Information Technology Course
Module Software Engineering
by Damir Dobric / Andreas Pech

Dimensionality Reduction with Principal Component Analysis

Trinh Nguyen Phuong Tran

***Abstract*—Looking into the last decade life of sciences revolution, together with tremendous high technologies and laboratory instrumentations, one of the most challenging obstacles that many problems facing is related to the high-dimensional data. For instance, to a single experiment, there are hundreds or thousands of measurements such as in image processing, mass spectrometry, time series analysis, internet search engines, and automatic text analysis. Nevertheless, in many cases, these datasets have much more variables than observations because these free variables are duplicated and dependent to each other. To deal with this challenge, Dimensionality Reduction Algorithms are introduced to compress the dataset as small as possible so that the execution time could be reduced, and it could be realized with more practical devices. This paper is addressed the Principal Component Analysis (PCA) [1], one of the most popular algorithms to reduce the dimension of the dataset. Moreover, using the experiment result, the efficiency of theorical method could be proved by an example application which is developed using .NET Core Technology [2].**

Keywords—pattern recognition, data mining, machine learning, principal component analysis, dimensionality reduction, LearningAPI, .NET Core.

I. Introduction

Dimensionality Reduction, as mentioned in the abstract, is one of the important techniques in Machine Learning. In many practical problems, not only the input datasets but also the feature vectors are signifcantly large, with the dimension up to several thousand variables. This high dimension leads to many negative consequences on execution time and computational resources for processing, calculating and storing the dataset. Therefore, reducing the number of data dimensions is an essential step in many problems. This is also considered a method of data compression.

Generally, Dimensionality Reduction is to find a function which takes the input $x \in \mathbb{R}^D$ where D is large in order to create the new dataset $z \in \mathbb{R}^K$ where K < D.

One of the simplest methods for Dimensionality Reduction based on a linear model is Principal Component Analysis (PCA). This method is based on the observations that data is not randomly and independently distributed in space but is distributed in a certain line or face. PCA considers a special case where those certain lines or faces are linear and belong to its subspace.

The detail of PCA is expressed in the second chapter. Then, this paper will introduce the implementation in .NET Core. Then, In the chapter III, a couple of test cases are designed to confirm the correctness, robustness, and reliability of the program. Finally, the document is expected to show the experiment results and evaluation so that readers could consider this approach for their further implementation.

II. Methods

A. *Principal Component Analysis (PCA)*

Assume that there is a dataset x = $[x_1, x_2, .. x_D] \in \mathbb{R}^D$ and a coordinate unit system $e_1, e_2, .. e_D$. Data is illustrated in this system is:

$$x = x_1 e_1 + x_2 e_2 + .. + x_D e_D$$

Now the coordinate system is changed to $u_1, u_2, .. u_D$ then the data x in the new system is:

$$x = y_1 u_1 + y_2 u_2 + .. + y_D u_D$$

, where y is the projection of x on the new system. Because $\boldsymbol{U}$ is an invertible and orthogonal matrix, means $U^{-1} = U^T$. Then it could be concluded $\boldsymbol{y} = \boldsymbol{U^T x}$.

As above discussion, PCA is to find a new coordinate system U^K and feature vector y to perform x in the new system U^K, see in the Figure 1. The main point in this algorithm is that system U^K could be divided into two subspaces: more important space and less important space. For example, in the Figure 1, the grey domain should be excluded while keeping only the green domain. By implementing this idea to remove a part of an input object, the most important features of the object are remained enough to reconstruct it.

To evaluate the importance of a feature in a feature vector, a covariance matrix is introduced in this method. Data in multi-dimensional space is correlated. According to each direction, their expectations and variances could be calculated. The bigger the variance, the more scattered the data plots in that dimension, the more information that direction has. In the other words, PCA is derived from analyzing the principal components of the covariance matrix vector.

PCA steps [3] are summarized in the following part of this section:

1. Find the mean vector of the input data:

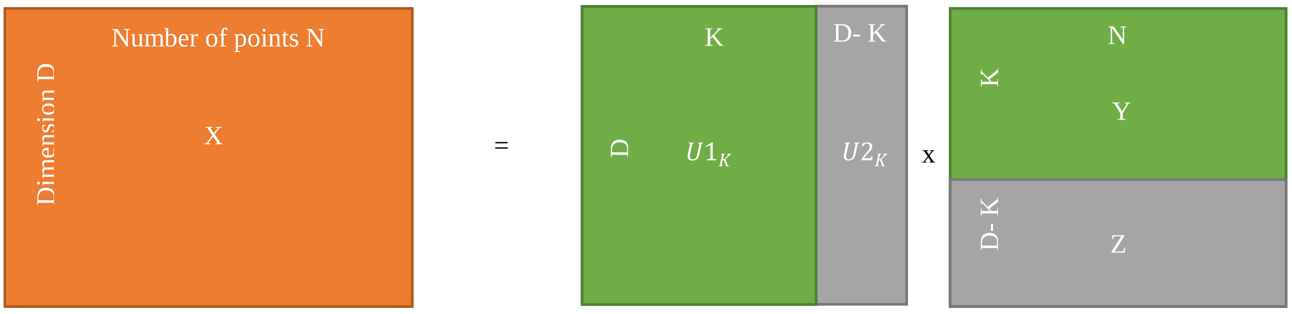


Figure 1. PCA - main idea

$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_N$$

2. Transform to the zero-corrected data:
$$\widehat{x_N} = x_N - \bar{x}$$
3. Find covariance matrix:
$$S = \frac{1}{N}\hat{X}\hat{X}^T$$
4. Compute eigenvalues and eigenvectors of the covariance matrix.
5. Sort the eigenvalues descending and choose only K most important elements.
6. Project the input data to the new subspace:
$$Y = U_K^T\hat{X}$$

As discussed in the above section, the target output of PCA is a data set Y which has a lower dimension than the original data set X.

B. *Project Structure*

In the solution directory, there are two projects created for implementing and testing. The projects rely on C# programming language, .NET Core 2.0 SDK, and Microsoft Visual Studio 2017 IDE. As illustrated in the Figure 2, the algorithm implementation project is to define the PCA Algorithm with the extension to integrate with

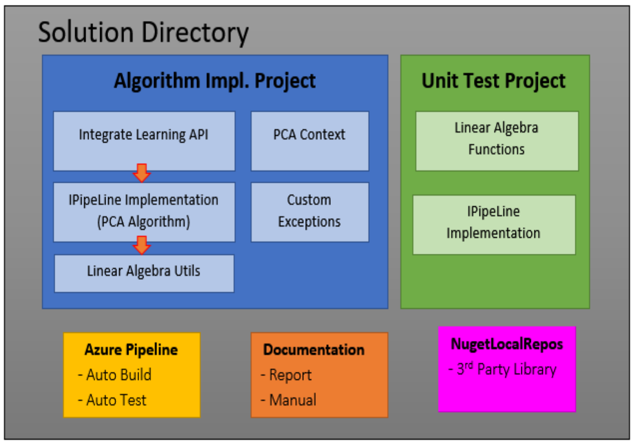


Figure 2. Project Structure

LearningAPI. The algorithm uses a couple of linear algebra computation which is derived from a separate wrapper class.

In the current solution, the Linear Algebra wrapper uses Math.NET library for these linear algebraic operations. This class is extended from an abstract class where all linear algebraic functions are defined. Because both Test project and PCA implementation project are calling the abstract class, it should be efficient when users would like to define another algorithm or use another library without invasion on the existed classes. Beside the main part, there are several supportive classes, which help to transform data, read data from csv or bitmap image.

In the green block, we define the Test Project. There are two testing strategies. Firstly, three test cases are defined for each function in Linear Algebra wrapper class to test all functions. These are test to prof that the functions return correct value, test to prof that the functions don't hard-coded the value, test to prof that the functions are able to deal with corrupted data. Then, we define a test for testing the whole algorithm and produce example data. Finally, there is a test which help to explain how the module could be integrated into the current LearningApi.

Three blocks below are used as the supplements and utilities. The Azure Pipeline is used to run build, test and packaging automatically for every single commit. This establishment is highly recommended to save the development time and increasing quality. Moreover, the Documentation is a folder containing reports, manuals, and presentations about the algorithm and the project. The NugetLocalRepos contains the 3rd Party Libraries dependencies which are used in the project.

C. *Development Strategy*

The development strategy for this project follows a robust proven process – Test Driven Development. In the development process, all test cases should be defined at design time, hence, developers could focus on the most

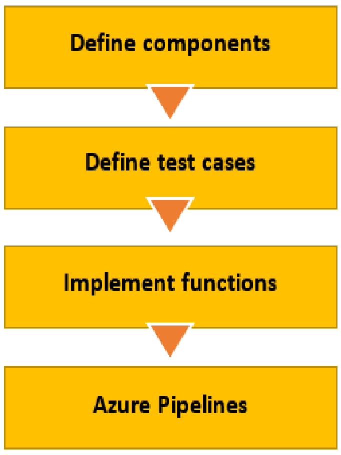


Figure 3. Implementation Process

necessary functions, and always tried to in mind all corner cases, which may cause problems. This style reduces the development cost and make the program becomes more flexible, agile, and clean. The implementation process includes not only program application but also building, testing, packaging, and deploying, which are controlled by Azure Pipelines service. The figure 3 is proposed as a summary for describing this process.

1) *Define main components*

The.re are two important components defined in this project an abstract class `GeneralLinearAlgebraUtils` which provide an abstract component for other components to call to the Linear Algebra implementation, and a `PCAPipelineModule` component, which serve as the API to use this algorithm. The PCA component then relys on the Linear Algebra component and follow the steps discussed in Chapter 2-section A. Also, the PCA component is defined with the extension of `LearningApi`, then it could be integrated to Learning API later. Because both the PCA component and Test component plug to the abstract class Linear Algebra, it could be efficient if the user would like to change the implementation of the abstract class without make any change on the existed class.

2) *Defining test cases*

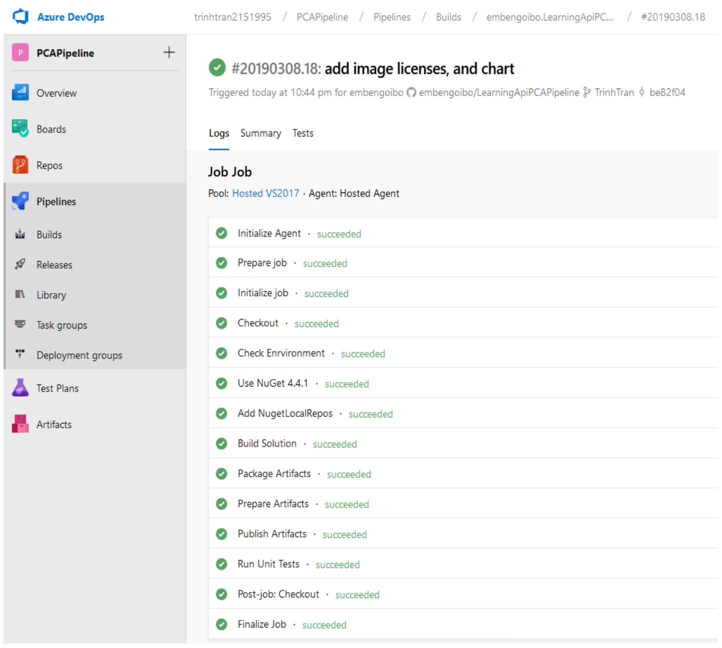
After having a list of all functions which the API should perform to get the output of PCA, the next step is to define the test cases for it. In this project, there are three test types. Firstly, there are 34 test cases used to test the correctness of all linear algebra functions. Secondly, the correctness, robustness, and reliability of PCA is tested with the actual data. In this case, we use AT&T Face database which is the popular and freely available for non-commercial usage. Finally, the library is integrated and tested with the Learning API.

3) *Implement linear algebra functions*

Based on the test cases above, the implementation `MathNetNumericsLinearAlgebraUtils` which provide the implementation for the abstract class `GeneralLinearAlgebraUtils` as discussed above. This class uses Math.NET library for the most complicated operators such as Singular Value Decomposition or eigen vectors decomposition. In this class, a couple of matrices, vectors, scalars operators are defined and follow the definition in its `GeneralLinearAlgebraUtils` as well.

4) *Run tests*

Test are run automatically by Azure Pipelines, after every commit to source control service. Azure Pipelines is an Azure DevOps service where the software could be continuously built, tested, and deployed to any platform and cloud. It helps to handle with the time consuming in development process. To implement Azure Pipelines, we recommend following a YAML schema. For more detail, refer to [4]. The figure 4 is the Azure Pipeline which displays all jobs as well as their statuses.


Azure DevOps   trinhtran2151995 / PCAPipeline / Pipelines / Builds / embengoibo.LearningApiPC... / #20190308.18

#20190308.18: add image licenses, and chart

Triggered today at 10:44 pm for embengoibo embengoibo/LearningApiPCAPipeline TrinhTran be82f04

Job Job

Pool: Hosted VS2017 · Agent: Hosted Agent

```
Initialize Agent · succeeded
Prepare job · succeeded
Initialize job · succeeded
Checkout · succeeded
Check Enrvironment · succeeded
Use NuGet 4.4.1 · succeeded
Add NugetLocalRepos · succeeded
Build Solution · succeeded
Package Artifacts · succeeded
Prepare Artifacts · succeeded
Publish Artifacts · succeeded
Run Unit Tests · succeeded
Post-job: Checkout · succeeded
Finalize Job · succeeded
```

Figure 4. Azure Pipeline Layout

D. UnitTest

1) *Linear Algebra Functions*

One of the most critical prerequisites which decide the correctness of the algorithm is the linear algebra functions. In the Unit Test project, there are three cases defined for each operation. Firstly, the project aims to inject a correct expectation and compare to the result going out from the function. Secondly, it continues injecting the wrong expectation and expect the result should not equal to these wrong values, this test aims to check whether the method is using hard-coded data or not. Finally, all functions are tested with corrupted data, for example data with wrong dimension. When injecting a wrong dimension input, it is expected to receive an exception return. To a basic and simple algebra operations, three testing steps could be enough to prove their correctness.

2) *Integration Test with LearningAPI*

`LearningAPI` uses a list of pipeline components which implements `IPipelineModule`. Hence, in order to integrate with `LearningApi`, `PCAPipelineModule` is also an implementation of `IPipelineModule`. When being run, `LearningAPI` will execute pipeline modules sequentially. Each pipeline component will receive data from the prior component, do computational task and return data to the next pipeline component in the list. In order to test the integrating of `PCAPipelineModule` with `LearningApi` module, we provide an extension method to `LearningApi` called `UsePCAPipelineModule`(). When this method is called, it will inject an instance of `PCAPipelineModule` to the pipeline module list of `LearningApi` with a unique name.

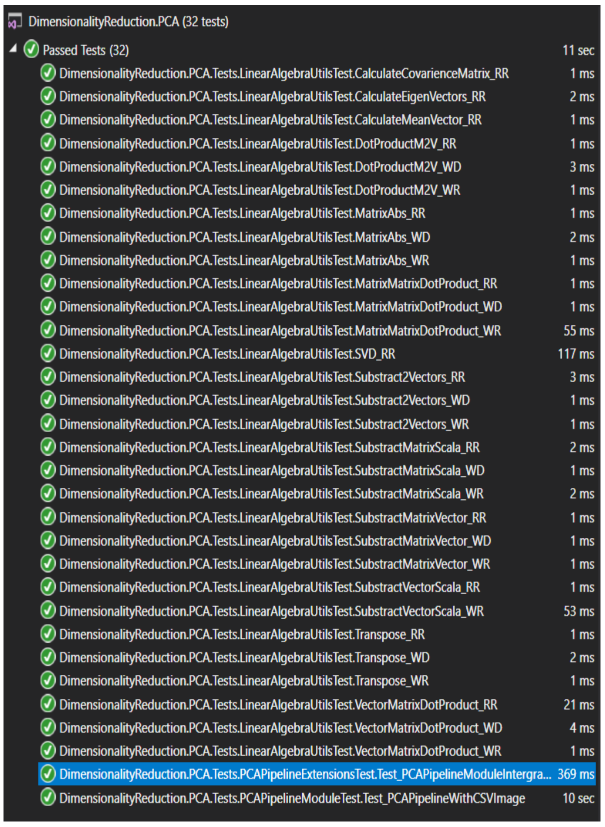
III. Results

This part presents the result after applying the testing strategies, in Chapter 2-section D, on the algorithm, in

Chapter 2-section C. Based on this result, the quality of this application could be proved and further improved.

A. *Linear Algebra Functions*

As discussed in the above section, the main algorithm is developed from many algebra operators and functions such as covariance matrix computation, matrix vector dot product, etc. Each function should be tested once with a true expectation, a wrong expectation and a wrong dimension. In summary, there are 32 test cases developed and run successfully with XUnit. These test cases are listed in the figure 5 below.



```
DimensionalityReduction.PCA (32 tests)
Passed Tests (32)                                                                      11 sec
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.CalculateCovarienceMatrix_RR   1 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.CalculateEigenVectors_RR       2 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.CalculateMeanVector_RR         1 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.DotProductM2V_RR               1 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.DotProductM2V_WD               3 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.DotProductM2V_WR               1 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.MatrixAbs_RR                   1 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.MatrixAbs_WD                   2 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.MatrixAbs_WR                   1 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.MatrixMatrixDotProduct_RR      1 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.MatrixMatrixDotProduct_WD      1 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.MatrixMatrixDotProduct_WR     55 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.SVD_RR                       117 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.Substract2Vectors_RR           3 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.Substract2Vectors_WD           1 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.Substract2Vectors_WR           1 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.SubstractMatrixScala_RR        2 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.SubstractMatrixScala_WD        1 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.SubstractMatrixScala_WR        2 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.SubstractMatrixVector_RR       1 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.SubstractMatrixVector_WD       1 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.SubstractMatrixVector_WR       1 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.SubstractVectorScala_RR        1 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.SubstractVectorScala_WR       53 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.Transpose_RR                   1 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.Transpose_WD                   2 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.Transpose_WR                   1 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.VectorMatrixDotProduct_RR     21 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.VectorMatrixDotProduct_WD      4 ms
  DimensionalityReduction.PCA.Tests.LinearAlgebraUtilsTest.VectorMatrixDotProduct_WR      1 ms
  DimensionalityReduction.PCA.Tests.PCAPipelineExtensionsTest.Test_PCAPipelineModuleIntergra... 369 ms
  DimensionalityReduction.PCA.Tests.PCAPipelineModuleTest.Test_PCAPipelineWithCSVImage   10 sec
```

Figure 5. Unit test (32 test cases). This image is taken by Microsoft Visual Studio 2017

B. *PCA on AT&T Face database*

As the aforementioned approach, after testing all element functions, the whole algorithm should be tested with an actual data. AT&T Face database contains a set of faces taken between April 1992 and April 1994 at the Olivetti Research Laboratory in Cambridge, UK [5]. This is the most popular and freely available (for non-commercial usage) database which is usually used for PCA, especially for finding Eigenface. To apply the application on this database, it should be either converted from image format to CSV format or it should come with 24-bit greyscale image of format bmp.

From the figure 6, we apply PCA to compress an input image and define expectation losses: 1%, 3%, 5%, 10% respectively. It illustrates that, as less components we remain, as less quality the output has, but as more efficient regarding the processing time.

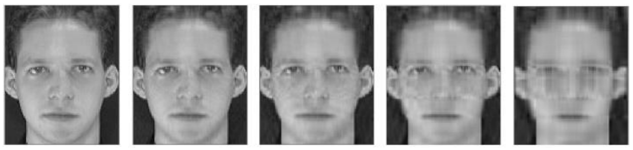


Figure 6. Reduce dimension to compress an image. From left to right: input image, loss 1%, loss 3%, loss 5%, loss 10%.

C. *Using with LearningApi*

Then we look into an example of integrating the algorithm library with the LearningApi as below.

```
LearningApi api = new LearningApi();
api.UseActionModule<object, double[][]>((notUsed, ctx) =>
{
    //This is an examle of how data
    //should be inject into PCAPipelineModule
    string filePath = Path.Combine(INPUT_DATA_DIR, "face1.csv");
    double[][] data = CsvUtils.ReadCSVToDouble2DArray(filePath, ',');

    //PCAPipelineModule receive data as double[][] array
    return data;
});

// Now we tell LearningApi to use a PCAPipelineModule instance,
// we named it "PCAPipelineModule-Ex1"
// The new data should only have the new dimensions with size 10
api.UsePCAPipelineModule(moduleName: "PCAPipelineModule-Ex1",
                         newDimensionSize: 10);

// or api.UsePCAPipelineModule(moduleName: "PCAPipelineModule-Ex1",
//                             maximumLoss: 0.05);
// to specify the maximum amount of loss that we want to have
// if we plan to use the result from PCA to estimate the orginal data.

// Now all modules will be run,
// output of the prior module
// will be pipe to the input of the next module
api.Run();

// After each calculation PCAPipelineModule will
// contains lots more useful information
// To retrieve this, we will use its name above
// to get back to module instance
PCAPipelineModule moduleInstance =
api.GetModule<PCAPipelineModule>("PCAPipelineModule");
double[][] explainedVariance = new double[2][];
explainedVariance[0] = moduleInstance.ExplainedVariance;
explainedVariance[1] = moduleInstance.CummulativeExplainedVariance;
```

D. *PCA Result Interpretation*

In order to interpret and show how effective PCA is, we calculate also two additional information: explained variances and cumulative explained variances.

Explained Variance of a component shows how many percent of original data could be reconstructed using only that component. Cumulative Variance of a component shows how many percent of original data could be reconstructed using that component and also all prior components. The figure 7 below, show the result on AT&T test database.

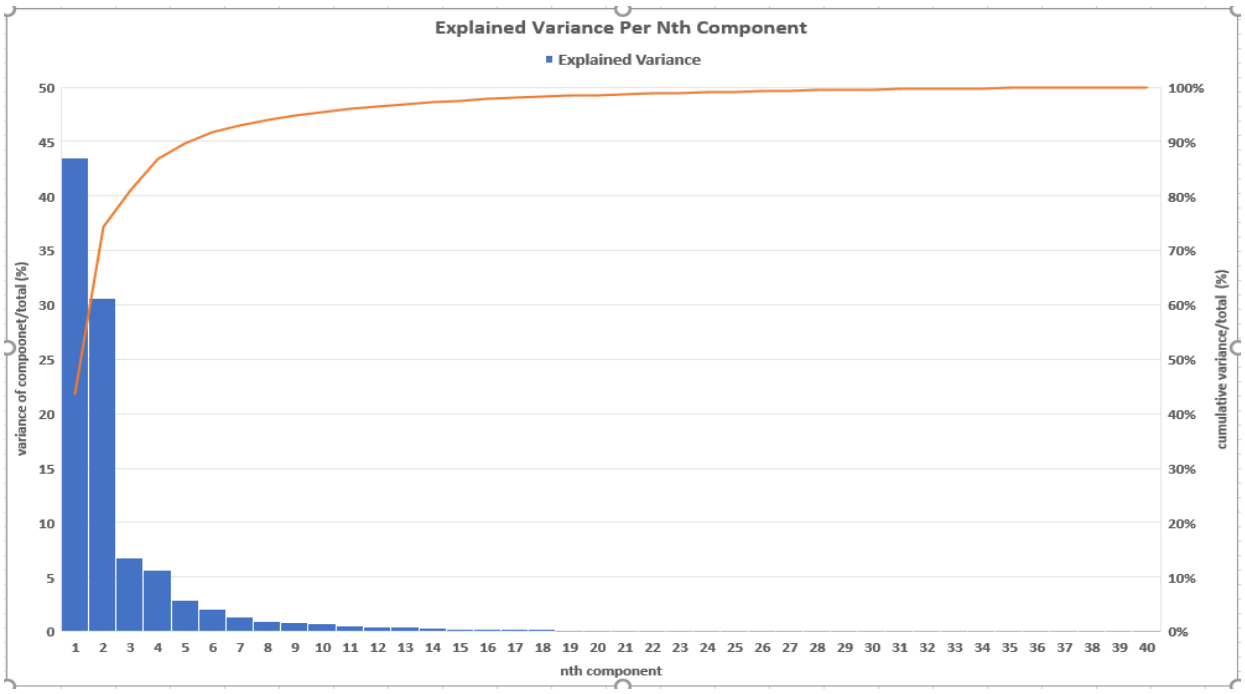


Figure 7. The number of components along with their importance to the data set. The blue column is the importance of nth component (descending). The red line is the level of output quality when cumulating from 1st component to nth component.

IV. Discussion and Conclusion

PCA Algorithm is used in many Machine Learning and Data Mining in case that the input data set is too large or sparse. The main idea of PCA is calculating the eigenvectors of the covariance matrix. Then, it chooses the most important impacts vectors and reduces a number of less important elements. Hence, the output of PCA has a lower dimension than the original one. In this project, with the support from Microsoft Visual Studio 2017, we learn how to use C# to develop a library which could be integrated in the Learning API. The solution is tested with the reasonable strategies to prove the correctness of the development. With the proposed project structure, users could define more class or change the libraries without an invasion on the existed classes. Moreover, in this project, we apply Azure Pipeline for automatically building, testing and packaging after each commit. This establishment upgrades the development process against the time consuming.

One of the limitations that developers could easily record is about the algorithm complexity. Because using PCA, user must calculate the eigenvectors which is hard to compute by hand, its results always depend on the 3rd party algorithm such as Singular Value Decomposition: different algorithm returns different results. Therefore, it is hard to cross check the result compared to different languages or libraries. The second limitation is that the implementation of PCA on all data does not depend on the class each data. PCA is an Unsupervised method. Hence, in this case, PCA is not effective for classification problems. For instance, in two-dimensional space, 2 classes are distributed along both sides of a straight line. There is a high probability to let PCA output the straight line as its principal component. Then projecting data on this line, both classes are mixed together and decrease the data classification. The third problem could happen to PCA is about devices memory. With a large-scale data set, the PCA calculation on the entire set is not feasible because the computers or processor are out of memory. These limitations could be resolved based on a certain data set characteristic.

In conclusion, PCA algorithm is compatible for student to start learning about Software Engineering development process. Besides its limitation, PCA is one of the most popular techniques using in many applications nowadays. It could be simple to find out an instruction and data base which are free and available on the Internet. Furthermore, PCA is complex enough to challenge we with a C# project but simple enough to set up a development process without investing too much effort.

V. References

[1] H. T. Vu, "Principal Component Analysis," 15 Jun 2017. [Online]. Available: https://machinelearningcoban.com/2017/06/15/pca/.

[2] O. Source, "UniversityOfAppliedSciencesFrankfurt/LearningApi: Machine Learning foundation on top of .NET Core," 2019. [Online]. Available: https://github.com/UniversityOfAppliedSciencesFrankfurt/LearningApi. [Accessed 2019].

[3] L. I. Smith, "A tutorial on Principal Components Analysis," February 26, 2002.

[4] O. Source, "YAML schema reference," 06 03 2019. [Online]. Available: https://docs.microsoft.com/en-us/azure/devops/pipelines/yaml-schema?view=azure-devops&tabs=schema. [Accessed 2019].