IN3062 module - Introduction to Artificial Intelligence

Coursework report

Jean-Louis Gosselin / Negar Vahab Pour

The ultimate goal of this coursework seemed enticing enough: to apply techniques specific to unsupervised learning such as PCA and K-means clustering on an unlabelled dataset, with the aim of classifying it.

But why choose *un*-supervised learning over supervised?

Although it must be very satisfying indeed to *teach* a machine to instantly identify and classify new unknown data based on recognised patterns and formulas (themselves based on past, *taught* data), the excitement of *discovering* first-hand (all thanks to the machine's own dimension-reducing and clustering abilities) hidden patterns and the overall inherent structure of any given data that contains no explicitly-provided labels is what fuelled our interest in and our will to conduct research into the field of unsupervised learning.

Compared with the sheer number of algorithms associated with supervised learning, there are only a handful algorithms in unsupervised learning to consider (PCA, K-means, Gaussian… - the first two of which will be used in our project). However, we hope that this coursework will demonstrate how challenging it has been to *jointly* implement PCA and K-means on an unlabelled dataset (not to mention the many intermediary steps required for instance to cleanse this dataset in order to give a better chance for both these techniques to work efficiently). We also hope that this report will sufficiently explain the overall techniques we have used to achieve our project – however partial or inconsistent the actual results of these operations[1] may appear at times.

PART 1 – the dataset

(a) Choice of data

We chose a dataset related to house sale prices (Kaggle, 2016).

However, to be absolutely honest, it didn't really matter what dataset we chose. What was important was to find a dataset that essentially befitted the use of PCA, meaning the following requirements were indispensable:

- a large number of columns/features (to implement dimension reduction)
- a variety of data types and NaN values (to demonstrate effective data cleansing techniques as well as effective reformatting of data into continuous data)
- lastly (and obviously), an unlabelled dataset

[1] *NB:* much Python code written in our script has been taken from multiple online sources, the links for some of which have been explicitly referenced in the Python script.

It is to be noted that the whole dataset (originally titled: “train set” on Kaggle) is used for this coursework: splitting data into training and testing sets does not generally apply to unsupervised PCA (no training data set *per se* exists).

(b) <u>Importance of data cleansing with regard to PCA</u>

Data cleansing took a substancial part of the coursework, and for good reason: dimension reduction in PCA can only be performed on continuous numeric data.
A variety of techniques were used to perform this data cleansing:

- eliminating all NaN values (in other words: replacing these with 0s)
- replacing certain persistent string values
- discarding all other non-numeric values
- converting all integer values to float values (as a precaution)

(c) <u>SALEPRICES column</u>

From inspecting the dataset, we happened to notice what seemed to be a relevant and important column (if not: *the* most important): the SALEPRICE column.

Indeed, the whole idea of conceiving a PCA-to-clustering-to-classifying project was borne of noticing the prices allocated to each house description, hoping this would have the machine find two main target patterns within that data structure (ie: a split in house prices ideally between “expensive” and “affordable”). This explains:

- the code which calculates the mean and average of this column
- the code that displays the distribution of house prices within the dataset
- the code that displays pie charts of the ratio of house sale prices

(d) <u>Data standardization / Min-Max scaling</u>

As stated on a webpage dedicated to PCA:

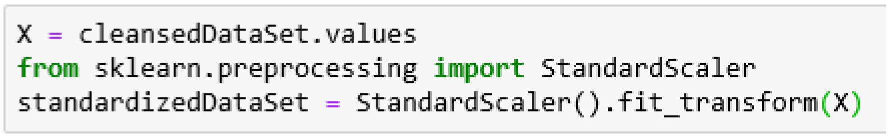
> In PCA variables are often scaled (i.e. standardized). This is particularly recommended when variables are measured in different scales (e.g: kilograms, kilometers, centimeters, …); otherwise, the PCA outputs obtained will be severely affected. The goal is to make the variables comparable. Generally variables are scaled to have i) standard deviation one and ii) mean zero. The standardization of data is an approach widely used in the context of gene expression data analysis before PCA and clustering analysis. We might also want to scale the data when the mean and/or the standard deviation of variables are largely different.
>
> (kassambara, 2017)

Furthermore:

> Standardizing the features so that they are centered around 0 with a standard deviation of 1 is not only important if we are comparing measurements that have different units, but it is also a general requirement for many machine learning algorithms.
>
> (Raschka, 2014)

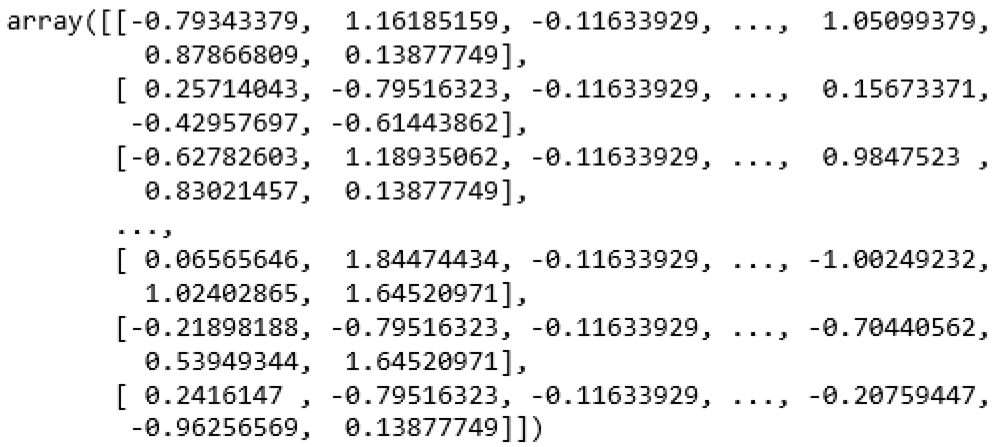
Data standardization (or normalization) is therefore an important step to successfully implementing PCA, since “PCA transformation is sensitive to the relative scaling of the original variables” (Silipo, 2015). However, as indicated in the script text, this part was purposefully set aside, as a section in its own right, since it is the tipping point of the entire coursework.

In short: the following implementation of standardization:



```
X = cleansedDataSet.values
from sklearn.preprocessing import StandardScaler
standardizedDataSet = StandardScaler().fit_transform(X)
```

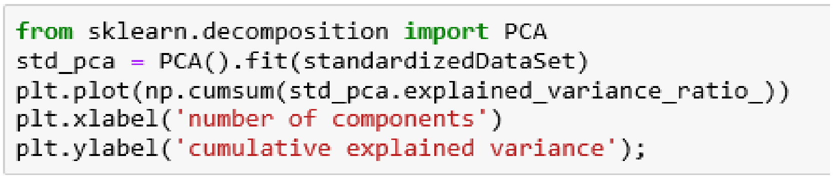
…produces the following values:



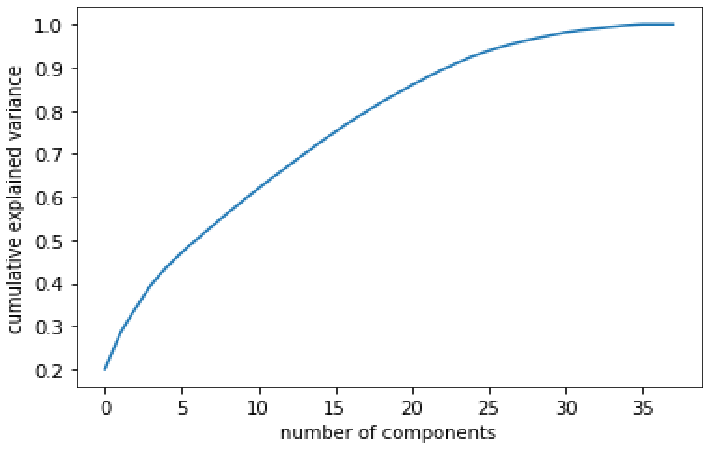
```
array([[-0.79343379,  1.16185159, -0.11633929, ...,  1.05099379,
         0.87866809,  0.13877749],
       [ 0.25714043, -0.79516323, -0.11633929, ...,  0.15673371,
        -0.42957697, -0.61443862],
       [-0.62782603,  1.18935062, -0.11633929, ...,  0.9847523 ,
         0.83021457,  0.13877749],
       ...,
       [ 0.06565646,  1.84474434, -0.11633929, ..., -1.00249232,
         1.02402865,  1.64520971],
       [-0.21898188, -0.79516323, -0.11633929, ..., -0.70440562,
         0.53949344,  1.64520971],
       [ 0.2416147 , -0.79516323, -0.11633929, ..., -0.20759447,
        -0.96256569,  0.13877749]])
```

…which seems to be in line with what we would expect for data standardization.

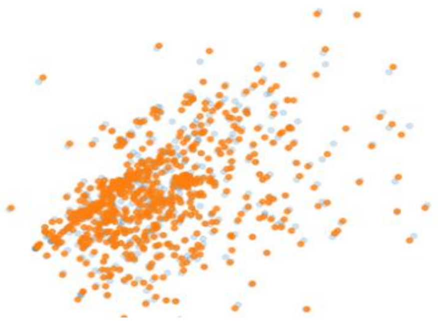
However, translating this into a scree plot designed to indicate the ideal number of components used for PCA gives us a very disappointing result. As shown in the graph below:



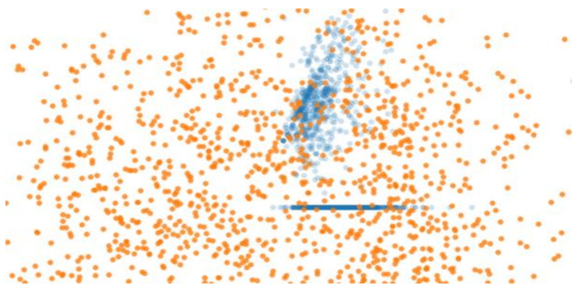
```
from sklearn.decomposition import PCA
std_pca = PCA().fit(standardizedDataSet)
plt.plot(np.cumsum(std_pca.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance');
```



…the final and decisive inflexion seems to occur at 35 components – shy of only 3 components from the initial total of 38 components, which is obviously insufficient. By fitting this standardized dataset to PCA, we get an even more disheartening juxtaposition between plotting of the PCA-transformed dataset (in orange) and that of the cleansed dataset (in blue – barely noticeable, since both virtually and quite logically overlap):



The same problem essentially occurs with Min-Max scaling (similar to standardization, but where "the data is scaled to a fixed range - usually 0 to 1" (Reschka, 2014). Although the scree plot reduces the breaking point down to 10 components, the resulting plotting is even more unexplainable:



It was therefore inevitable unfortunately to omit data standardization in our subsequent implementation of PCA, having to resort to the cleansed dataset instead (which won't retain most of the data information, i.e. the ***variation*** in the data).

PART 2 – PCA

As stated in this online article:

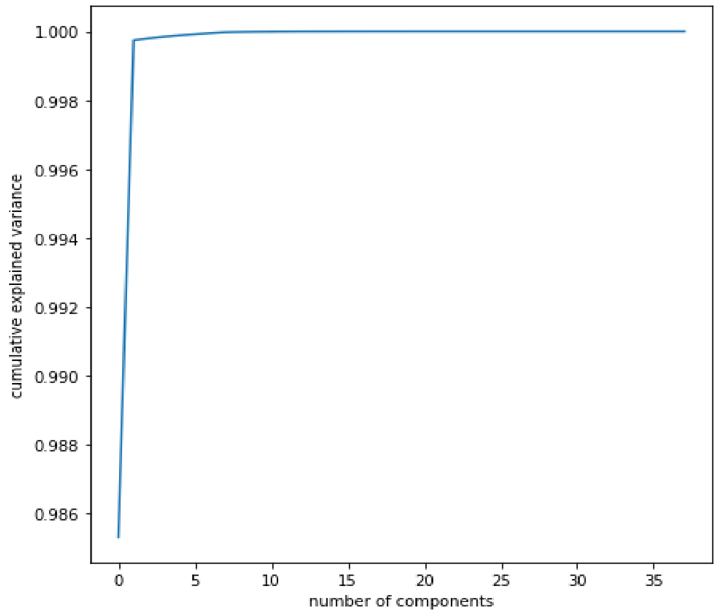
> High dimensionality means that the dataset has a large number of features. The primary problem associated with high-dimensionality in the machine learning field is model overfitting, which reduces the ability to generalize beyond the examples in the training set. Richard Bellman described this phenomenon in 1961 as the Curse of Dimensionality where "Many algorithms that work fine in low dimensions become intractable when the input is high-dimensional."

(Goonewardana, 2019)

Our 38-column cleansed dataset (originally: 81 columns) conveniently presents us with the urge to perform PCA upon it.

(a) Scree plot ("elbow curve")

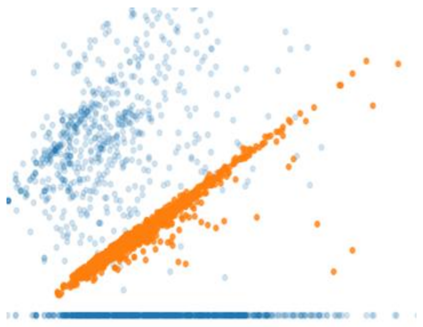
Implementing the scree plot using the cleansed dataset rather than the (expected) standardized dataset gives us a more realistic figure for the number of components required (ie: 2):



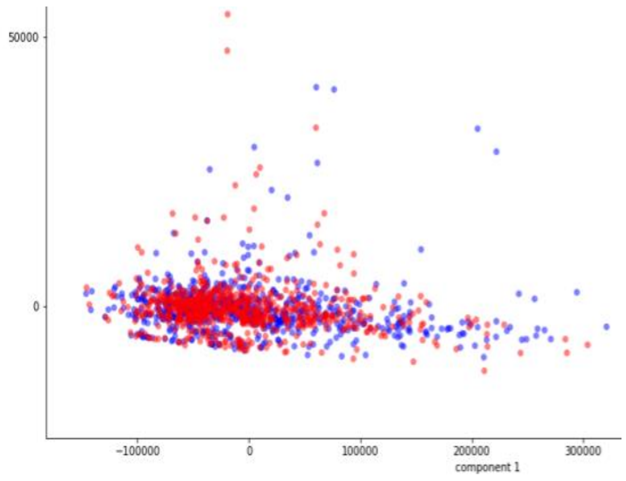
This "elbow curve" graph demonstrates that nearly 99% of the variance can be attributed to just 5% of all 38 features and 1% to the other 95% of these features, indicating high feature redundancy.

(b) PCA-tranformed dataset

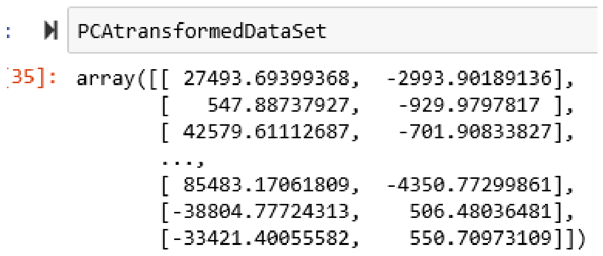
Implementing PCA based on this cleansed dataset (this time: with only 2 components giving us a more realistic plot graph), we then juxtapose both cleansed and PCA_transformed datasets:



Of course, we can notice the effects of not using a properly PCA-transformed, standardized dataset, since many PCA scores deviate (some: substantially) from the plotted line we would normally expect to see. These outliers are further graphically displayed in the axis-rotated diagram below:



This plotting seems to be very much in line however with the values of the PCA-transformed dataset itself, which is a good sign:



```
PCAtransformedDataSet

35]: array([[ 27493.69399368,  -2993.90189136],
            [   547.88737927,   -929.9797817 ],
            [ 42579.61112687,   -701.90833827],
            ...,
            [ 85483.17061809,  -4350.77299861],
            [-38804.77724313,    506.48036481],
            [-33421.40055582,    550.70973109]])
```

(c) <u>Comparing accuracy scores</u>

In order for us to verify that correlation between high dimensionality and PCA-induced low dimensionality, we have proceeded with a series of identical blocks of code, each taking a different value for the PCA() function. These functions range from the initial 38 components (from the cleansed dataset) down to 2 components (confirmed by the above scree plot), as so: 38-30-20-10-8-6-4-3-2. This gives us the following accuracy scores:

```
PCA(10) -> 0.2958904109589041
PCA(8) -> 0.30684931506849317
PCA(6) -> 0.336986301369863
PCA(4) -> 0.3424657534246575
PCA(3) -> 0.3917808219178082
PCA(2) -> 0.4575342465753425
```

As we can see, starting with PCA(10), accuracy *increases* when components *decrease*. This is quite baffling, since the logical outcome should have been the total opposite (referring, here, to the example given to us related to digit recognition). We have tried our best to interpret this result, and the only reason we could find for these baffling results is the lack of data standardization, discussed previously.

PART 3 – Clustering

As for determining the number of clusters and plotting clusters with centroids, the code seems to have implemented this successfully. This can be verified by running the code in the script.

PART 4 - Classification

For the final part of our coursework, we have attempted to square the circle by implementing a classifying decision tree based on the clustering results obtained above. Although pipelining and numpy array conversion run smoothly, an error notification relating to out-of-bound indices seems to have put a stop to our goals of completing the unsupervised journey we had set ourselves on achieving, from unlabelled, unclassified data to PCA-transformed, clustered data – an error which in our opinion is all due to the lack of classes/labels.

PART 5 - Conclusion

Following on this classification error mentioned above, it is clear from reading this article that:

> “unlabeled data would in fact be used to *improve* classification, not initiate it. This would be done by:
>
> - using Naïve Bayes to learn classes from a small labeled data set
> - extending it to a large unlabeled data set using the EM (expectation–maximization) iterative clustering algorithm.”
>
> (Whitten, 2017)

With more time, we would have most definitiely investigated this further, in order to have unsupervised learning flowing nicely into supervised learning models.

REFERENCE LIST

Goonewardana, H. (2019), “PCA: Application in Machine Learning”, medium.com, Available at: https://medium.com/apprentice-journal/pca-application-in-machine-learning-4827c07a61db (accessed 18 Dec 2018)

Kaggle (2016) Available at: https://www.kaggle.com/dgawlik/house-prices-eda/notebook (accessed 18 Dec 2019)

kassambara (2017) “PCA - Principal Component Analysis Essentials”, sthda.com, Available at: http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical-guide/112-pca-principal-component-analysis-essentials/ (accessed 18 Dec 2019)

Raschka, S. (2014), “About Feature Scaling and Normalization”, Available at: https://sebastianraschka.com/Articles/2014_about_feature_scaling.html (accessed 18 Dec 2019)

Silipo, R. (2015), “Seven Techniques for Data Dimensionality Reduction”, kdnuggest.com, Available at: https://www.kdnuggets.com/2015/05/7-methods-data-dimensionality-reduction.html (accessed 18 Dec 2018)

Whitte, I. (2017), “Beyond supervosed and unsupervised learning”, scientistdirect.com, Available at: https://www.sciencedirect.com/topics/computer-science/unlabeled-data (accessed 18 Dec 2019)