S21CSE471: Statistical Methods in AI **Spring 2021**

Principal Component Analysis

Prepared by: 2020202013,2020201051,2020201058

Principal Component Analysis (PCA) is one of the most popular and widely used dimension reduction techniques to transform the larger dataset into a smaller dataset by identifying the correlations and patterns with preserving most of the valuable information.It is a unsupervised technique.The primary aim of Principal Components Analysis (PCA) is the dimension reduction. In this note, we start with basic information about eigen vectors and eigenvalues which will be useful for Principal Components Analysis (PCA).

1 Linear Dimensionality Reduction

Let us start with a feature representation x corresponding to a physical phenomena.These features are either what we could think of as relevant or what is feasible in practice. Their can be redundancy and correlation within these features too. They may not be the best for our problem also. A problem of interest to us is to find a new feature representation z, often lower in dimension, as $z = U * x$. If x is a d-dimensional vector and z is a k-dimensional vector (where $k < d$), U is a $k * d$ matrix.

This dimensionality reduction makes the downstream computations more efficient. In some cases storage/memory is also made efficient through it. It also results in speedup in model training due to faster computation due to reduced dimensionality attributed to the removal of unwanted columns in the data. Models finds it easy to learn and train from structured data.

When we do the dimensionality reduction, we may be removing useful information (or noise). Due to this, there is a small chance that we may reduce the accuracy of the model when performing downstream tasks like SVM classification.If "noise" (irrelevant information) is removed or suppressed in the entire process, we may expect some increase in the accuracy. At the same time, if we loose some "relevant information", then we may loose the accuracy.

If our original x is "raw-data" (let's say an image is represented as a vector), then such techniques are treated as principled ways to define and extract the features.It is used to overcome feature redundancy in the dataset. Also, it aims to capture valuable information explaining high variance which results in providing the best accuracy.It can also be implemented by only keeping the most relevant variables from the original dataset (this technique is called feature selection).

It makes the data visualizations easy to handle. In general, dimensionality reduction schemes aim at no major reduction in accuracy, but in a lower dimension.Even in some cases the accuracy on test data is increased(which generally decreases in most cases after dimensionality reduction) indicating a clear win-win situation for us.

2 Need of PCA and its Precursors

PCA 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.It takes care of multicollinearity by removing redundant features.

For example, you have two variables – 'time spent on treadmill in minutes' and 'calories burnt'. These variables are highly correlated as the more time you spend running on a treadmill, the more calories you

will burn. Hence, there is no point in storing both as just one of them does what you require.If PCA were not invented till date then the different methods which could be applied for dimension reduction are as follows:

- By finding the co-relation among the variables: This can be done by computing Co-variance matrix of dimension $d * d$ for a d- dimension data.

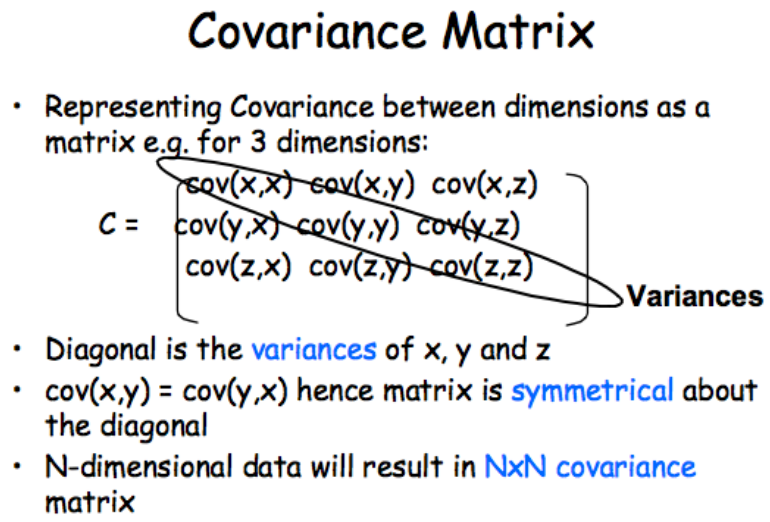

Figure 1: Co-variance Matrix for 3 dimensions.

Each value of that matrix will represent the correlation between the two variables while the diagonal will represent the variance of variables.The correlation matrix is symmetric, because it measures only a linear dependence between the pairs of variables. Large values in this matrix indicate serious collinearity between the variables involved. However, the nonexistence of extreme correlations does not imply lack of collinearity.

- By Classification: Let suppose we will begin with 1-d out of given 100-d data and lets an SVM learn on it.The one dimension with more accuracy we will keep it as it will be more important. Then we will go to 2-d.Her, also we will perform similar operation and pick up the more important dimension.So, this approach will only be useful when the original dimensions are not very large, hence, a naive approach.
- By dropping the dimensions: Here, we will start at the highest dimension lets say 100-d.Then will keep dropping the dimension one by one and keep checking the accuracy until we find the desired one.

3 Intuition

Lets suppose we have a 1-d data (or almost 1-d data) as shown in figure 2 drawn below.

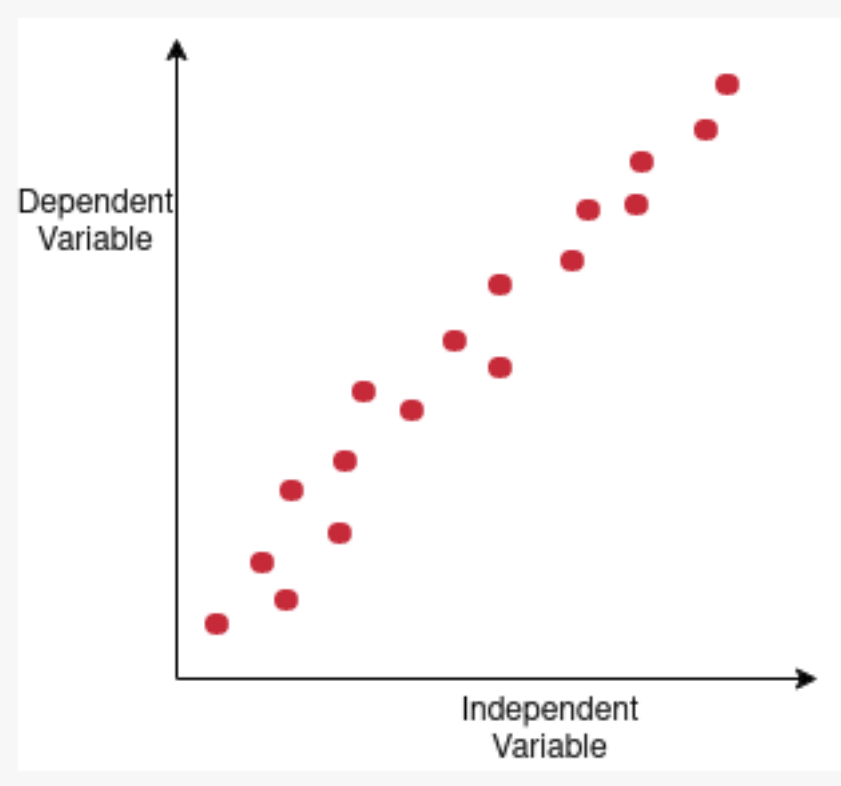

Figure 2: 1-d data Scatter plot.

Inorder to recover this almost 1-d data we will project the data to make it 1-d. To project we need to find the projection(line of data) on the x-axis and the y-axis. Let the direction of line be u_1.So, the direction vector will be $\vec{u}_1$ as shown in figure 3.

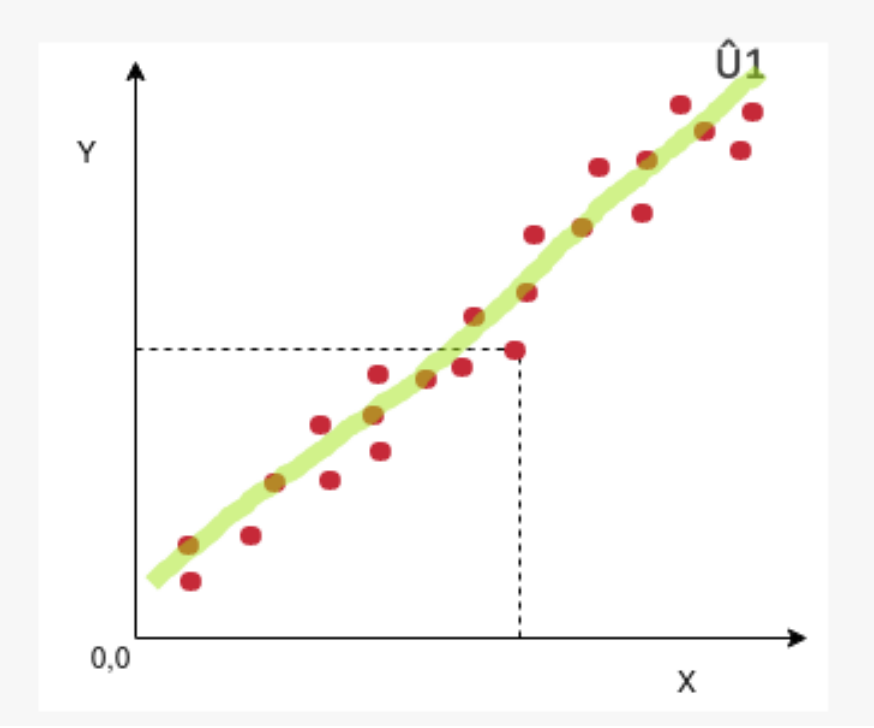

Figure 3: Direction Vector $\vec{u}_1$.

The variance of the data will be maximized in the direction of vector $\vec{u}_1$.

4 Eigen vectors and Eigen values

In this section, we will discuss basic concepts of Linear Algebra about eigen vectors and eigenvalues which will be useful for Principal Components Analysis (PCA).

For any matrix A_{n*n}, its eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ and eigen vectors $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n$ determine the amount by which a vector is scaled in the direction given by each eigen vector and the scaling factor is given by eigenvalues. Greater the eigenvalue, the more a vector's component in the corresponding eigen vector's direction is stretched when multiplied with A_{n*n}.

Suppose we have a vector $\vec{u}$ that we need to multiply with A_{n*n} matrix that has eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ and eigen vectors $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n$.

Let components of $\vec{u}$ in directions of the eigen vectors be $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_n$. Also

$$\vec{u} = \vec{u}_1 + \vec{u}_2 + \cdots + \vec{u}_n$$

Now $A * \vec{u}$ will be:

$$A * \vec{u} = \lambda_1 \vec{u}_1 + \lambda_2 \vec{u}_2 + \cdots + \lambda_n \vec{u}_n$$

5 Co-variance matrix of data features

Suppose A_{d*n} is a matrix with features of data where each data point is arranged in a column.

Let M be the vector with mean of each feature.

The co-variance matrix will be

$$V = E[(A - M)(A - M)^T]$$

which is a $d * d$ matrix giving co-variance of features.

6 Derivation of reduction to Optimization Problem

Let $X = [X_1\ X_2\ \ldots\ldots\ X_n]_{d*n}$, where $X_i \in \mathbb{R}^d$

Let U is the dimension on which we want to project X to get Y that has maximum variance.

$Y = U^T X$ $Y = [Y_1\ Y_2\ \ldots\ldots\ Y_n]_{1*n}$

$Var(U^T X) = \mathbb{E}[(U^T X - \mathbb{E}[U^T X])(U^T X - \mathbb{E}[U^T X])^T]$

$since, \mathbb{E}[X] = \mu$

$therefore, \mathbb{E}[U^T X] = U^T \mu$

$Var(U^T X) = \mathbb{E}[(U^T X - U^T \mu)(U^T X - U^T \mu)^T]$

$Var(U^T X) = U^T \mathbb{E}[(X - \mu)(X - \mu)^T] U$

$Var(U^T X) = U^T SU$, where $S = \mathbb{E}[(X - \mu)(X - \mu)^T]$

Now reduced to a standard Optimisation Problem :
It can be solved using the Lagrange's Multiplier Method. We will directly write the results here as it will be out of scope to derive the $J(U)$ relation here.

$$\underset{U_1}{max}(U_1^T SU_1), \ such \ that \ U_1^T U = 1$$

$$J(U) = U_1^T SU_1 - \lambda(U_1^T SU_1 - 1)$$

$$J(U) = 2SU_1 - 2\lambda U_1$$

$$2SU_1 - 2\lambda U_1 = 0$$

$$SU_1 = \lambda U_1$$

$$\underset{U_1}{max}(U_1^T SU_1)$$

$$\underset{U_1}{max}(U_1^T \lambda U_1)$$

$$\underset{U_1}{max}(\lambda U_1^T U_1)$$

$$\underset{U_1}{max}(\lambda), \ since \ U_1^T U = 1$$

Therefore, the best dimension to project so as to maximize the variance is the eigen vector corresponding to the largest eigen value. The second best will be the second largest one and so on.

7 Principal Components Analysis Algorithm

The prerequisites for PCA are ready.

Since V is a $d * d$ matrix, we can find its eigenvalues and eigen vectors using Singular Value Decomposition(SVD).

Let the eigenvalues arranged in non-increasing order be $\lambda_1, \lambda_2, \ldots, \lambda_d$ and the corresponding eigen vectors will be $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_d$.

These eigen vectors are the Principal Components and the amount of variance (or information) they capture is directly measured by their respective eigenvalue. More the eigenvalue compared to other eigenvalues, more the variance captured in the Principal Component.

The Principal Components are independent of one another as eigen vectors are perpendicular which is what is needed with PCA.

The fraction of variance captured by PC_m is given by $\frac{\lambda_m}{\sum_{i=1}^{d} \lambda_i}$.

Finally we must reorient the data to the principal components. Suppose that m out of d components are chosen. Let PM_{d*m} be the matrix where the principal components are arranged along columns.

Then the new data set will be:

$$A' = PM^T * A$$

.

Here A' will be an $m * d$ matrix of features of data reoriented along principal components.

7.1 Algorithm

- Center the data by subtracting the mean μ.
- Compute the co-variance matrix.
- Compute eigen values and eigen vectors of the above computed co-variance matrix.
- Take the p eigen vectors corresponding to the largest p eigen values. Keep the eigen vectors as the rows and create a matrix U of $p * d$.
- Compute the reduced dimensional vectors as $z_i = U * x_i$

7.2 How many dimensions should we select?

We have a ratio of eigen values that are covered by first p values out of total d values.

$$r_p = \frac{\sum_{i=p+1}^{d} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$$

This is also an estimate of how much information is lost in doing the dimensional reduction.

8 Applications of PCA

8.1 Improves Virtualisation

- It is very hard to visualize and understand the data in high dimensions. PCA transforms a high dimensional data to low dimensional data (2 dimension) so that it can be visualized easily.
- We can use 2D Scree Plot to see which Principal Components result in high variance and have more impact as compared to other Principal Components.
- Even the simplest IRIS dataset is 4 dimensional which is hard to visualize. We can use PCA to reduce it to 2-dimension for better visualization.

8.2 Face Recognition and Face representation

A classical application of PCA is in face recognition and face representation.Let us assume that we are given N face images each of $\sqrt{d} * \sqrt{d}$. We can visualize this as a d dimensional vector. Assume that our

input is all the faces from a passport office. All the faces are approximately of the same size. They are all frontal. And also expression neutral.

- Let us assume that mean μ is subtracted from each of the face. If all of the inputs were faces, then the mean is also looking very similar to a face.
- Let the input be $x_1...x_N$ be the mean subtracted faces and X be the $d*N$ matrix of all these faces.
- We are interested in finding the eigen vectors of the matrix XX^T . Say they are $u_1...u_k$. Note that k ¡= min(N, d).
- We then project each of the inputs x to these new eigen vectors and obtain a new feature. $z_i = u_i^T xi = 1, ..., k$
- The new representation z is obtained like this. You can also look at this as forming a $k*d$ matrix U which has its rows eigen vector.
- We obtain now a set of compact (k dimensional) representation z_i for each of the input face images x_i , where i = 1, . . . , N

8.3 Neuroscience

- A technique known as spike-triggered co-variance analysis uses a variant of Principal Components Analysis in Neuroscience to identify the specific properties of a stimulus that increase a neuron's probability of generating an action potential.
- PCA as a dimension reduction technique is used detect coordinated activities of large neuronal ensembles. It has been used in determining collective variables, that is, order parameters, during phase transitions in the brain.

8.4 Quantitative Finance

PCA is a methodology to reduce the dimensionality of a complex problem. Say, a fund manager has 1000 stocks in his portfolio. To analyze these stocks quantitatively a stock manager will require a co-relational matrix of the size 1000 * 1000, which makes the problem very complex.// However if he was to extract, 10 Principal Components which best represent the variance in the stocks best, this would reduce the complexity of problem while still explaining the movement of all 1000 stocks. Some other applications of PCA include:

- Analyzing the shape of the yield curve
- Implementation of interest rate models
- Developing asset allocation algorithms
- Developing long short equity trading algorithms
- Forecasting portfolio returns

8.5 Anomaly detection

PCA can be used in network anomaly detection. In the time-link matrix X, x_{ji} represents the amount of traffic on link j in the network during time-interval i. In the two pictures on the left, traffic appears periodic and reasonably deterministic on the selected principal component, which asserts that these two are normal behaviors. In the contrast, traffic "spikes" in the pictures on the right, which indicates anomalous behavior in this flow.

8.6 Part of Speech tagging

Unsupervised part-of-speech tagging is a common task in natural language processing, as manually tagging a large corpus is expensive and time-consuming. Here it is common to model each word in a vocabulary by its context distribution, i.e., x_{ji} is the number of times that word i appears in context j. The key idea of unsupervised POS tagging is that words appearing in similar contexts tend to have same POS tags. Hence, a typical tagging technique is to cluster words according to their contexts. However, in any given corpus, any given context may occur rather infrequently (the vectors x_i are too sparse), so PCA has been used to find a more suitable, comparable representation for each word before clustering is applied.

8.7 Latent Semantic Analysis

Another application of PCA is in text analysis. Let d to be the total number of words in the vocabulary; then each document x i R^d is a vector of word counts, and x_{ji} is the frequency of word j in document i. After we apply PCA here, the similarity between two documents is now $z_i^T z_j$, which is often more informative than the raw measure $x_i^T x_j$. Notice that there may not be significant computational savings, since the original word-document matrix was sparse, while the reduced representation is typically dense.

8.8 Multi-task learning

In multi-task learning, one is attempting to solve n related learning tasks simultaneously, e.g., classifying documents as relevant or not for n users. Often task i reduces to learning a weight vector x_i which produces for example the classification rule. Our goal is to exploit the similarities amongst these tasks to do more effective learning overall. One way to accomplish this is to use PCA is to identify a small set of eigen-classifiers among the learned rules $x_1, ..., x_n$. Then, the classifiers can be retrained with an added regularization term encouraging each x_i to lie near the subspace spanned by the principal directions. These two steps of PCA and retraining are iterated until convergence. In this way, low-dimensional representation of classifiers can help to detect the shared structures between independent tasks.

9 PCA limitations and extensions

While PCA has a great number of applications, it has its limitations as well:

- Squared Euclidean reconstruction error is not appropriate for all data types. Various extensions, such as exponential family PCA, have been developed for binary, categorical, count, and non-negative data.
- PCA can only find linear compression's of data. Kernel PCA is an important generalization designed for non-linear dimensional reduction.

9.1 Extensions of PCA

9.1.1 Multi-linear Principal Component Analysis

MPCA is a multi-linear extension of principal component analysis (PCA). MPCA is employed in the analysis of n-way arrays, i.e. a cube or hyper-cube of numbers, also informally referred to as a "data

tensor".Tensor methods that aim to solve three important challenges typically addressed by PCA: dimensional reduction, i.e., low-rank tensor approximation; supervised learning, i.e., learning linear sub-spaces for feature extraction; and robust low-rank tensor recovery.

9.1.2 Bayesian Principal Component Analysis

BPCA a probabilistic reformulation of PCA with Bayesian model selection, is a systematic approach to determining the number of essential principal components (PCs) for data representation. However, it assumes that data are Gaussian distributed and thus it cannot handle all types of practical observations, e.g. integers and binary values.

9.1.3 Kernel Principal Component Analysis

Kernel PCA will have all the advantages of the regular PCA, as well as the implicit nonlinear mapping to a feature space $\mathcal{F}$ where the features representing the structure in the data may be better extracted. For example, data classification may be much more easily carried out in $\mathcal{F}$ due to linear separability.

Algorithm

- Choose a kernel mapping $k(\mathbf{x}_m, \mathbf{x}_n)$.
- Obtain $\mathbf{K}$ based on training data $\{\mathbf{x}_n, (n = 1, \cdots, N)\}$.
- Solve eigenvalue problem of $\mathbf{K}$ to get λ_i and $\mathbf{a}_i$.
- For each given data point $\mathbf{x}$, obtain its principal components in the feature space: $(f(\mathbf{x}) \cdot \phi_i) = \sum_{n=1}^{N} a_n^{(i)} k(\mathbf{x}, \mathbf{x}_n)$
- Do whatever processing (e.g., feature selection, classification) in the feature space $\mathcal{F}$.

A principal advantage of this technique is that one can carry out non-linear dimension reduction with little dependence on the dimension of the non-linear feature space. However, one has to form and operate on a n × n matrix (which can be quite expensive). It is common to approximate the kernel when n is large using random or deterministic low-rank approximations.

9.1.4 Sparse Principal Component Analysis

Sparse principal component analysis (sparse PCA) is a specialised technique used in statistical analysis and, in particular, in the analysis of multivariate data sets. It extends the classic method of principal component analysis (PCA) for the reduction of dimensionality of data by introducing sparsity structures to the input variables.

References

[1] Applications of PCA:
`https://web.stanford.edu/ lmackey/stats306b/doc/stats306b-spring14-lecture8`$_s$*cribed.pdf*

[2] `https://en.wikipedia.org/wiki/Principal`$_c$*omponent*$_a$*nalysis*

[3] `https://iq.opengenus.org/applications-of-pca/`

[4] `https://www.i2tutorials.com/what-are-the-pros-and-cons-of-the-pca/`

[5] More Math in Latex, Link:
https://ctan.math.illinois.edu/info/Math_into_LaTeX-4/Short_Course.pdf

[6] PCA Lecture

[7] Extensions of PCA:
https://ieeexplore.ieee.org/document/8417937

[8] http://fourier.eng.hmc.edu/e161/lectures/kernelPCA/node4.html

[9] Images:
https://stats.stackexchange.com/questions/241449/matrix-and-regression-model

[10] https://builtin.com/data-science/step-step-explanation-principal-component-analysis