	1.3	0.09	0.33	1.08	0.08	0.46	1.33	0.05	**0.24**	**1.07**	0.03
"1" vs "5"	256	0.35	0.30	2.4	0.21	0.34	2.28	0.27	0.33	2.3	0.19	**0.01**	**1.2**	0.03
SECOM	691	0.07	0.34	1.3	0.96	0.35	2.9	0.79	0.33	2.9	0.79	**0.31**	**1.0**	0.46
Spam	57	0.17	**0.20**	**1.00**	1.0	0.22	1.03	1.0	0.20	1.02	1.0	0.21	1.02	1.0

Lower loss gives lower error. Our experiments are not exhaustive, but their role is modest: to motivate minimizing loss as the right machine learning objective for sparse encoders (Problem 1).

Our main contribution is polynomial sparse encoder algorithms with *theoretical guarantees* that solve Problem 1. We give a $(1+\epsilon)$-optimal encoder with sparsity $O(k/\varepsilon)$ (Theorem 7). This sparsity cannot be beaten by *any* algorithm that guarantees a $(1+\varepsilon)$-approximation (Theorem 8). Ours is the first theoretical guarantee for a k-component sparse linear encoder with respect to the optimal PCA. Our algorithm constructs sparse PCA features (columns of the encoder H) which preserve almost as much information as optimal PCA-features. Our second technical contribution (Theorem 11) is an algorithm to construct sparse features *iteratively* (typical of sparse-PCA algorithms). Iterative algorithms are notoriously hard to analyze, and we give the first theoretical guarantees for an iterative sparse encoder. (Detailed proofs are postponed to a full version.)

Notation. Let $\rho \le \min\{n,d\} = \mathrm{rank}(\mathrm{X})$ (typically $\rho = d$). We use $\mathrm{A},\mathrm{B},\mathrm{C},\ldots$ for matrices and $\mathbf{a},\mathbf{b},\mathbf{c},\ldots$ for vectors. The Euclidean basis is $\mathbf{e}_1,\mathbf{e}_2,\ldots$ (dimension can inferred from context). For an $n\times d$ matrix X, the singular value decomposition (SVD) gives $\mathrm{X} = \mathrm{U}\Sigma\mathrm{V}^{\mathrm{T}}$, where the columns of $\mathrm{U}\in\mathbb{R}^{n\times\rho}$ are the left singular vectors, the columns of $\mathrm{V}\in\mathbb{R}^{d\times\rho}$ are the right singular vectors, and $\Sigma\in\mathbb{R}^{\rho\times\rho}$ is the positive diagonal matrix of singular values $\sigma_1 \ge \cdots \ge \sigma_\rho$; U and V are orthonormal, $\mathrm{U}^{\mathrm{T}}\mathrm{U} = \mathrm{V}^{\mathrm{T}}\mathrm{V} = \mathrm{I}_\rho$. For integer k, we use $\mathrm{U}_k\in\mathbb{R}^{n\times k}$ (resp. $\mathrm{V}_k\in\mathbb{R}^{d\times k}$) for the first k left (resp. right) singular vectors, and $\Sigma_k\in\mathbb{R}^{k\times k}$ is the diagonal matrix with the top-k singular values. We can view a matrix as a row of columns. So, $\mathrm{X} = [\mathbf{f}_1,\ldots,\mathbf{f}_d]$, $\mathrm{U} = [\mathbf{u}_1,\ldots,\mathbf{u}_\rho]$, $\mathrm{V} = [\mathbf{v}_1,\ldots,\mathbf{v}_\rho]$, $\mathrm{U}_k = [\mathbf{u}_1,\ldots,\mathbf{u}_k]$ and $\mathrm{V}_k = [\mathbf{v}_1,\ldots,\mathbf{v}_k]$. We use $\mathbf{f}$ for the columns of X, the *features*, and we reserve $\mathbf{x}_i$ for the data points (rows of X), $\mathrm{X}^{\mathrm{T}} = [\mathbf{x}_1,\ldots,\mathbf{x}_n]$. $\mathrm{A} = [\mathbf{a}_1,\ldots,\mathbf{a}_k]$ is $(r_1,\ldots,r_k)$-sparse if $\|\mathbf{a}_i\|_0 \le r_i$; if all r_i are equal to r, we say the matrix is r-sparse. The Frobenius (Euclidean) norm of a matrix A is $\|\mathrm{A}\|_{\mathrm{F}}^2 = \sum_{ij}\mathrm{A}_{ij}^2 = \mathrm{Tr}(\mathrm{A}^{\mathrm{T}}\mathrm{A}) = \mathrm{Tr}(\mathrm{A}\mathrm{A}^{\mathrm{T}})$. The pseudo-inverse $\mathrm{A}^{\dagger}$ of A with SVD $\mathrm{U}_\mathrm{A}\Sigma_\mathrm{A}\mathrm{V}_\mathrm{A}^{\mathrm{T}}$ is $\mathrm{A}^{\dagger} = \mathrm{V}_\mathrm{A}\Sigma_\mathrm{A}^{-1}\mathrm{U}_\mathrm{A}^{\mathrm{T}}$; $\mathrm{A}\mathrm{A}^{\dagger} = \mathrm{U}_\mathrm{A}\mathrm{U}_\mathrm{A}^{\mathrm{T}}$ is a symmetric projection operator. For matrices A,B with $\mathrm{A}^{\mathrm{T}}\mathrm{B} = \mathbf{0}$, a generalized Pythagoras theorem holds, $\|\mathrm{A}+\mathrm{B}\|_{\mathrm{F}}^2 = \|\mathrm{A}\|_{\mathrm{F}}^2 + \|\mathrm{B}\|_{\mathrm{F}}^2$. $\|\mathrm{A}\|_2$ is the operator norm (top singular value) of A.

Discussion of Related Work. PCA is the optimal (and most popular) linear auto-encoder. Nonlinear auto-encoders became prominent with auto-associative neural networks [7, 3, 4, 17, 18]. There is some work on sparse linear auto-encoders (e.g. [15]) and a lot of research on "sparse PCA". The importance of sparse factors in dimensionality reduction has been recognized in some early work: the *varimax* criterion [11] has been used to rotate the factors to encourage sparsity, and this has been used in multi-dimensional scaling approaches to dimension reduction [20, 12]. One of the first attempts at sparse PCA used axis rotations and component thresholding [6]. The traditional formulation of sparse PCA is as cardinality constrained variance maximization: $\max_{\mathbf{v}}\mathbf{v}^{\mathrm{T}}\mathrm{A}\mathbf{v}$ subject to $\mathbf{v}^{\mathrm{T}}\mathbf{v} = 1$ and $\|\mathbf{v}\|_0 \le r$, which is NP-hard [14]. The exhaustive algorithm requires $O\big(dr^2\binom{d}{r}\big)$

computation which can be improved to $O(d^{q+1})$ for a rank-q perturbation of the identity [2]. These algorithms are not practical. Several heuristics exist. [22] and [24] take an L_1 penalization view. DSPCA (direct sparse PCA) [9] also uses an L_1 sparsifier but solves a relaxed convex semidefinite program which is further refined in [8] where they also give a tractable sufficient condition for testing optimality. The simplest algorithms use greedy forward and backward subset selection. For example, [16] develop a greedy branch and bound algorithm based on spectral bounds with $O(d^3)$ running time for forward selection and $O(d^4)$ running time for backward selection. An alternative view of the problem is as a sparse matrix reconstruction problem; for example [21] obtain sparse principal components using regularized low-rank matrix approximation. Most existing SPCA algorithms find one sparse principal component. One applies the algorithm iteratively on the residual after projection to get additional sparse principal components [13].

There are no polynomial algorithms with optimality guarantees. [1] considers sparse PCA with a non-negativity constraint: they give an algorithm with input parameter k and running time $O(d^{k+1}\log d + d^k r^3)$ which constructs a sparse component within $(1 - \frac{n}{r}\|\mathrm{X} - \mathrm{X}_k\|_2/\|\mathrm{X}\|_2)$ from optimal. The running time is not practical when k is large; further, the approximation guarantee relies on rapid spectral decay of X and only applies to the first component, not to further iterates.

Explained Variance vs. Loss. For symmetric auto-encoders, minimizing loss is equivalent to maximizing the *symmetric explained variance* $\|\mathrm{XHH}^\dagger\|_{\mathrm{F}}^2$ due to the identity

$$var(\mathrm{X}) = \|\mathrm{X}\|_{\mathrm{F}}^2 = \|\mathrm{X}(\mathrm{I} - \mathrm{HH}^\dagger) + \mathrm{XHH}^\dagger\|_{\mathrm{F}}^2 = \|\mathrm{X} - \mathrm{XHH}^\dagger\|_{\mathrm{F}}^2 + \|\mathrm{XHH}^\dagger\|_{\mathrm{F}}^2$$

(the last equality is from Pythagoras' theorem). The PCA auto-encoder is symmetric, $\mathrm{V}_k^\dagger = \mathrm{V}_k^{\mathrm{T}}$. So the optimal encoder for maximum variance or minimum loss are the same: PCA. But, when it comes to approximation, an approximation algorithm for loss can be converted to an approximation algorithm for variance maximization (the reverse is not true).

Theorem 2. *If* $\|\mathrm{X} - \mathrm{XHH}^\dagger\|_{\mathrm{F}}^2 \leq (1+\varepsilon)\|\mathrm{X} - \mathrm{X}_k\|_{\mathrm{F}}^2$, *then* $\|\mathrm{XHH}^\dagger\|_{\mathrm{F}}^2 \geq \left(1 - \frac{\rho-k}{k}\varepsilon\right)\|\mathrm{X}_k\|_F^2$.

When factors are not decorrelated, explained variance is not well defined [24], whereas loss is well defined for any encoder. Minimizing loss and maximizing the explained variance are both ways of encouraging H to be close to V_k. However, when H is constrained (for example to be sparse), these optimization objectives can produce very different solutions. From the machine learning perspective, symmetry is an unnecessary constaint on the auto-encoder. All we want is an encoder that produces a compact representation of the data while capturing as much information as possible.

2 Optimal Sparse Linear Encoder

We show a black-box reduction of sparse encoding to the column subset selection problem (CSSP). We then use column subset selection algorithms to construct provably accurate sparse auto-encoders. For $\mathrm{X} = [\mathbf{f}_1, \ldots, \mathbf{f}_d]$, we let $\mathrm{C} = [\mathbf{f}_{i_1}, \mathbf{f}_{i_2} \ldots, \mathbf{f}_{i_r}]$ denote a matrix formed using r columns "sampled" from X, where $1 \leq i_1 < i_2 \cdots < i_r \leq d$ are distinct column indices. We can use a matrix $\Omega \in \mathbb{R}^{d\times r}$ to perform the sampling, $\mathrm{C} = \mathrm{X}\Omega$, where $\Omega = [\mathbf{e}_{i_1}, \mathbf{e}_{i_2} \ldots, \mathbf{e}_{i_r}]$ and $\mathbf{e}_i$ are the standard basis vectors in $\mathbb{R}^d$ (post-multiplying X by $\mathbf{e}_i$ "samples" the ith column of X). The columns of C span a subspace in the range of X. A sampling matrix can be used to construct an r-sparse matrix.

Lemma 3. *Let* $\Omega = [\mathbf{e}_{i_1}, \mathbf{e}_{i_2} \ldots, \mathbf{e}_{i_r}]$ *and let* $\mathrm{A} \in \mathbb{R}^{r\times k}$ *be any matrix. Then* $\Omega\mathrm{A}$ *is* r*-sparse.*

Define $\mathrm{X_C} = \mathrm{CC}^\dagger\mathrm{X}$, the projection of X onto the column space of C. Let $\mathrm{X}_{\mathrm{C},k} \in \mathbb{R}^{n\times d}$ be the optimal rank-k approximation to $\mathrm{X_C}$ obtained via the SVD of $\mathrm{X_C}$.

Lemma 4 (See, for example, [5])**.** $\mathrm{X}_{\mathrm{C},k}$ *is a rank-*k *matrix whose columns are in the span of* C. *Let* $\hat{\mathrm{X}}$ *be any rank-*k *matrix whose columns are in the span of* C. *Then,* $\|\mathrm{X} - \mathrm{X}_{\mathrm{C},k}\|_{\mathrm{F}} \leq \|\mathrm{X} - \hat{\mathrm{X}}\|_{\mathrm{F}}$.

That is, $\mathrm{X}_{\mathrm{C},k}$ is the best rank-k approximation to X whose columns are in the span of C. An efficient algorithm to compute $\mathrm{X}_{\mathrm{C},k}$ is also given in [5]. The algorithm runs in $O(ndr + (n+d)r^2)$ time.

2.1 Sparse Linear Encoders from Column Subset Selection

We show that a set of columns C for which $\mathrm{X}_{\mathrm{C},k}$ is a good approximation to X can produce a good sparse linear encoder. In the algorithm below we assume (not essential) that C has full column rank. The algorithm uses standard linear algebra operations and has total runtime in $O(ndr + (n+d)r^2)$.

Blackbox algorithm to compute encoder from CSSP

Inputs: $\mathrm{X} \in \mathbb{R}^{n\times d}$; $\mathrm{C} \in \mathbb{R}^{n\times r}$ with $\mathrm{C} = \mathrm{X}\Omega$ and $\Omega = [\mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_r}]$; $k \leq r$.
Output: r-sparse linear encoder $\mathrm{H} \in \mathbb{R}^{d\times k}$.

1: Compute a QR-factorization of C as $\mathrm{C} = \mathrm{QR}$, with $\mathrm{Q} \in \mathbb{R}^{n\times r}$, $\mathrm{R} \in \mathbb{R}^{r\times r}$.
2: Obtain the SVD of $\mathrm{R}^{-1}(\mathrm{Q}^{\mathrm{T}}\mathrm{X})_k$, $\mathrm{R}^{-1}(\mathrm{Q}^{\mathrm{T}}\mathrm{X})_k = \mathrm{U}_{\mathrm{R}}\Sigma_{\mathrm{R}}\mathrm{V}_{\mathrm{R}}^{\mathrm{T}}$.
($\mathrm{U}_{\mathrm{R}} \in \mathbb{R}^{r\times k}$, $\Sigma_{\mathrm{R}} \in \mathbb{R}^{k\times k}$ and $\mathrm{V}_{\mathrm{R}} \in \mathbb{R}^{d\times k}$)
3: Return $\mathrm{H} = \Omega\mathrm{U}_{\mathrm{R}} \in \mathbb{R}^{d\times k}$.

In step 2, even though $\mathrm{R}^{-1}(\mathrm{Q}^{\mathrm{T}}\mathrm{X})_k$ is an $r \times d$ matrix, it has rank k, hence the dimensions of $\mathrm{U}_{\mathrm{R}},\ \Sigma_{\mathrm{R}},\ \mathrm{V}_{\mathrm{R}}$ depend on k, not r. By Lemma 3, the encoder H is r-sparse. Also, H has orthonormal columns, as is typically desired for an encoder ($\mathrm{H}^{\mathrm{T}}\mathrm{H} = \mathrm{U}_{\mathrm{R}}^{\mathrm{T}}\Omega^{\mathrm{T}}\Omega\mathrm{U}_{\mathrm{R}} = \mathrm{U}_{\mathrm{R}}^{\mathrm{T}}\mathrm{U}_{\mathrm{R}} = \mathrm{I}$). In every column of our encoder, the non-zeros are at the *same* r coordinates which is much stronger than r-sparsity. The next theorem shows that our encoder is good if C contains a good rank-k approximation $\mathrm{X}_{\mathrm{C},k}$.

Theorem 5 (Blackbox encoder from CSSP)**.** *Given* $\mathrm{X} \in \mathbb{R}^{n\times d}$, $\mathrm{C} = \mathrm{X}\Omega \in \mathbb{R}^{n\times r}$ *with* $\Omega = [\mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_r}]$ *and* $k \leq r$*, let* H *be the* r*-sparse linear encoder produced by the algorithm above in* $O(ndr + (n+d)r^2)$ *time. Then, the loss satisfies*

$$\ell(\mathrm{H}, \mathrm{X}) = \|\mathrm{X} - \mathrm{XH}(\mathrm{XH})^{\dagger}\mathrm{X}\|_{\mathrm{F}}^2 \leq \|\mathrm{X} - \mathrm{X}_{\mathrm{C},k}\|_{\mathrm{F}}^2.$$

The theorem says that if we can find a set of r columns within which a good rank-k approximation to X exists, then we can construct a good sparse linear encoder. What remains is to find a sampling matrix Ω which gives a good set of columns $\mathrm{C} = \mathrm{X}\Omega$ for which $\|\mathrm{X} - \mathrm{X}_{\mathrm{C},k}\|_{\mathrm{F}}^2$ is small. The main tool to obtain C and Ω was developed in [5] which gave a constant factor deterministic approximation algorithm and a relative-error randomized approximation algorithm. We state a simplified form of the result and then discuss various ways in which this result can be enhanced. *Any algorithm to construct a good set of columns can be used as a black box to get a sparse linear encoder.*

Theorem 6 (Near-optimal CSSP [5])**.** *Given* $\mathrm{X} \in \mathbb{R}^{n\times d}$ *of rank* ρ *and target rank* k*:*

(i) (Theorem 2 in [5]) For sparsity parameter $r > k$*, there is a deterministic algorithm which runs in time* $T_{V_k} + O(ndk + dk^3)$ *to construct a sampling matrix* $\Omega = [\mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_r}]$ *and corresponding columns* $\mathrm{C} = \mathrm{X}\Omega$ *such that*

$$\|\mathrm{X} - \mathrm{X}_{\mathrm{C},k}\|_{\mathrm{F}}^2 \leq \left(1 + (1 - \sqrt{k/r})^{-2}\right)\|\mathrm{X} - \mathrm{X}_k\|_{\mathrm{F}}^2.$$

(ii) (Simplified Theorem 5 in [5]) For sparsity parameter $r > 5k$*, there is a randomized algorithm which runs in time* $O(ndk + dk^3 + r\log r)$ *to construct a sampling matrix* $\Omega = [\mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_r}]$ *and corresponding columns* $\mathrm{C} = \mathrm{X}\Omega$ *such that*

$$\mathbb{E}\left[\|\mathrm{X} - \mathrm{X}_{\mathrm{C},k}\|_{\mathrm{F}}^2\right] \leq \left(1 + \frac{5k}{r - 5k}\right)\|\mathrm{X} - \mathrm{X}_k\|_{\mathrm{F}}^2.$$

Our "batch" sparse linear encoder uses Theorem 6 in our black-box CSSP-encoder.

"Batch" Sparse Linear Encoder Algorithm

Inputs: $\mathrm{X} \in \mathbb{R}^{n\times d}$; rank $k \leq \mathrm{rank}(\mathrm{X})$; sparsity $r > k$.
Output: r-sparse linear encoder $\mathrm{H} \in \mathbb{R}^{d\times k}$.

1: Use Theorem 6-(ii) to compute columns $\mathrm{C} = \mathrm{X}\Omega \in \mathbb{R}^{n\times r}$, with inputs $\mathrm{X},\ k,\ r$.
2: Return H computed with $\mathrm{X},\ \mathrm{C},\ k$ as input to the CSSP-blackbox encoder algorithm.

Using Theorem 6 in Theorem 5, we have an approximation guarantee for our algorithm.

Theorem 7 (Sparse Linear Encoder)**.** *Given* $\mathrm{X} \in \mathbb{R}^{n\times d}$ *of rank* ρ*, the target number of sparse PCA vectors* $k \leq \rho$*, and sparsity parameter* $r > 5k$*, the "batch" sparse linear encoder algorithm above runs in time* $O(ndr + (n+d)r^2 + dk^3)$ *and constructs an* r*-sparse encoder* H *such that:*

$$\mathbb{E}\left[\|\mathrm{X} - \mathrm{XH}(\mathrm{XH})^{\dagger}\mathrm{X}\|_{\mathrm{F}}^2\right] \leq \left(1 + \frac{5k}{r - 5k}\right)\|\mathrm{X} - \mathrm{X}_k\|_{\mathrm{F}}^2.$$

Comments. The expectation is over the random choices in the algorithm, and the bound can be boosted to hold with high probability or even deterministically. 2. The guarantee is with respect to $\|\mathrm{X} - \mathrm{X}_k\|_{\mathrm{F}}^2$ (optimal dense PCA): sparsity $r = O(k/\varepsilon)$ suffices to mimic top-k (dense) PCA.

We now give the lower bound on sparsity, showing that our result is worst-case optimal. Define the *row-sparsity* of H as the number of its rows that are non-zero. When k=1, the row-sparsity equals the sparsity of the single factor. The row-sparsity is the total number of dimensions which have non-zero loadings among *all* the factors. Our algorithm produces an encoder with row-sparsity $O(k/\varepsilon)$ and comes within $(1+\varepsilon)$ of the minimum possible loss. This is worst case optimal:

Theorem 8 (Lower Bound)**.** *There is a matrix* X *for which any linear encoder that achieves a* $(1+\varepsilon)$*-approximate loss as compared to PCA must have a row-sparsity* $r \geq k/\varepsilon$*.*

The common case that in the literature is with $k = 1$ (top sparse component). Our lower bound shows that $\Omega(1/\varepsilon)$-sparsity is required and our algorithm asymptotically achieves this lower bound. To prove Theorem 8, we show the converse of Theorem 5: a good linear auto-encoder with row-sparsity r can be used to construct r columns C for which $\mathrm{X}_{\mathrm{C},k}$ approximates X.

Lemma 9. *Suppose* H *is a linear encoder for* X *with row-sparsity* r *and decoder* G*. Then,* $\mathrm{BHG} = \mathrm{CY}$*, where* C *is a set of* r *columns of* X *and* $\mathrm{Y} \in \mathbb{R}^{r\times d}$*.*

Given Lemma 9, a sketch of the rest of the argument is as follows. Section 9.2 of [5] demonstrates a matrix for which there do not exist r good columns. Since a good r-sparse encoder gives r good columns, no r-sparse encoder can be $(1+k/r)$-optimal: No linear encoder with row-sparsity r achieves a loss within $(1+k/r)$ of PCA. Our construction is asymptotically worst case optimal. The lower bound holds for general linear auto-encoders, and so this lower bound also applies to the symmetric auto-encoder $\mathrm{HH}^\dagger$, the traditional formulation of sparse PCA. When $k = 1$, for any r-sparse unit norm $\mathbf{v}$, there exists X for which $\|\mathrm{X} - \mathrm{X}\mathbf{v}\mathbf{v}^\mathrm{T}\|_\mathrm{F}^2 \geq (1+\frac{1}{r})\|\mathrm{X} - \mathrm{X}_1\|_\mathrm{F}^2$, or in terms of the symmetric explained variance $\mathbf{v}^\mathrm{T}\mathrm{B}^\mathrm{T}\mathrm{B}\mathbf{v} \leq \|\mathrm{B}_1\|_\mathrm{F}^2 - \frac{1}{r}\|\mathrm{B} - \mathrm{B}_1\|_\mathrm{F}^2$.

3 Iterative Sparse Linear Encoders

Our CSSP-based algorithm is "batch" in that all k factors are constructed simultaneously. Every feature in the encoder is r-sparse with non-zero loadings on the *same* set of r original dimensions; and, you cannot do better with a row-sparsity of r. Further, the batch algorithm does not distinguish between the k-factors. That is, there is no top component, second component, and so on.

The traditional techniques for sparse PCA construct the factors iteratively. We can too: run our batch algorithm in an iterative mode, where in each step we set $k = 1$ and compute a sparse factor for a residual matrix. By constructing our k features iteratively (and adaptively), we identify an ordering among the k features. Further, we might be able to get each feature sparser while still maintaining a bound on the row-sparsity. We now give an iterative version of our algorithm. In each iteration, we augment H by computing a top sparse encoder for the residual obtained using the current H.

Iterative Sparse Linear Encoder Algorithm

Inputs: $\mathrm{X} \in \mathbb{R}^{n\times d}$; rank $k \leq \mathrm{rank}(\mathrm{X})$; sparsity parameters $r_1, \ldots, r_k$.
Output: $(r_1, \ldots, r_k)$-sparse linear encoder

1: Set the residual $\Delta = \mathrm{X}$ and $\mathrm{H} = [\,]$.
2: **for** $i = 1$ to k **do**
3: Use the batch algorithm to compute encoder $\mathbf{h}$ for Δ, with $k = 1$ and $r = r_i$.
4: Add $\mathbf{h}$ to the encoder: $\mathrm{H} \leftarrow [\mathrm{H}, \mathbf{h}]$.
5: Update the residual Δ: $\Delta \leftarrow \mathrm{X} - \mathrm{XH}(\mathrm{XH})^\dagger\mathrm{X}$.
6: Return the $(r_1, \ldots, r_k)$-sparse encoder $\mathrm{H} \in \mathbb{R}^{n\times k}$.

The next lemma bounds the reconstruction error for this iterative step in the algorithm.

Lemma 10. *Suppose, for* $k \geq 1$*,* $\mathrm{H}_k = [\mathbf{h}_1, \ldots, \mathbf{h}_k]$ *is an encoder for* X*, satisfying*

$$\|\mathrm{X} - \mathrm{XH}_k(\mathrm{XH}_k)^\dagger\mathrm{X}\|_\mathrm{F}^2 = \mathsf{err}.$$

Given a sparsity $r > 5$ *and* $\delta \leq 5/(r-5)$*, one can compute in time* $O(ndr + (n+d)r^2)$ *an* r*-sparse feature* $\mathbf{h}_{k+1}$ *such that the reconstruction error of the encoder* $\mathrm{H}_{k+1} = [\mathbf{h}_1, \ldots, \mathbf{h}_k, \mathbf{h}_{k+1}]$ *satisfies*

$$\mathbb{E}\left[\|\mathrm{X} - \mathrm{XH}_{k+1}(\mathrm{XH}_{k+1})^\dagger\mathrm{X}\|_\mathrm{F}^2\right] \quad = \quad (1+\delta)(\mathsf{err} - \|\mathrm{X} - \mathrm{XH}_k(\mathrm{XH}_k)^\dagger\mathrm{X}\|_2^2).$$

Lemma 10 gives a bound on the reconstruction error for an iterative addition of the next sparse encoder vector. To see how Lemma 10 is useful, consider target rank $k = 2$. First construct $\mathbf{h}_1$ with sparsity $r_1 = 5 + 5/\varepsilon$, which gives $(1+\varepsilon)\|\mathrm{X} - \mathrm{X}_1\|_{\mathrm{F}}^2$ loss. Now construct $\mathbf{h}_2$, also with sparsity $r_2 = 5 + 5/\varepsilon$. The loss for $\mathrm{H} = [\mathbf{h}_1, \mathbf{h}_2]$ is bounded by

$$\ell(\mathrm{H}, \mathrm{X}) \leq (1+\varepsilon)^2\|\mathrm{X} - \mathrm{X}_2\|_{\mathrm{F}}^2 + \varepsilon(1+\varepsilon)\|\mathrm{X} - \mathrm{X}_1\|_2^2.$$

On the other hand, our batch algorithm uses sparsity $r = 10 + 10/\varepsilon$ in each encoder $\mathbf{h}_1, \mathbf{h}_2$ and achieves reconstruction error $(1+\varepsilon)\|\mathrm{X} - \mathrm{X}_2\|_{\mathrm{F}}^2$. The iterative algorithm uses sparser features, but pays for it a little in reconstruction error. The second term is small, $O(\varepsilon)$, and depends on $\|\mathrm{X} - \mathrm{X}_1\|_2^2 = \sigma_2^2$, which in practice is smaller than $\|\mathrm{X} - \mathrm{X}_2\|_{\mathrm{F}}^2 = \sigma_3^2 + \cdots + \sigma_d^2$. Using the iterative algorithm, we can tailor the sparsity of each encoder vector separately to achieve a desired accuracy. It is algebraically intense to prove a bound for a general choice of the sparsity parameters $r_1, \ldots, r_k$, so for simplicity, we prove a bound for a specific choice of the sparsity parameters which slowly increase with each iterate. The proof idea is similar to our example with $k = 2$.

Theorem 11 (Iterative Encoder)**.** *Given $\mathrm{X} \in \mathbb{R}^{n\times d}$ of rank ρ and $k < \rho$, set $r_j = 5 + \lceil 5j/\varepsilon \rceil$ in our iterative encoder to compute the $(r_1, \ldots, r_k)$-sparse encoder $\mathrm{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k]$. Then, for every $\ell = 1, \ldots, k$, the encoder $\mathrm{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k]$ has reconstruction error*

$$\mathbb{E}\left[\|\mathrm{X} - \mathrm{X}\mathrm{H}_\ell(\mathrm{X}\mathrm{H}_\ell)^\dagger\mathrm{X}\|_{\mathrm{F}}^2\right] \qu\leq \quad (e\ell)^\varepsilon\|\mathrm{X} - \mathrm{X}_\ell\|_{\mathrm{F}}^2 + \varepsilon\ell^{1+\varepsilon}\|\mathrm{X}_\ell - \mathrm{X}_1\|_{\mathrm{F}}^2. \tag{1}$$

The running time to compute all the encoder vectors is $O(ndk^2\varepsilon^{-1} + (n+d)k^3\varepsilon^{-2})$.

Comments. This is the first theoretical guarantee for iterative sparse encoders. Up to a small additive term, we have a relative error approximation because $(e\ell)^\varepsilon = 1 + O(\epsilon\log\ell)$ grows slowly with ℓ. Each successive encoder vector has a larger sparsity (as opposed to a fixed sparsity $r = 5k + 5k/\varepsilon$ in the batch algorithm). If we used a constant sparsity $r_j = 5 + 5k/\varepsilon$ for every encoder vector in the iterative algorithm, the relative error term becomes $1 + O(\varepsilon\ell)$ as opposed to $1 + O(\varepsilon\log\ell)$. Just as with the PCA vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots$, we have a provably good encoder for any ℓ by taking the first ℓ factors $\mathbf{h}_1, \ldots, \mathbf{h}_\ell$. In the batch-encoder $\mathrm{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_k]$, we cannot guarantee that $\mathbf{h}_1$ will give a reconstruction comparable with X_1. The detailed proof is in the supplementary matrrial.

Proof. (Sketch) For $\ell \geq 1$, we define two quantities Q_ℓ, P_ℓ for that will be useful in the proof.

$$\begin{aligned} Q_\ell &= (1+\varepsilon)(1+\tfrac{1}{2}\varepsilon)(1+\tfrac{1}{3}\varepsilon)(1+\tfrac{1}{4}\varepsilon)\cdots(1+\tfrac{1}{\ell}\varepsilon); \\ P_\ell = Q_\ell - 1 &= (1+\varepsilon)(1+\tfrac{1}{2}\varepsilon)(1+\tfrac{1}{3}\varepsilon)(1+\tfrac{1}{4}\varepsilon)\cdots(1+\tfrac{1}{\ell}\varepsilon) - 1. \end{aligned}$$

Using Lemma 10 and induction, we can prove a bound on the loss of H_ℓ.

$$\mathbb{E}\left[\|\mathrm{X} - \mathrm{X}\mathrm{H}_\ell(\mathrm{X}\mathrm{H}_\ell)^\dagger\mathrm{X}\|_{\mathrm{F}}^2\right] \leq Q_\ell\|\mathrm{X} - \mathrm{X}_\ell\|_{\mathrm{F}}^2 + Q_\ell\sum_{j=2}^{\ell}\sigma_j^2\frac{P_{j-1}}{Q_{j-1}}. \tag{$*$}$$

When $\ell = 1$, the claim is that $\mathbb{E}[\|\mathrm{X} - \mathrm{X}\mathrm{H}_1(\mathrm{X}\mathrm{H}_1)^\dagger\mathrm{X}\|_{\mathrm{F}}^2] \leq (1+\varepsilon)\|\mathrm{X} - \mathrm{X}_1\|_{\mathrm{F}}^2$ (since the summation is empty), which is true by construction of $\mathrm{H}_1 = [\mathbf{h}_1]$ because $r_1 = 5 + 5/\varepsilon$. For the induction step, we apply Lemma 10 with $\delta = \varepsilon/(\ell+1)$, condition on $\mathsf{err} = \|\mathrm{X} - \mathrm{X}\mathrm{H}_\ell(\mathrm{X}\mathrm{H}_\ell)^\dagger\mathrm{X}\|_{\mathrm{F}}^2$ whose expectation is given in $(*)$, and use iterated expectation. The details are given in the supplementary material. The first term in the bound (1) follows by bounding Q_ℓ using elementary calculus:

$$\log Q_\ell = \sum_{i=1}^{\ell}\log\left(1 + \frac{\varepsilon}{i}\right) \leq \sum_{i=1}^{\ell}\frac{\varepsilon}{i} \leq \varepsilon\log(e\ell),$$

where we used $\log(1+x) \leq x$ for $x \geq 0$ and the well known upper bound $\log(e\ell)$ for the ℓth harmonic number $1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{\ell}$. Thus, $Q_\ell \leq (e\ell)^\varepsilon$. The rest of the proof is to bound the second term in $(*)$ to obtain the second term in (1). Obeserve that for $i \geq 1$,

$$P_i = Q_i - 1 = \varepsilon\frac{Q_i}{Q_1} + \frac{Q_i}{Q_1} - 1 \leq \varepsilon\frac{Q_i}{Q_1} + Q_{i-1} - 1 = \varepsilon\frac{Q_i}{Q_1} + P_{i-1},$$

where we used $Q_i/Q_1 \leq Q_{i-1}$ and we define $P_0 = 0$. After some algebra which we omit,

$$\sum_{j=2}^{\ell}\sigma_j^2\frac{P_{j-1}}{Q_{j-1}} \leq \frac{\varepsilon}{Q_1}\|\mathrm{X}_\ell - \mathrm{X}_1\|_{\mathrm{F}}^2 + \sum_{j=3}^{\ell}\sigma_j^2\frac{P_{j-2}}{Q_{j-2}}.$$

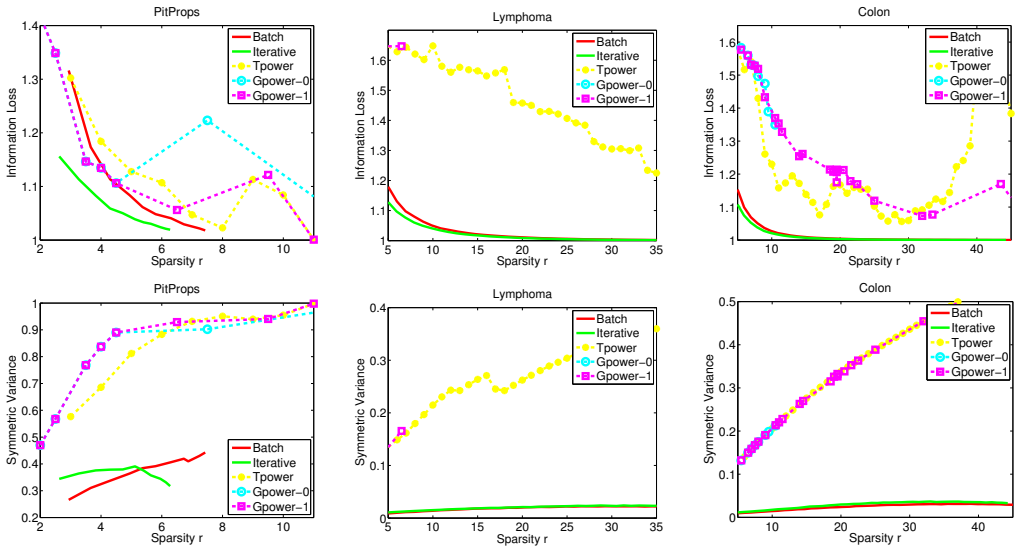

Figure 1: Performance of the sparse encoder algorithms on the PitProps data (left), Lymphoma data (middle) and Colon data (right) data: We Information loss (top) and symmetric explained variance (bottom) with $k = 2$. Our algorithms give the minimum information loss which decreases inversely with r as the theory predicts. It is no surprise that existing sparse-PCA algorithms do better at maximizing symmetric explained variance.

Using this reduction it is now an elementary task to prove by induction that

$$\sum_{j=2}^{\ell} \sigma_j^2 \frac{P_{j-1}}{Q_{j-1}} \leq \frac{\varepsilon}{Q_1} \sum_{j=1}^{\ell-1} \|\mathrm{X}_\ell - \mathrm{X}_j\|_{\mathrm{F}}^2.$$

Since $\|\mathrm{X}_\ell - \mathrm{X}_j\|_{\mathrm{F}}^2 \leq \|\mathrm{X}_\ell - \mathrm{X}_1\|_{\mathrm{F}}^2 (\ell - j)/(\ell - 1)$, we have that

$$\sum_{j=2}^{\ell} \sigma_j^2 \frac{P_{j-1}}{Q_{j-1}} \leq \frac{\varepsilon \|\mathrm{X}_\ell - \mathrm{X}_1\|_{\mathrm{F}}^2}{Q_1(\ell-1)} \sum_{j=1}^{\ell-1} \ell - j = \frac{\varepsilon \ell \|\mathrm{X}_\ell - \mathrm{X}_1\|_{\mathrm{F}}^2}{2Q_1}.$$

Using $(*)$, we have that

$$\|\mathrm{X} - \mathrm{X}\mathrm{H}_\ell(\mathrm{X}\mathrm{H}_\ell)^\dagger \mathrm{X}\|_{\mathrm{F}}^2 \leq (e\ell)^\varepsilon \|\mathrm{X} - \mathrm{X}_\ell\|_{\mathrm{F}}^2 + \frac{\varepsilon \ell \|\mathrm{X}_\ell - \mathrm{X}_1\|_{\mathrm{F}}^2}{2} \cdot \frac{Q_\ell}{Q_1}.$$

The result finally follows because

$$\log \frac{Q_\ell}{Q_1} = \sum_{i=2}^{\ell} \log\left(1 + \frac{\varepsilon}{i}\right) \leq \varepsilon \sum_{i=2}^{\ell} \frac{1}{i} \leq \varepsilon(\log(e\ell) - 1) = \varepsilon \log \ell,$$

and so $Q_\ell / Q_1 \leq \ell^\varepsilon$. ■

4 Demonstration

We empirically demonstrate our algorithms against existing state-of-the-art sparse PCA methods. The inputs are $\mathrm{X} \in \mathbb{R}^{n \times d}$, the number of components k and the sparsity parameter r. The output is the sparse encoder $\mathrm{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k] \in \mathbb{R}^{n \times k}$ with $\|\mathbf{h}_i\|_0 \leq r$; H is used to project X onto some subspace to obtain a reconstruction $\hat{\mathrm{X}}$ which decomposes the variance into two terms:

$$\|\mathrm{X}\|_{\mathrm{F}}^2 = \|\mathrm{X} - \hat{\mathrm{X}}\|_{\mathrm{F}}^2 + \|\hat{\mathrm{X}}\|_{\mathrm{F}}^2 = \text{Loss} + \text{Explained Variance}$$

For symmetric auto-encoders, minimizing loss is equivalent to maximizing the symmetric explained variance, the path traditional sparse-PCA takes,

$$\text{Symmetric Explained Variance} = \|\mathrm{X}\mathrm{H}\mathrm{H}^\dagger\|_{\mathrm{F}}^2 / \|\mathrm{X}_k\|_{\mathrm{F}}^2 \leq 1$$

To capture how informative the sparse components are, we can use the normalized loss:

$$\text{Loss} = \|\mathrm{X} - \mathrm{X}\mathrm{H}(\mathrm{X}\mathrm{H})^\dagger \mathrm{X}\|_{\mathrm{F}}^2 / \|\mathrm{X} - \mathrm{X}_k\|_{\mathrm{F}}^2 \geq 1.$$

We report the symmetric explained variance primarily for historical reasons because existing sparse PCA methods have constructed auto-encoders to optimize the symmetric explained variance.

We implemented an instance of the sparse PCA algorithm of Theorem 7 with the deterministic technique described in part (i) in Theorem 6. (This algorithm gives a constant factor approximation, as opposed to the relative error approximation of the algorithm in Theorem 7, but it is deterministic and simpler to implement.) We call this the "Batch" sparse linear auto-encoder algorithm. We correspondingly implement an "Iterative" version with fixed sparsity r in each principal component. In each step of the iterative sparse auto-encoder algorithm we use the above batch algorithm to select one principal component with sparsity at most r.

We compare our to the following state-of-the-art sparse PCA algorithms: (1) **T-Power:** truncated power method [23]. (2) **G-power-ℓ_0:** generalized power method with ℓ_0 regularization [10]. (3) **G-power-ℓ_1:** generalized power method with ℓ_1 regularization [10]. All these algorithms were designed to operate for $k = 1$ (notice our algorithms handle any k) so to pick k components, we use the "deflation" method suggested in [13]. We use the same data sets used by these prior algorithms (all available in [23]): PitProps ($\mathrm{X} \in \mathbb{R}^{13\times 13}$); Colon ($\mathrm{X} \in \mathbb{R}^{500\times 500}$); Lymphoma ($\mathrm{X} \in \mathbb{R}^{500\times 500}$).

The qualitative results for different k are similar so we only show $k = 2$ in Figure 1. The take-away is that loss and symmetric variance give very different sparse encoders (example encoders $[\mathbf{h}_1, \mathbf{h}_2]$ with $r = 5$ are shown on the right). This underlines why the correct objective is important. The machine learning goal is to preserve as much information as possible, which makes loss the compeling objective. The figures show that as r increases, our algorithms deliver near-optimal $1 + O(1/r)$ normalized loss, as the theory guarantees. The "iterative" algorithm has better empirical performance than the batch algorithm.

Batch		**Iter.**		**TP**		**GP-ℓ_0**		**GP-ℓ_1**	
$\mathbf{h}_1$	$\mathbf{h}_2$	$\mathbf{h}_1$	$\mathbf{h}_2$	$\mathbf{h}_1$	$\mathbf{h}_2$	$\mathbf{h}_1$	$\mathbf{h}_2$	$\mathbf{h}_1$	$\mathbf{h}_2$
0	0	0	0	0.5	0	0.7	0	0.6	0
−0.8	−0.3	−0.6	−0.8	0.5	0	0.7	0	0.6	0
0	0	0	−0.4	0	0.6	0	0.7	0	0.7
0	−0.8	0	−0.2	0	0.6	0	0.7	0	0.7
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0.3	0	0	0	0
0	0	−0.7	−0.1	0.4	0	0	0	0	0
−0.3	0.3	0	0	0	0	0	0	0	0
0	0	−0.3	0	0.4	0	0	0	0.5	0
0	0	−0.1	0	0.4	−0.2	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	−0.2	0	0.3	0	0	0	0
0.5	−0.4	0	0	0	0	0	0	0	0

Summary. Loss minimization and variance maximization give very different encoders under a sparsity constraint. The empirical performance of our loss minimization algorithms follows the theory. Our iterative algorithm is empirically better though it has a slightly worse theoretical guarantee.

5 Discussion

Historically, sparse PCA was cardinality constrained variance maximization. Variance *per se* has no intrinsic value, and is hard to define for non-orthogonal or correlated encoders, which is to be expected once you introduce a sparsity constraint. Our definition of loss is general and captures the machine learning goal of preserving as much information as possible.

We gave theoretical guarantees for sparse encoders. Our iterative algorithm has a weaker bound than our batch algorithm, yet the iterative algorithm is better empirically. Iterative algorithms are tough to analyze, and it remains open whether a tighter analysis can be given. We conjecture that the iterative algorithm is as good or better than the batch algorithm, though proving it seems elusive.

Finally, we have not optimized for running times. Considerable speed-ups may be possible without sacrificing accuracy. For example, in the iterative algorithm (which repeatedly calls the CSSP algorithm with $k = 1$), it should be possible significantly speed up the generic algorithm (for arbitrary k) to a specialized one for $k = 1$. We leave such implementation optimizations for future work.

Acknowledgments. Magdon-Ismail was partially supported by NSF:IIS 1124827 and by the Army Research Laboratory under Cooperative Agreement W911NF-09-2-0053 (the ARL-NSCTA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] M. Asteris, D. Papailiopoulos, and A. Dimakis. Non-negative sparse PCA with provable guarantees. In *Proc. ICML*, 2014.

[2] M. Asteris, D. Papailiopoulos, and G. Karystinos. Sparse principal component of a rank-deficient matrix. In *Proc. ISIT*, 2011.

[3] P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2:53–58, 1988.

[4] H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59:291–294, 1988.

[5] C. Boutsidis, P. Drineas, and M. Magdon-Ismail. Near-optimal column-based matrix reconstruction. *SIAM Journal on Computing*, 43(2), 2014.

[6] J. Cadima and I. Jolliffe. Loadings and correlations in the interpretation of principal components. *Applied Statistics*, 22:203–214, 1995.

[7] G. Cottrell and P. Munro. Principal components analysis of images via back propagation. In *Proc. SPIE 1001, Visual Communications and Image Processing '88*, 1988.

[8] A. d'Aspremont, F. Bach, and L. E. Ghaoui. Optimal solutions for sparse principal component analysis. *Journal of Machine Learning Research*, 9:1269–1294, June 2008.

[9] A. d'Aspremont, L. El Ghaoui, M. I. Jordan, and G. R. G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. *SIAM Review*, 49(3):434–448, 2007.

[10] M. Journée, Y. Nesterov, P. Richtárik, and R. Sepulchre. Generalized power method for sparse principal component analysis. *The Journal of Machine Learning Research*, 11:517–553, 2010.

[11] H. Kaiser. The varimax criterion for analytic rotation in factor analysis. *Psychometrika*, 23(3):187–200, 1958.

[12] J. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

[13] L. W. Mackey. Deflation methods for sparse PCA. In *Proc. NIPS*, pages 1017–1024, 2009.

[14] M. Magdon-Ismail. Np-hardness and inapproximability of sparse pca. *arXiv:1502.05675*, 2015.

[15] A. Makhzani and B. Frey. k-sparse autoencoders. In *ICLR*, 2014.

[16] B. Moghaddam, Y. Weiss, and S. Avidan. Generalized spectral bounds for sparse LDA. In *ICML*, 2006.

[17] E. Oja. Data compression, feature extraction and autoassociation in feedforward neural networks. In *Artificial Neural Networks*, volume 1, pages 737–745, 1991.

[18] E. Oja. Principal components, minor components and linear neural networks. *Neural Networks*, 5:927–935, 1992.

[19] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572, 1901.

[20] J. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5):401–409, 1969.

[21] H. Shen and J. Z. Huang. Sparse principal component analysis via regularized low rank matrix approximation. *Journal of Multivariate Analysis*, 99:1015–1034, July 2008.

[22] N. Trendafilov, I. T. Jolliffe, and M. Uddin. A modified principal component technique based on the lasso. *Journal of Computational and Graphical Statistics*, 12:531–547, 2003.

[23] X.-T. Yuan and T. Zhang. Truncated power method for sparse eigenvalue problems. *The Journal of Machine Learning Research*, 14(1):899–925, 2013.

[24] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of Computational & Graphical Statistics*, 15(2):265–286, 2006.