Dong (Bob) Lee

02/13/18

Independent Machine Learning Assignment 2

**Abstract/Motivation**

The paper goes over the idea of unsupervised machine learning algorithms (where this paper will go over the principal component analysis), and techniques of scaling values, where the scaling techniques compliments unsupervised machine learning algorithms by increasing the accuracy. The paper will also go over the compare and contrast between the supervised machine learning algorithm, and the unsupervised machine learning algorithm.

**Introduction**

Sometimes, when the given data becomes too complicated, it’s near obligated that any machine learning model, when the given data is too complicated (meaning, some data points could be way “off” when it comes to other data points, huge amount of dimensions), will drop it’s accuracy, being able to correctly predict the data; therefore, there is a need to scale data to increase the accuracy of the machine learning model. Chaining to the previous concept that was mentioned, unsupervised learning could take a greater advantage of scaling, better than the supervised learning, and the reason is because in supervised learning, the programmer is implementing the algorithms, already knowing how each predictions will look like; therefore, accuracy of the data in unsupervised learning must be in bigger concern than the supervised learning. Even so, since, unsupervised learning machine learning algorithm gives the users the output, where we do not know if the answer is correct; therefore, it should be only used in the expletory setting.

**Method/measurement**

The preprocessing in the machine learning is categorized under an unsupervised transformation of dataset, where a particular dataset become mutated, in order to be suitable for an algorithm, or things a like.

For example, sckit API StandardScaler takes in a set of data, which will ensures that for each feature the mean is 0 and the variance 1 (Introduction to Machine Learning with Python). Another example could be MinMaxScaler, when it comes to the unsupervised transformation, where MinMaxScaler takes in a dataset, and mutates the all the values in between the 0’s and 1’s.

The above procedures can be categorized as a rescaling of the data. The reason why we need to scale data for machine learning algorithm is because the segregations between the data becomes more clear, where one could notice the significant increase in accuracy for many machine learning algorithms, when the algorithm was used with rescaled data.

Let’s assume we have a data set for of cancer, we could import it from the sklearn.dataset, where the code would look like this:

from sklearn.model\_selection import train\_test\_split

from sklearn.datasets import load\_breast\_cancer as cancer

X\_train, X\_test, y\_train, y\_test = train\_test\_split(cancer.data, cancer.target,

random\_state=1)

And when we fit it into support vector machine object, where there will be a demonstration in the next page.

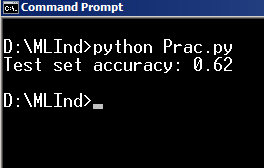
from sklearn.svm import SVC

svm=SVC(C=100)

svm.fit(X\_train, y\_train)

print("Test set accuracy: {:.2f}".format(svm.score(X\_test, y\_test)))

From the above sources,



The machine learning model we currently have maintains the accuracy of 0.62. Compare to the previous data sets, breast cancer data set has 5 feature for each elements, where machine learning model will definitely have a difficult time predicting inputs with greater accuracy. Later on, the book introduces a new concept called principal component analysis, which heavily incorporates the idea of principal component. PCA (principla component analysis) is a compression tool, which enables the user to compress multidimensional data into smaller dimension. For example, referring back to the breast\_cancer dataset, with the code demonstrated in the next page.

from sklearn.datasets import load\_breast\_cancer

from sklearn.model\_selection import train\_test\_split

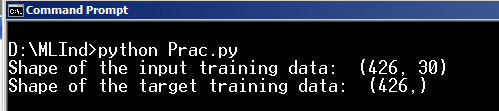
cancer = load\_breast\_cancer()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(cancer.data, cancer.target,

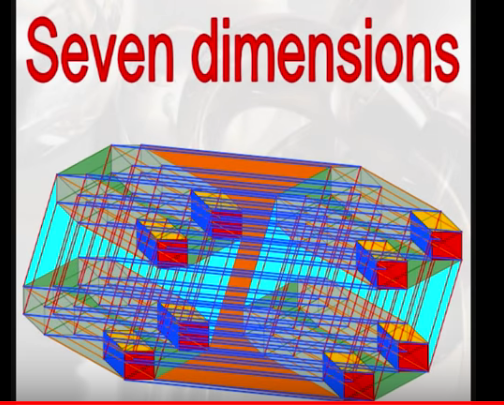
random\_state=1)

print("Shape of the input training data: ", X\_train.shape)

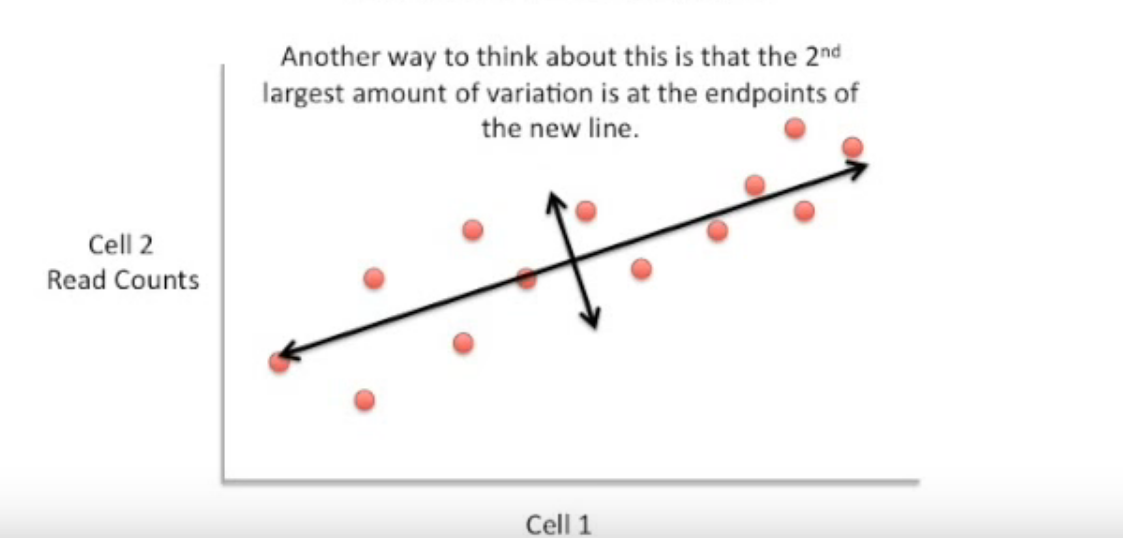
print("Shape of the target training data: ", y\_train.shape)



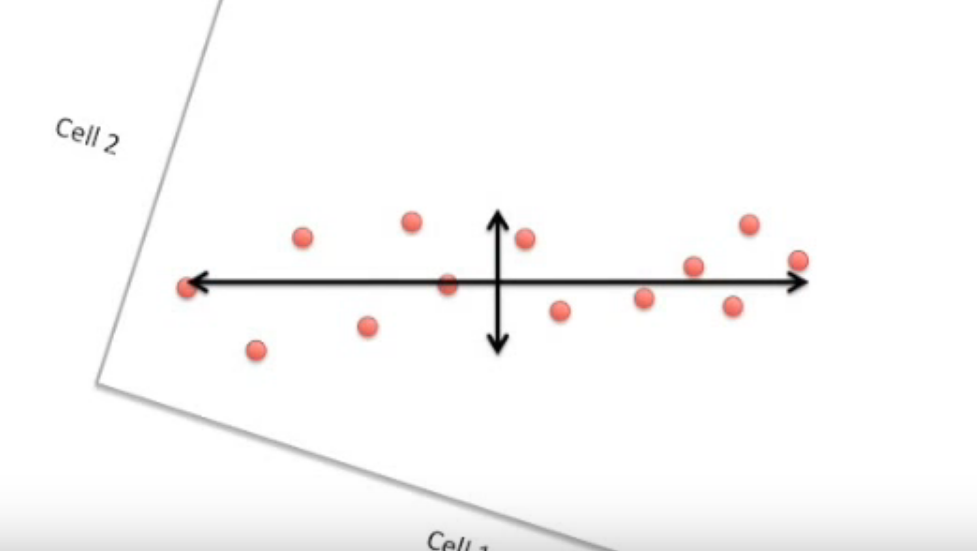
We’ll be able to see this resultant, which implies that breast cancer input training data is logically equivalent to 426 by 30 matrix. In a situation like this, in order for this data to be even visually digestible (because visually representing anything beyond 6th dimension becomes almost impossible to graph), there needs to be a compression of some sort, unless one wants to work their way around to classify data in a situation like below:



Therefore, sometimes, PCA is needed to compress the data in order to for the data to be visually human digestible. The mechanism of PCA is that it, utilizes on principle components, where they are groups of orthogonal vectors, sprouting from the most concentrated area of the data, when looked at it from the graphical perspective. To put it generally, PCA takes dataset with a lot of dimensions, and be able to flattens it to 2 or 3 dimensions so we can look at it. PCA rotates the dataset, such that the Cartesian graph’s origin appears as if it’s in the center of the most concentrated portion of the data; meaning, if the data points are spread out like the picture below:

,

PCA would make it so that the data points will be represented in angled perspective, like:



where the principle components behaves as an x-axis and y-axis, in the case above.

In a representation of code form:

scaler = StandardScaler()

scaler.fit(cancer.data)

X\_scaled = scaler.transform(cancer.data)

pca = PCA(n\_components=2)

pca.fit(X\_scaled)

X\_pca = pca.transform(X\_scaled)

print("X\_pca is: ", X\_pca)

plt.figure(figsize=(8, 8))

the block of code above, under the assumption that we have imported correct libraries to run them, scales the dataset of breast cancer into more algorithm digestable form, in order to get more accuracy. The X\_scaled is assigned with the value being returned by scaler.transform(cancer.data), which is the data that has been scaled with the StandardScaler(). The variable pca is assigned with the PCA object, where the n\_components represents the principle components; since, it’s not 3, the end result of the plot will give us the 2 dimensional graph, where the above program outputs. I have set up my program so that it prints out the contents inside the X\_pca, where the shape of it is:



Upon transpose, the matrix can be represented in 2 dimensional graph.

The code in the next page is a program that displays the plot of compressed version of the breast cancer dataset,

import matplotlib.pyplot as plt

import mglearn

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

from sklearn.datasets import load\_breast\_cancer

cancer = load\_breast\_cancer()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(cancer.data, cancer.target, random\_state=1)

scaler = StandardScaler()

scaler.fit(cancer.data)

X\_scaled = scaler.transform(cancer.data)

pca = PCA(n\_components=2)

pca.fit(X\_scaled)

X\_pca = pca.transform(X\_scaled)

plt.figure(figsize=(8, 8))

plt.legend(cancer.target\_names, loc="best")

mglearn.discrete\_scatter(X\_pca[:, 0], X\_pca[:, 1], cancer.target)

plt.legend(cancer.target\_names, loc="best")

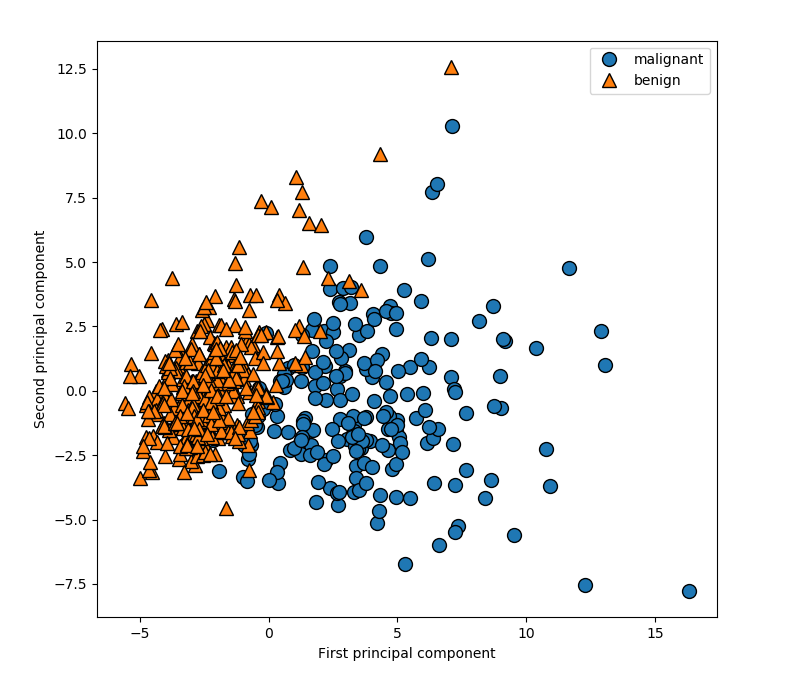
plt.gca().set\_aspect("equal")

plt.xlabel("First principal component")

plt.ylabel("Second principal component")

plt.show()

Aside from the last block of the code above everything else was introduced before, where the last block of the code above finalizes things by making the plot look pretty; however, mglearn is a library provided by the author, where the discrete\_scatter plots the data point, with different colors, according to the target values we pass in (which is the third argument of the function in this case). When the above program runs, we should be able to see something along the line of like the picture in the next page.



Originally, the dataset regarding the breast cancer was 426 by 30 matrix (after the test train split), but with the power of PCA, the graphical representation can be condensed to whooping 2 by n matrix, where n is the number of columns, after the transpose for the array.

**Result**

Implementation of PCA proves to be powerful; however, one needs to recognize that this was entirely done without any supervision, hence, this type of learning is called unsupervised learning. When we were dealing with the dataset of breast cancer, looking at the initial data, there was near to no way of deciphering which data points should be considered malignant, or benign. Looking at the plot, one could still be thrown into a confusion, for certain data points collapse together, despite being categorical different from each other (where, in a case like this, regression is impossible, and so is Kth-Algorithm), where we let the PCA decide if the data point is truly benign, or malignant.

**Bibliography**

**Formal**

Müller, Andreas Christian., and Sarah Guido. Introduction to machine learning with Python: a guide for data scientists. OReilly, 2017.

**Informal**

joshstarmer. “StatQuest: Principal Component Analysis (PCA) clearly explained.” *YouTube*, YouTube, 13 Aug. 2015, www.youtube.com/watch?v=\_UVHneBUBW0.

**Feedback for instructor**

**What was too difficult, too easy?**

Nearly every part of this “run” was ridiculously easy because libraries nicely encapsulates everything about scaling and PCA, and concepts, in an abstract sense, isn’t that hard to understand either.

**What would have made the learning experience better?**

Personnally, I wish the programs I write in the machine learning independent study is more geared towards to writing things in system programming; from now one, I am going to try my best to do just that.

**What did you learn?**

The concrete idea that machine learning is still largely related with data science, where it’s a study of field, which is further away from purity of computer science. The idea of PCA solidifies that, for PCA in itself has an origin from field of statistic.

**How did you learn it?**

Implementing programs regarding unsupervised learning, and comparing my previous experiences with the learning experiences I gain from programming machine learning things.