Acta Technica *62* (2017), No. 5A, 163–172

The application of machine learning algorithms in multivariate statistical analysis

Cong Gu[2,3]

Abstract. Machine learning and statistics are closely related fields. The ideas of machine learning, from methodological principles to theoretical tools, have had a long pre-history in multivariate statistical analysis. Principal Component Analysis (PCA) is one of the most important methods in the multivariate statistical analysis, which can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning). In this paper, PCA was discussed and combed out under the framework of machine learning by introducing the encoder function and Frobenius norm. These two approaches to build and to solve PCA model were given, and machine learning takes a much more efficient algorithm in principal component analysis, rather than the traditional statistical approach in the multivariate statistical analysis.

Key words. Machine learning, principal component analysis (PCA), encoder function, Frobenius norm.

1. Introduction

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed. Machine learning tasks are typically classified into three broad categories, depending on the nature of the learning "signal" or "feedback" available to a learning system, as following:

(1) Supervised learning [1]. The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.

(2) Unsupervised learning [2,3]. No labels are given to the learning algorithm,

[1]Acknowledgment - The author would like to thank the anonymous referees for their helpful comments to improve the earlier version of the paper. This research was supported by the National Science Foundation of China (Grant: 11401604).

[2]Workshop 1 - College of Science, Zhongyuan University of Technology, Zhengzhou, China; e-mail: gucong@zju.edu.cn

[3]Workshop 2 - School of Mathematical Sciences, Zhejiang University, Hangzhou, China; e-mail: gucong@zju.edu.cn

leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).

(3) Reinforcement learning [4,5]. A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). The program is provided feedback in terms of rewards and punishments as it navigates its problem space.

The principal component analysis was invented in 1901 by Karl Pearson as an analogue of the principal axis theorem in mechanics; it was later independently developed and named by Harold Hotelling in the 1930s. Depending on the field of application, it is also named the discrete Karhunen-Loève Transform (KLT) in signal processing, the Hotelling Transform in multivariate quality control, the Proper Orthogonal Decomposition (POD) in mechanical engineering, etc. Machine learning takes a much more efficient algorithm in the principal component analysis (PCA), which is one of the most important methods in the multivariate statistical analysis, rather than the traditional statistical approach[6,7].

2. Methodology

Let $X_1, X_2, \ldots, X_n$ be the features of some type of data, such as several courses' scores of a student, several performance parameters of an automobile (i.e., maximum speed, turn radius, and so on), etc.

Now we are given dataset of m observations $X^i; i = 1, \cdots, m$, and $X^i \in R^n$ for each i($n << m$). Thus we could use the following data matrix $X \in R^{m \times n}$:

$$X = \begin{pmatrix} x_1^{(1)} & \cdots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(m)} & \cdots & x_n^{(m)} \end{pmatrix} \tag{1}$$

The i-th row represents n features of the point $X^{(i)}$, and the j-th column represents m observations of the j-th feature X_j.

Suppose we want to visualize the m observations with measurements on a set of n features $X_1, X_2, \ldots, X_n$, as part of an exploratory data analysis. We could do this by examining 2D scatter plots of the data, each of which contains the m observations' measurements on two of the features. If n is large, it will be impossible to look at all of them. Besides, most likely none of them will be informative since they each contain just a small fraction of the total information present in the dataset. We would like to find a low-dimensional representation of the data that captures as much of the information as possible.

The principal component analysis (PCA) provides a tool to do this. The central idea of PCA is to reduce the dimensionality of a dataset which consists of a large number of interrelated variables, while retaining as much as possible of the variation present in the dataset. This is achieved by transforming to a new set of variables, the principal components (PCs), which are uncorrelated and ordered so that the

first few retain most of the variation present in all of the original variables.

The first PC Y_1 of the features $X_1, X_2, \ldots, X_n$ is the normalized linear combination of the features as follows:

$$Y_1 = l_{11}X_1 + l_{12}X_2 + \cdots + l_{1n}X_n \tag{2}$$

that has the largest variance. By the word "normalized", we mean that $\sum_{j=1}^{n} l_{1j}^2 = 1$. We refer to the $l_{1i}(i = 1, \ldots, n)$ as the loadings of the first PC, and $L_1 = (l_{11}, l_{12}, \ldots, l_{1n})^T$ is called the PC loading vector.

Given a $m \times n$ dataset X described by Eq. (1), how do we compute the first principal component? Since we are only interested in variance, we assume that each of the variables in X has been centered to have mean zero (that is, the column means of X are zero). We then look for the linear combination of the sample feature values of the form

$$y_1^{(i)} = l_{11}x_1^{(i)} + l_{12}x_2^{(i)} + \cdots + l_{1n}x_n^{(i)} \tag{3}$$

that has largest sample variance, subject to the constraint that $\sum_{j=1}^{n} l_{1j}^2 = 1$. In other words, the first principal component loading vector solves the optimization problem

$$\max_{l_{11},\ldots,l_{1n}} \left[\frac{1}{n} \sum_{i=1}^{m} \left(\sum_{j=1}^{n} l_{ij}x_j^{(i)} \right)^2 \right] Subjectto \sum_{j=1}^{n} l_{1j}^2 = 1 \tag{4}$$

From Eq. (3), we can write the objective in Eq. (4) as $\frac{1}{n}\sum_{i=1}^{m}(y_1^{(i)})^2$. Since $\frac{1}{n}\sum_{i=1}^{m} x_1^{(i)} = 0$, the average of the $y_1^{(1)}, y_1^{(2)}, \cdots, y_1^{(m)}$ will be zero as well. So the objective that we are maximizing in Eq. (4) is just the sample variance of the m values of $y_1^{(i)}$. We refer to $y_1^{(1)}, y_1^{(2)}, \cdots, y_1^{(m)}$ as the scores of the first PC.

After the first PC Y_1 of the features has been determined, the second PC is the linear combination of $X_1, X_2, \ldots, X_n$ that has maximal variance out of all linear combinations that are uncorrelated with Y_1. The second PC scores $y_2^{(1)}, y_2^{(2)}, \cdots, y_2^{(m)}$ take the form

$$y_2^{(i)} = l_{12}x_1^{(i)} + l_{22}x_2^{(i)} + \cdots + l_{2n}x_n^{(i)} \tag{5}$$

where $L_2 = (l_{21}, l_{22}, \ldots, l_{2n})^T$ is the second PC loading vector. It turns out that constraining Y_2 to be uncorrelated with Y_1 is equivalent to constraining L_2 to be orthogonal to L_1.

Finally, we get k principal components $Y_1, Y_2, \ldots, Y_k$, where $k \leq n$, as follows:

$$\begin{cases} Y_1 = l_{11}X_1 + l_{12}X_2 + \cdots + l_{1n}X_n \\ Y_2 = l_{21}X_1 + l_{22}X_2 + \cdots + l_{2n}X_n \\ \vdots \\ Y_k = l_{k1}X_1 + l_{k2}X_2 + \cdots + l_{kn}X_n \end{cases} \tag{6}$$

Each PC is a linear combination of the original variables, and uncorrelated with each other.

3. Results and discussion

3.1. PCA under the framework of machine learning

Suppose we have a collection of $m$observations $\{X^{(1)}, \cdots, X^{(m)}\}$ in R^n. And we would like to apply "lossy compression" to these points under the framework of machine learning. Here, lossy compression means storing these points in a way that requires less memory but may lose some precision.

First of all, we encode these points to represent a lower-dimensional version of them. For each input $X^{(i)} \in R^n$, we will find a corresponding code vector $Y^{(i)} \in R^k$. If k is smaller than n, it will take less memory to store the code points than the original data. We will want to find some encoding function that produces the code for an input,$f(x) = y$, and a decoding function that produces the reconstructed input given its code, $x \approx g(y) = g(f(x))$.

PCA is defined by our choice of the decoding function. Specifically, to make the decoder very simple, we choose to use matrix multiplication to map the code back into R^n. Let $g(y) = Dy$, where $D \in R^{n\times k}$ is the matrix defining the decoding. Computing the optimal code for this decoder could be a difficult problem. To keep the encoding problem easy, PCA constrains the columns of D to be orthogonal to each other, and all of the columns of D to have unit norm.

In order to turn this basic idea into an algorithm, the first thing we need to do is figure out how to generate the optimal code point y^* for each input point x. One way to do this is to minimize the distance between the input x and its reconstruction $g(y^*)$. We can measure this distance using the squared L^2 norm:

$$y^* = \arg\min_y \|x - g(y)\|_2^2 \tag{7}$$

In the field of machine learning, the function we want minimize or maximize is called the objective function or criterion. When we are minimizing it, we may also call it the cost function, loss function, or error function.

Theorem 1. The solution of the optimization problem described by Eq. (7), is

$$y^* = D^T x \tag{8}$$

Proof: The function being minimized simplifies to

$$(x - g(y))^T(x - g(y)) = x^T x - 2x^T g(y) + g(y)^T g(y)$$

We can rewrite the optimization problem again, to omit the first term, since this term does not depend on y:

$$y^* = \arg\min_y -2x^T g(y) + g(y)^T g(y)$$

By substituting in the definition of $g(y)$:

$$y^* = \arg\min_y -2x^T Dy + y^T D^T Dy = \arg\min_y -2x^T Dy + y^T y$$

Finally, solve this optimization problem using vector calculus:

$$\nabla_y(-2x^T Dy + y^T y) = 0$$

$$-2D^T x + 2y = 0$$

$y = D^T x$

This makes the algorithm efficient: we can optimally encode x just using a matrix-vector operation. To encode a vector, we apply the encoder function

$$f(x) = D^T x \tag{9}$$

Using a further matrix multiplication, we can also define the PCA reconstruction operation:

$$r(x) = g(f(x)) = DD^T x \tag{10}$$

The last thing is to choose the encoding matrix D. In order to do this, we revisit the idea of minimizing the distance between inputs and reconstructions. However, since we will use the same matrix D to decode all of the points, we can no longer consider the points in isolation. Instead, we must minimize the Frobenius norm $\|A\|_F$ of the matrix of errors computed over all dimensions and all points:

$$D^* = \arg\min_D \|A\|_F = \arg\min_D \sqrt{\sum_{i,j} A_{i,j}^2} subject to D^T D = I_l \tag{11}$$

where

$$A_{i,j} = x_j^{(i)} - r(x^{(i)})_j. \tag{12}$$

To derive the algorithm for finding D^*, we will start by considering the case where $k = 1$. In this case, D is just a single vector d. Substituting Eq. (10) and Eq. (12) into Eq. (11) and simplifying D into d, the optimization problem reduces to

$$d^* = \arg\min_d \left\| x^{(i)} - dd^T x^{(i)} \right\|_2^2 subject to \|d\|_2 = 1 \tag{13}$$

Theorem 2. The optimal d^* in Eq. (13) is given by the eigenvector $X^T X$ corresponding to the largest eigenvalue.

Proof. It is more conventional to write scalar coefficients on the left of vector

they operate on. We therefore usually write such a formula as

$d^* = \arg\min_d \left\|x^{(i)} - d^T x^{(i)} d\right\|_2^2$ subject to $\|d\|_2 = 1$

or, exploiting the fact that a scalar is its own transpose, as

$d^* = \arg\min_d \left\|x^{(i)} - x^{(i)T} dd\right\|_2^2$ subject to $\|d\|_2 = 1$

Rewrite the problem in terms of a single design matrix of examples, rather than a sum over separate example vectors. Let $X \in R^{m\times n}$ be the matrix defined by Eq. (1), such that $X_{i,:}$ denotes the i-th row of X. i.e., the i-th observation of the dataset. We can now rewrite the problem as

$d^* = \arg\min_d \left\|X - Xdd^T\right\|_F^2$ subject to $d^T d = 1$

Disregarding the constraint for the moment, we can simplify the Frobenius norm portion as follows:

$$\begin{aligned}
&\arg\min_d \left\|X - Xdd^T\right\|_F^2 \\
&= \arg\min_d Tr\left(\left(X - Xdd^T\right)^T \left(X - Xdd^T\right)\right)
\end{aligned}$$

$$= \arg\min_d Tr\left(X^T X\right) - Tr\left(X^T Xdd^T\right) - Tr\left(dd^T X^T X\right) + Tr\left(dd^T X^T Xdd^T\right)$$

$$= \arg\min_d -2Tr\left(X^T Xdd^T\right) + Tr\left(dd^T X^T Xdd^T\right)$$

$$= \arg\min_d -2Tr\left(X^T Xdd^T\right) + Tr\left(X^T Xdd^T dd^T\right)$$

$$= \arg\min_d -Tr\left(X^T Xdd^T\right)$$

$$= \arg\max_d Tr\left(X^T Xdd^T\right)$$

At this point, we re-introduce the constraint:

$d^* == \arg\max_d Tr\left(X^T Xdd^T\right)$ subject to $d^T d = 1$

This optimization problem may be solved using eigendecompsition. Specifically, the optimal d is given by the eigenvector $X^T X$ corresponding to the largest eigenvalue.

In the general case, where $k > 1$, the matrix D is given by the l eigenvectors corresponding to the largest eigenvalues.

3.2. An example

Suppose we are given the dataset of 100 students' examination scores of 6 courses (see Table 1). The question is:

(1) Can we use one or two variables to represent the 6 variables of the data.

(2) How much of the original information do the one or two integrated variables

contain.

(3) Can we use the integrated variables to sort out the students.

Table 1.The data of 100 students' examination scores of 6 courses

Student ID	MATH	PHYS	CHEM	LITERAT	HISTORY	ENGLISH
1	65	61	72	84	81	69
2	77	77	76	64	70	55
3	67	63	49	65	67	57
4	80	69	75	74	74	63
5	74	70	80	84	81	74
6	78	84	75	62	71	64
7	66	71	67	52	65	57
8	77	71	57	72	86	71
9	83	100	79	41	67	50
...	...	...	...	...	...	...

The data points in the case are six dimensional, that is, each observation is a point in the 6 dimensional space. We want to represent 6 dimensional space in a lower dimensional space.

Let's think about an easier case. Assume that only two variables are represented by the horizontal and vertical coordinates, therefore each observation has two coordinates corresponding to the two axis value. If the data form an oval shaped lattice (which is possible under the assumption that the variables under normal). So this ellipse has a long axis and a short axis (see Figure 1). In the short axis direction, the data changes little, and in the extreme case, if degenerate into a short axis, the long axis direction can only explain the change of these points. In this way, from 2D to 1D dimensionality reduction naturally completed.

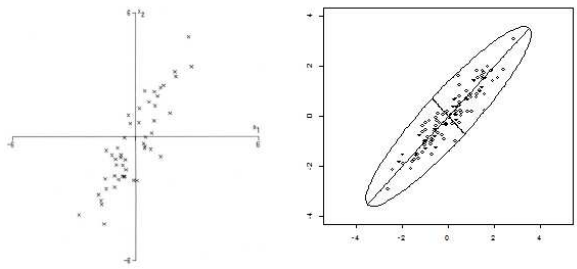

Fig. 1. Plot of observations on two variables

When the coordinate axis is parallel to the axis of the ellipse, the variables representing the long axis describe the main changes of the data, while the variables representing the short axis describe the minor changes of the data. However, the

coordinate axis is usually not parallel to the axis of the ellipse. Therefore, it is necessary to find the minor axis of the ellipse and transform it so that the new variable is parallel to the axis of the ellipse.

If the axis variable represents most of the information contained in the data, using these two variables instead of the original variables, and the dimensionality reduction is done.

The situation of multidimensional variables is similar to that of two dimensions. There are also high dimensional ellipsoids, but they can't be seen directly. Firstly, the principal axis of the ellipsoid of high dimension is found out, and then the longest axis representing most of the data information is used as the new variable, then principal component analysis is basically completed. Note that, similar to the two-dimensional case, the principal axes of the high dimensional ellipsoid are orthogonal to each other. These mutually orthogonal new variables are linear combinations of the original variables, which we have already defined as the principal components (PCs). The fewer the principal components, the better the dimensionality reduction. What is the standard? Some literatures suggest that the total length of the spindle selected is about 85% of the total length of the spindle.

Back to the example. We solve this problem by PCA.

Table 2. The total variance explained of the 6 eigenvalues

Component	Initial Eigenvalues			Extraction Sums of Squared Loadings		
	Total Variance	% of Variance	Cumulative %	Total Variance	% of Variance	Cumulative %
1	3.735	62.254	62.254	3.735	62.254	62.254
2	1.133	18.887	81.142	1.133	18.887	81.142
3	0.457	7.619	88.761			
4	0.323	5.376	94.137			
5	0.199	3.320	97.457			
6	0.153	2.543	100.000			

The cumulative value of the first two components accounted for 81.142% of the total variance. The contribution of the posterior eigenvalues is less and less.

Table 3. The principal component matrix

	Component					
	1	2	3	4	5	6
MATH	-0.806	0.353	-0.040	0.468	0.021	0.068
PHYS	-0.674	0.531	-0.454	-0.240	-0.001	-0.006
CHEM	-0.675	0.513	0.499	-0.181	0.002	0.003
LITERAT	0.893	0.306	-0.004	-0.037	0.077	0.320
HISTORY	0.825	0.435	0.002	0.079	-0.342	-0.083
ENGLISH	0.836	0.425	0.000	0.074	0.276	-0.197

Here each column represents the coefficient (proportion) of a principal component as a linear combination of the original variables. For example, the first principal component serves as a linear combination of six original variables, mathematics, physics, chemistry, literature, history and English, with coefficients (proportions) of -0.806, -0.674, -0.675, 0.893, 0.825, 0.836.

Let $X_1, \cdots, X_6$ to represent the original six variables, and $Y_1, \cdots, Y_6$ to represent the new principal component. Then, the relationship between $X_1, \cdots, X_6$ and the first and second principal components Y_1, Y_2 is as follows:

$$\begin{cases} X_1 = -0.806Y_1 + 0.353Y_2 \\ X_2 = -0.674Y_1 + 0.531Y_2 \\ X_3 = -0.675Y_1 + 0.513Y_2 \\ X_4 = 0.893Y_1 + 0.306Y_2 \\ X_5 = 0.825Y_1 + 0.435Y_2 \\ X_6 = 0.836Y_1 + 0.425Y_2 \end{cases}$$

These coefficients are the principal component loadings, which represent the correlation coefficients between the principal components and the corresponding original variables. The greater the absolute value of the correlation coefficient, the greater the representativeness of the principal component to the variable. It can be seen that the first principal component explains all variables very well. The last principal component and the original variable are less relevant.

4. Conclusion

PCA is one of the unsupervised learning algorithms, which can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning). Machine learning takes a much more efficient algorithm in PCA, rather than the traditional statistical approach. Machine learning and statistics are closely related fields. The ideas of machine learning, from methodological principles to theoretical tools, have had a long pre-history in statistics. Leo Breiman distinguished two statistical modeling paradigms: data model and algorithmic model, wherein "algorithmic model" means more or less the machine learning algorithms like Random forest[8]. Some statisticians have adopted methods from machine learning, leading

to a combined field that they call statistical learning[9]. Recently, deep Learning is a new area of machine learning research, which has been introduced with the objective of moving machine learning closer to one of its original goals: Artificial Intelligence, and has attracted much attention from scholars and research institutions [10].

References

[1] C. Farabet, C. Couprie, L. Najman, Y. LeCun: *Learning hierarchical features for scene labeling.* IEEE transactions on pattern analysis and machine intelligence *35* (2013), No. 8, 1915–1929.

[2] Q. V. Le, M. Ranzato, R. Monga: *Building high-level features using large scale unsupervised learning.* IEEE international conference on acoustics (2013).

[3] Y. Bengio: *Deep learning of representations for unsupervised and transfer learning.* ICML unsupervised and transfer learning *27* (2012), No. 3, 17–36.

[4] Y. Bengio, A. Courville, P. Vincent: *Representation learning: A review and new perspectives.* IEEE transaction on pattern analysis and machine intelligence *35* (2013), No. 8, 1798–1828.

[5] V. Mnih: *Human-level control through deep reinforcement learning.* Nature *518* (2015), No. 7150, 529–533.

[6] E. J. Cands, X. Li, Y. Ma, J. Wright: *Robust principal component analysis.* Journal of the ACM *58* (2011).

[7] T. Bouwmans, E. Zahzah: *Robust PCA via principal component pursuit: a review for a comparative evaluation in video surveillance.* Special issue on background models challenge (2014).

[8] L. Breiman: *Statistical modeling: the two cultures.*Statistical science *16* (2001), No. 3, 199–231.

[9] G. James, D. Witten, T. Hastie, R. Tibshirani: *An introduction to statistical learning. Springer.*New York (2013).

[10] J. Schmidhuber: *Deep learning in neural networks.*An overview. Neural networks *20*, (2014), No. 61, 85–117.

Received November 16, 2017