Effectiveness of Sparse Features: An Application of Sparse PCA

Anonymous Author(s)
Affiliation
Address
email

Abstract

Principal Component Analysis (PCA) is one of the most widely used unsupervised learning techniques to extract features from unlabeled data and form a basis of uncorrelated features that can represent the original data with the minimum loss of information. Because of the reduced feature space dimension, PCA is also used simply for data compression. However, beyond the ability to reconstruct the original data with the minimum reconstruction error, these learned features can be useful in carrying out predictive tasks. Recently, many researchers have been investigating the effectiveness of enforcing sparsity on the learned features and proposed various optimization schemes and techniques to learn sparse features and applied them in different domains of tasks. In this paper, we present an overview of the concept of sparse PCA (SPCA) [11], and we apply it to the classification of handwritten digits. We demonstrate the effectiveness of SPCA by comparing our classification result to that attained by the standard PCA and other approaches. We further relate SPCA to other studies of sparse features in the field.

1 Introduction

Principal Component Analysis (PCA) is a classic tool for dimensionality reduction and data visualization. Since data representations are often redundant, extracting the principal components can give a better understanding of which components indeed play a role in representing data, and what these principal component directions (also known as *loadings*, or *filters*; we keep consistent and use the term *loadings* in the rest of this paper) are. Thus, PCA is also widely employed as one of the standard techniques for data preprocessing. However, despite its popularity, PCA has a major shortcoming: each principal component is a linear combination of all the original features, and their coefficients are typically non-zero. This can make interpretation hard, especially when a certain number of principal components is chosen. Zou et al. introduced an approach to retrieving the principal components with sparse loadings, called *sparse principal component analysis* (SPCA) [11], and different optimization techniques have been proposed to solve for these sparse principal components [4] [11]. Beyond producing a sparser and naturally more interpretable representation of data, sparse features have proved effective in many predictive tasks such as image classification and object recognition [3] [8] [10]. In this paper, we apply SPCA to the handwritten digit recognition problem and compare the result we obtain to that using classic PCA. Our result shows significant improvement in the classification of handwritten digit images with SPCA. In the next section, we present an overview of SPCA and how introducing extra constraints to the optimization problem would enforce sparse PCA loadings. Rather than delving into the technicality of the details in solving the optimization problem to obtain the sparse principal components, we simply present a necessary (and not very technical) background of the version of SPCA we employ in this work and then focus on our experiment in applying SPCA to images of handwritten digits and our encouraging result in Section 3.

2 Background

PCA can be viewed as looking for the set of directions that, when projecting data onto them, gives the maximum variance in the projected data, or alternatively as seeking the principal directions that can reconstruct the original data with the minimum loss of information, that is, to minimize the reconstruction error. One most common approach to computing PCA is via the *singular value decomposition* (SVD). Suppose $\mathbf{X}$ is the standardized $n \times d$ data matrix, where each row $\mathbf{x}_i$ is one data instance in d dimension (i.e. with d features). The SVD of $\mathbf{X}$ is

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T \tag{1}$$

The columns of $\mathbf{U}$ are the *left singular vectors*; the non-zero entries of the diagonal matrix $\mathbf{S}$ are the corresponding *singular values*; and the columns of $\mathbf{V}$ are the *right singular vectors*. In this decomposed form, $\mathbf{Z} = \mathbf{U}\mathbf{S}$ represent the principal components of the data, whereas the columns of $\mathbf{V}$ are the PCA loadings. We can also easily truncate the matrices to obtain the top-k principal components and loadings.

Lasso has been provably effective and widely adopted to enforce sparse solutions in many optimization problems. As PCA is an optimization problem that searches for the directions of maximum variance, or equivalently those minimizing the reconstruction error, we can introduce the L_1 penalty to enforce sparsity the same way as in a linear regression problem. Indeed, PCA has been shown to be exactly a *ridge regression* problem [3]. Jolliffe et al. proposed the *SCoTLASS* procedure [4] to solve for directions of maximum variance with the extra absolute-value constraints:

$$\begin{gathered}\mathbf{v}_k = \arg\max_{\mathbf{v}} \mathbf{v}^T(\mathbf{X}^T\mathbf{X})\mathbf{v} \\ subject\ to\ \sum_{i=1}^{d} \mathbf{v}_i \leq \lambda\ and\ \mathbf{v}^T\mathbf{v} = 1\end{gathered} \tag{2}$$

and ensure that the kth principal direction is uncorrelated (i.e. orthogonal) to the first $k-1$ directions. Nevertheless, this constrained optimization problem is not convex, and thus the computations are hard and time-consuming. Zou et al., on the other hand, approached the PCA problem by adding the L_1 penalty to its ridge regression formulation to encourage sparsity, making it a combination of ridge and *lasso regression* problem (called *elastic net*). They solved the optimization problem

$$\begin{gathered}(\mathbf{A}^*, \mathbf{B}^*) = \arg\min_{\mathbf{A},\mathbf{B}} \sum_{i=1}^{n} ||\mathbf{x}_i - \mathbf{A}\mathbf{B}^T\mathbf{x}_i||_2^2 + \lambda_1 \sum_{j=1}^{d} ||\mathbf{b}_j||_2^2 + \sum_{j=1}^{d} \lambda_{2,j} ||\mathbf{b}_j||_1 \\ subject\ to\ \mathbf{A}^T\mathbf{A} = \mathbf{I}\end{gathered} \tag{3}$$

where $\mathbf{b}_j$ denotes the jth column of matrix $\mathbf{B}$, and obtained that $\mathbf{b}_j$ is proportional to the jth PCA loading that exhibit sparse nature. Now this problem is convex in one variable provided the other is fixed, although it is not jointly convex in $\mathbf{A}$ and $\mathbf{B}$. It can be solved using methods based on coordinate descent, that is, switching between optimizing in one variable while holding the other fixed. However, Mairal et al. observed empirically that a preconditioned *least angle regression* (LARS) algorithm [2] solves the problem with higher accuracy for all possible values of λ_2 [7] especially when SPCA gives up the property that the loadings are uncorrelated. Zou et al. also remarked that, based on empirical evidence, the ridge coefficient λ_1 in (3) mainly serves the preconditioning effect for $n < d$, and the solution does not change much varying λ_1. So for $n < d$, a default choice can be $\lambda_1 = 0$ [11].

As a result, an alternative formulation was introduced [7], which drops the ridge penalty term and solves the L_1-regularized least squares problem:

$$\begin{gathered}(\mathbf{U}^*, \mathbf{V}^*) = \arg\min_{\mathbf{U},\mathbf{V}} \frac{1}{2}||\mathbf{X} - \mathbf{U}\mathbf{V}||_2^2 + \lambda||\mathbf{V}||_1 \\ subject\ to\ ||\mathbf{U}_k||_2 = 1\ for\ all\ 0 \leq k < n\end{gathered} \tag{4}$$

which finds a matrix factorization with sparsity constraints that can reconstruct the original matrix with minimum loss of information. This problem is termed *sparse coding*, and efficient algorithms have been proposed for solving it [2] [6]. The normalized columns of $\mathbf{V}^*$ are the sparse PCA loadings.

3 Experiment

In this work, we apply the SPCA technique to the widely used handwritten digit dataset to extract sparse features, called SPCA loadings. We then compare the SPCA loadings to the standard PCA loadings and highlight the significant difference between them. Finally, both sets of these loadings are used to represent the data in a highly reduced dimension, and the predictive task is carried out with both representations.

3.1 Problem

We applied both PCA and SPCA to the MNIST handwritten digit training dataset to learn features in a unsupervised manner [5]. That is, we do not look at the labels (though they are provided with the training data) when extracting the (sparse) PCA loadings. We used these loadings to represent both the training and test data and perform classification of the handwritten digits in the test dataset.

The MNIST dataset consists of a training set of 50000 collected handwritten digits each digitized to a 28×28 grayscale (thus with dimension 784) image, as well as a test set of 10000 images for the purpose of experimenting with different classification techniques.

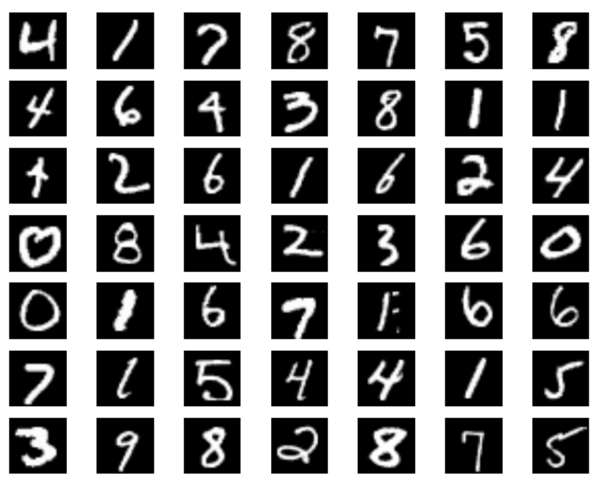

Figure 1: A random selection of 49 handwritten digit images from the MNIST training dataset; we can see a great variety of handwriting styles - thickness of digits; relative length of the two tails of ”7”; distance between the two tips of ”4”; inclinedness of digits.

Figure 1 displays a random sample of 49 handwritten digit images from the training set. Even in this small sample, we observe a great variety of handwriting styles - thickness of digits; relative length of the two tails of ”7”; distance between the two tips of ”4”; inclinedness of digits - which clearly introduce difficulty in classifying these digits.

3.2 Approach

We learned 49 SPCA loadings (and also the standard PCA loadings for comparison) based on the handwritten digits in the training dataset. To extract these SPCA loadings, we used a machine learning package *scikits-learn* for Python [9], which implements (4) and solves the optimization problem with the LARS algorithm. We note that the only preprocessing of the data was the centering of each feature across all data instances. We also note that this implementation uses a *batch* technique, which iterates over smaller batches of the set of all 784 features to solve the optimization problem, rather than using the entire feature set at once. This can yield great gain in computational efficiency at the expense of slight accuracy loss. Here, we chose this implementation simply for the gain in speed.

With these SPCA (and PCA) loadings, we transformed the handwritten digit images in the test dataset to the SPCA (and PCA) feature space to obtain their corresponding (sparse) principal components. Then the *K-Nearest Neighbors* (k-NN) algorithm was used for the classification with Euclidean distance between (transformed) feature vectors as the distance measure. We have used cross validation on the training dataset to select the optimal K that minimizes the maximum cross validation error. For a comparison purpose, we also performed classification on the test dataset with the raw pixel values as the features using the k-NN algorithm. However, we do not expect that the SPCA and PCA approaches would give classification errors as low as that with the raw pixel values since we have reduced the data dimensionality to less than 7% of the original. Our main focus is on the comparison between the results obtained using PCA loadings and SPCA loadings.

3.3 Result

We examined the SPCA loadings and found that, unlike those PCA loadings which identify rather global features, they are much more ”local”. This concept of local features can be obviously seen in Figure 2. It is also worth highlighting that while PCA loadings that correspond to smaller singular values usually capture more high-frequency features, this phenomenon does not seem to appear in the SPCA loading due to their sparse nature.

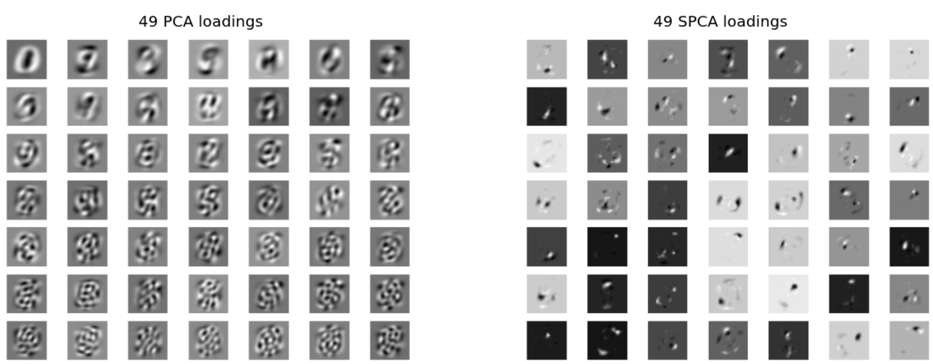

Figure 2: Left: the 49 PCA loadings. Right: the 49 SPCA loadings. The sparsity in the SPCA loadings are obvious. Due to this sparse nature, these loadings identify features that are much more local.

We ended up with result that demonstrates the benefits of SPCA loadings in the classification of these handwritten digits. The classification error is much smaller with the SPCA loadings than with the PCA loadings (See Table 1). While there is still more to explore with these SPCA loadings, the result suggests that they do play a role in identifying important features that distinguish the handwritten digits, and these features are particularly helpful in the predictive tasks.

4 Discussion and connections to other works

In this paper, we applied the SPCA to extract features from the MNIST handwritten digit dataset and demonstrate that these features are provably effective in the classification task. Perhaps we could have reached a classification error level comparable to that obtained with pixel-by-pixel representation with an increased number of SPCA loadings. Moreover, we used k-NN for the classification task because of its simplicity (and, in spite of being simple, this memory-based algorithm has been

Table 1: Classification errors on the MNIST handwritten digit test dataset using k-NN with raw pixels, PCA loadings, and SPCA loadings. SPCA proves effective over the dense PCA.

Features	**Classification error**
784 pixels	0.028
49 PCA loadings	0.0749
49 SPCA loadings	0.054

shown to be very effective in a wide range of classification or regression problems). Instead of k-NN, we could have used other popular supervised learning methods such as *support vector machine* or *neural network* to possibly achieve better classification with the same SPCA loadings. Nevertheless, Our result suggests that the sparse features, while losing some minimal information in the sense of reconstructing the original data, offer a more compact and interpretable representation of the data and thus make them much more distinguishable in classification.

In addition to experimenting with the number of SPCA loadings to learn, we also discuss the controling hyper-parameter λ in (4), which determines the level of sparsity in the SPCA loadings. A very large λ will essentially shrink all components of the loadings to 0, whereas too small values of λ have little effect in enforcing sparsity. In our work, we did not spend much time experimenting with different values of this hyper-parameter. Therefore, an optimal value of λ could have yielded a better classification result in our experiment. However, this also brings up the arduous problem of tuning the controling hyper-parameter(s), and in general there is no standard approach to this task since the optimal hyper-parameter also depends on the available data at hand and the predictive task (if the data are to be used for such purpose). Ngiam et al. proposed *sparse filtering* [8], an algorithm that learns sparse features without requiring a sparsity-controling hyper-parameter and thus avoids the extensive tuning of hyper-parameter(s). In [8], they applied this algorithm to learn sparse features from natural images and evaluated their effectiveness on an object classification task. But the problem still remains to select the number of sparse features to learn (unless in cases where a strict dimensionality reduction is specified), and various approaches have been proposed, such as to preserve certain level of data variance.

Computational feasibility and efficiency is also an issue for algorithms that learn sparse features. Ngiam et al. argued that the sparse coding requires a unreasonably long convergence time to solve the optimization problem (4) when the input data have a large number of features [8], and that their proposed sparse filtering remedies this inefficiency to some extent [8].

A lot of recent research have explored the effectiveness of sparse representation and how to automatically learn these sparse features from unlabeled data (i.e. in a unsupervised way), as well as finding various application domains. Raina et al. proposed to learn *transferable* sparse features from random natural images with sparse coding, and applied the learned bases to image classification, handwritten character recognition, webpage classification, etc [10]. Jenatton et al. "[went] beyond sparse PCA and [proposed] *structured sparse PCA* (SSPCA)," which learns features that are "not only sparse but also respect some a priori structural constraints deemed relevant to model the data" [3]. They have successfully applied their proposed structured approaches to the tasks of denoising of synthetic signals and face recognition and reached significant results demonstrating the benefits of SSPCA. Boutsidis et al. introduced both deterministic and randomized algorithms for learning sparse features that are linear combinations of a (specified) small number of original features and can achieve comparable reconstruction error attained by PCA loadings [1]. They also argued that "input sparseness is closely related to feature selection and automatic relevance determination", an observation that is backed up by our experiment result in this work.

We hope that our work, as well as all other related studies, will encourage more investigation in the field of exploiting sparse representation of data in various domains.

References

[1] C. Boutsidis, P. Drineas, and M. Magdon-Ismail. Sparse Features for PCA-Like Linear Regression. To appear in NIPS, 2011.

[2] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least Angle Regression. *Annals of Statistics*, 32(2):407–499, 2004.

[3] R. Jenatton, G. Obozinski, and F. Bach. Structured Sparse Principal Component Analysis. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, Sardinia, Italy, 2010.

[4] I. T. Jolliffe, N. T. Trendafilov, and M. Uddin. A Modified Principal Component Technique Based on the Lasso. *Journal of Computational and Graphical Statistics*, 12(3):531–547, 2003.

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haner. Gradient-based Learning Applied to Document Recognition. In *Proceedings of the IEEE*, number 11, pages 2278–2324, 1998.

[6] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient Sparse Coding Algorithms. In *Advances of Neural Information Processing Systems 20*, Vancouver, Canada, 2007.

[7] J Mairal, F. Bach, J. Ponce, and G. Sapiro. Online Dictionary Learning for Sparse Coding. In *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

[8] J. Ngiam, P. W. Koh, Z. Chen, S. Bhaskar, and A. Y. Ng. Sparse Filtering. To appear in NIPS, 2011.

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

[10] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught Learning: Transfer Learning from Unlabeled Data. In *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, USA, 2007.

[11] H. Zou, T. Hastie, and Tibshirani R. Sparse Principal Component Analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286, 2006.