

Assignment 4

Introduction to Data Science

Magnus Lindberg Christensen (fwz302)

March 18, 2024

In the following assignment I will be working with *Principle Component Analysis* (PCA), *Gradient Descent*, and *Linear Regression*.

Exercise 1

In the following exercise I will implement a function for PCA and utilise for analysis of the *occupancy* and *pesticide* training datasets.

My Implementation

My implementation consists of a PCA function `def PCA()`. The function utilises the concept of **Eigenvalue Decomposition**. The function takes an argument `data`. Before using the function, the target values has been dropped from the data, and the rest of the data is then standardized by performing **z-score scaling**, using the function `StandardScaler` from `scikit-learn`. I then define the covariance matrix of the standardized data by using `np.cov(data.T)`. Here, I use the transpose of the data, to focus on the features (columns) rather than calculating based on each data point (rows). This as well reduces the amount of eigenvectors and eigenvalues. I retrieve the eigenvectors and eigenvalues using `np.eig()`, which takes the covariance matrix and returns the values. Hereafter, based on the list of eigenvalues, I use `np.argsort()` to find the correct indices to sort the list in monotonically decreasing order. Hereafter, I sort both the list of eigenvalues and the list of eigenvectors to follow this order. Since the vectors are always unit vectors, I return these as is. Also, the assignment states that the function should return the variance captured by each principle component, which is exactly what is described by the eigenvalues; the absolute variance captured by each component. Hereby, this is as well returned as is.

Occupancy Plot

In Figure 1 you can see the principle component index and its corresponding explained variance for the standardized occupancy training dataset, plotted against each other. Here it is clear, that the variance captured by each component stabilizes with the increase in components.

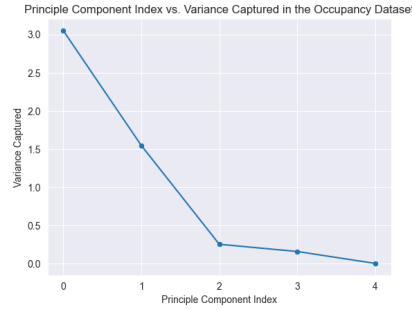
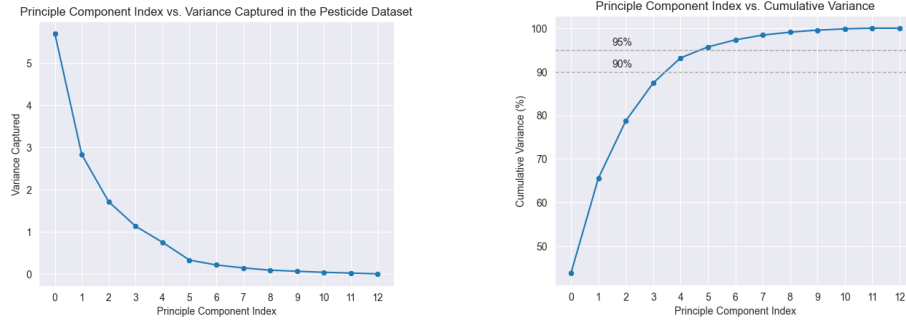


Figure 1: Principle Component Index vs. Variance Captured for the *Occupancy* dataset

Pesticide Plot and Interpretation

In Figure 2 you can see the principle component index vs. the variance captured, and the principle component index vs. the cumulative variance, respectively. Additionally, Figure 2.b has horizontal lines showing the 90% and 95% marks of variance coverage. From this we can conclude that if you were to cover 90% of the data, you would need the first five components, indexes 0 – 4, and hereby five dimensions. If you instead were to cover 95% of the data, you would need the first 6 components, indexes 0 – 5, and hereby six dimensions.



(a) Principle Component Index vs. Variance Captured

(b) Principle Component Index vs. Cumulative Variance

Figure 2: Data visualization for the *Pesticide* dataset

Exercise 2

In the following exercise I will project the data from both of the datasets onto the first two principle components. The two plots can be seen in Figure 3.a and

3.b.

My implementation of the projection consists of a function called `projection` which takes in the standardized data that should be projected onto. The data is then put into the PCA function to retrieve the eigenvectors of the data, hence the principle components. These components are used to define the new matrix, where the data has been projected onto the components. This is done by taking the dot product between the standardized data and the principle components retrieved from the PCA function. Hereby we retrieve a matrix on the form *data points x principle components*. This new basis matrix are then indexed and visualized using a scatterplot, with the x axis being all the data from the first principle component and the y axis are all the data from the second principle component.

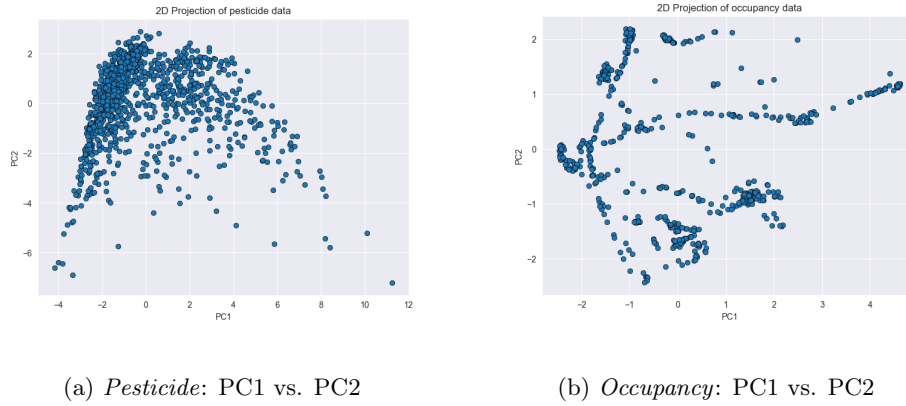


Figure 3: PC1 vs. PC2 visualized for both of the datasets.

Exercise 3

When working with data a common way of preprocessing are by standardization which includes centering and normalization. In the following exercise I will discuss whether each of these preprocessing steps has an effect on the result or not.

Centering: The centering step includes subtracting the mean of each feature from the data, hereby centering the distribution around 0. This centering procedure are not dependent on the features, as this step can be computed regardless of the scaling of features among themselves. Further, centering the data will NOT have an effect on the PCA results. The immediate reason of this is that the PCA computation are based on the relative distance between the data points and captures the variance, and since the spread of the data doesn't change, the PCA components will not change either.

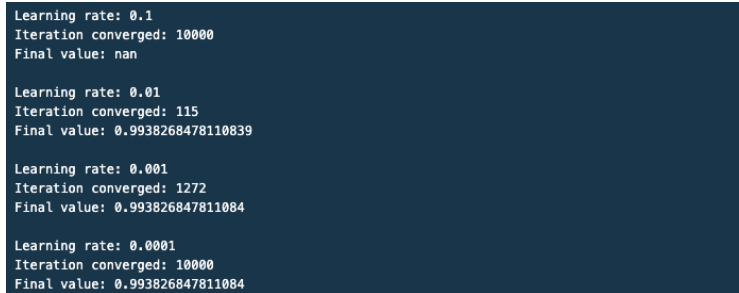
Normalization: The normalization procedure deals with significantly different scaled features by rescaling these often to be between 0 and 1. Hereby, it's

obvious that this procedure is very much dependent on the features and their scales. Further, opposite to centering, normalizing the data will effect the final results, as the relative distance, and hereby variance as well as spread of the data has now changed. Here, high variance features will not anymore have a dominating effect on the PCA results.

Exercise 4

In the following exercise, I will apply gradient descent to find the minimum of the function $f(x) = e^{-x/2} + 10x^2$.

To begin the exercise I found the derivative $f'(x) = 20x - \frac{e^{-x/2}}{2}$. Hereafter I ran gradient descent with the learning rates $learning_rates = \{0.1, 0.01, 0.001, 0.0001\}$ until threshold convergence or maximum iterations had been reached. The final results from this can be seen in Figure 4.



```
Learning rate: 0.1
Iteration converged: 10000
Final value: nan

Learning rate: 0.01
Iteration converged: 115
Final value: 0.9938268478110839

Learning rate: 0.001
Iteration converged: 1272
Final value: 0.993826847811084

Learning rate: 0.0001
Iteration converged: 10000
Final value: 0.993826847811084
```

Figure 4: Learning rates with its corresponding iteration converged and final function value

Three iterations visualization

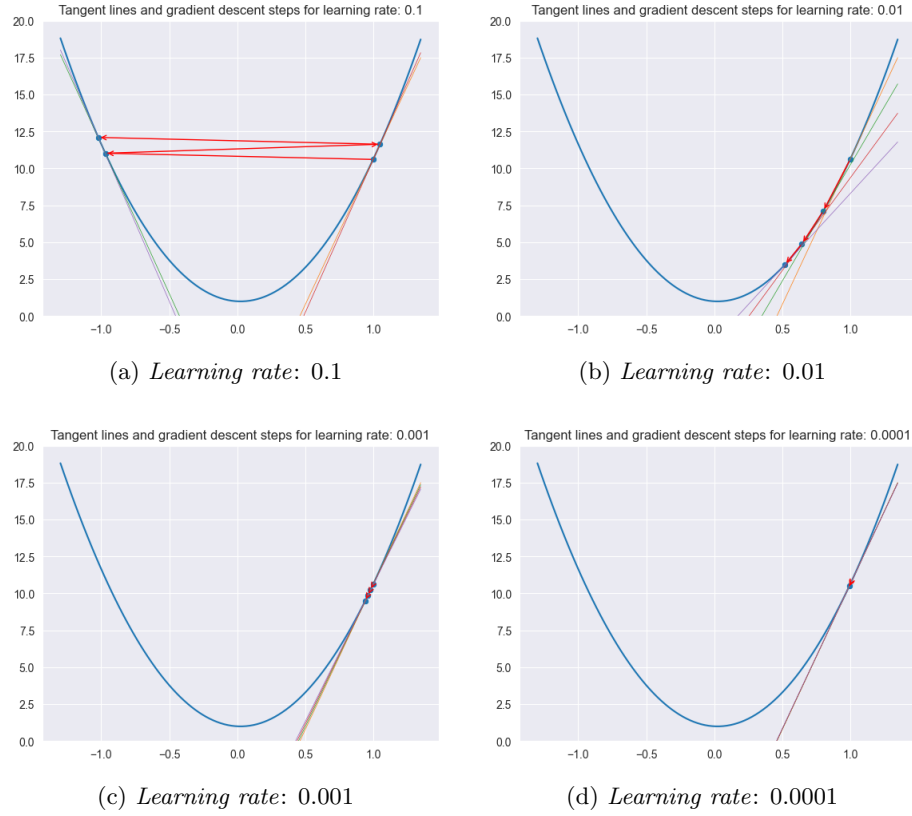


Figure 5: Visualization of the tangent lines and gradient descent steps for the first three iterations of each learning rate

Ten iterations visualization

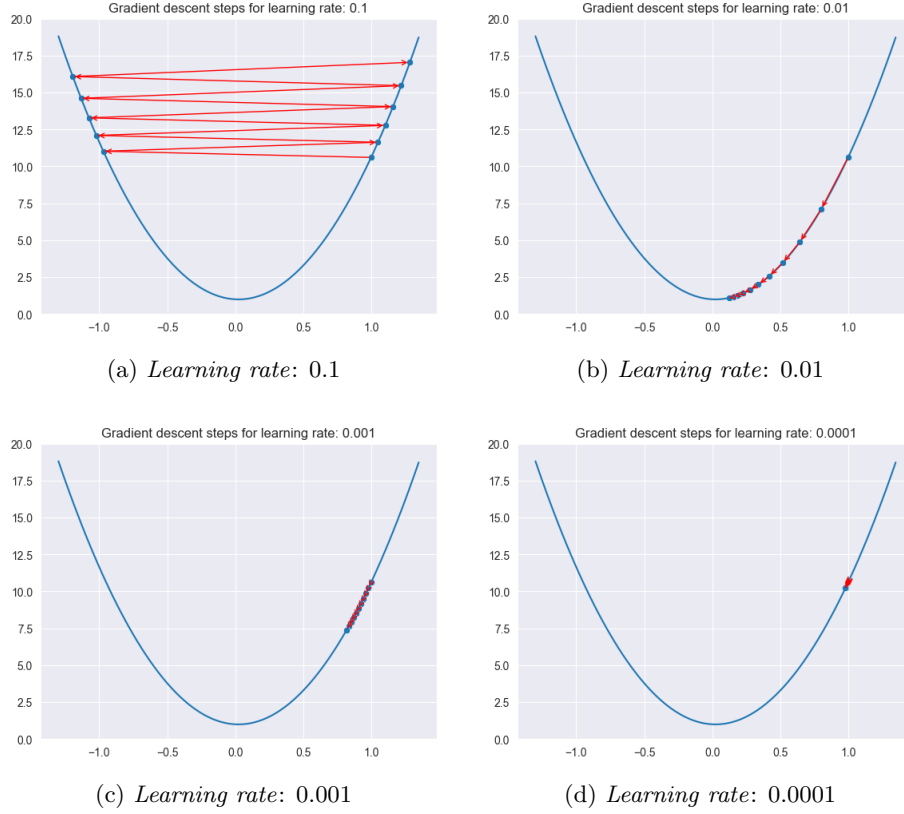


Figure 6: First ten iterations in gradient descent for the four different learning rates

Preferred learning rate discussion

Based on the results in Figure 4, 5 and 6, the most preferable learning rate seems to be 0.01. The reason for this is that a learning rate of 0.1 seems too large for this purpose. For the same reason it reaches a value of *nan*, as each descent diverges from the minimum, and hereby never converges, but rather becomes larger.

On the other end, 0.001 does converge, however with a lot of iterations and not within the maximum allowed iterations, indicating that the learning rate is too small, and takes a lot of time to converge.

Both 0.01 and 0.001 reaches convergence within a reasonable amount of iterations, and with reasonable final values. However, for computational reasons, we will choose 0.01 as the most preferable one.

Exercise 5

In the following exercise I will implement and use linear regression to predict the *temperature* of a building based on three other properties: *relative humidity*, *light*, and *CO2*.

To develop the linear regression I have created a function called *linear_regression* which takes a data input X as a parameter. I have used the Linear Regression model that comes with scikit-learn. In the function the data is being split in $y = t = \text{temperature}$ and X as the rest of the relevant data. Hereafter the model is being fit using these two values like *linear_model = LinearRegression().fit(X = X, y = t)*. Hereafter it's possible to retrieve the offset value w_0 by the method *linear_model.intercept_*. Also, the parameters w_i can be found by *linear_model.coef_*. These are then combined in an array and returned as w which the assignment demanded. The input data had been indexed beforehand to include the columns which was needed.

The parameters w_0 and w_1 from performing linear regression with only a single feature can be seen in Figure 7

```
Retrieved parameters
w0: 21.063442592115397
w1: -0.0016593317912939184
```

Figure 7: Parameters w_0 and w_1 from fitting the linear regression with a single feature

```
Retrieved parameters
wi: [ 2.11268924e+01 -6.36497125e-02  1.89231177e-03  1.42641684e-03]
```

Figure 8: All parameters w_i from fitting the linear regression with three features

In Figure 8 you can see the parameters retrieved from fitting the linear regression with three features. Here, w_0 = temperature, w_1 = relative humidity, w_2 = light, and w_3 = CO2. From this we can retrieve different information. First, the temperature will always be ≈ 21.12 degrees celsius if non of the features are considered ($= 0$). Then we can see that the relative humidity have a slightly negative impact on the temperature, meaning that the temperature will decrease a little bit when the humidity is increasing. This is without taking into account that the data is about occupancy, and that people might open a window when occupying a room. The same, but opposite behavior can be said for the light, w_2 which indicates that the temperature will slightly increase with more light. The same relationship can be found for CO2, w_3 where there also is a slightly positive correlation, meaning that a higher CO2 level will be followed by a slightly higher temperature level.

Exercise 6

In the following exercise I will implement the root mean square error (RMSE), and use this to predict how well the linear regression model performs.

```
RMSE result using one feature; relative_humidity  
1.2067126376680757
```

Figure 9: RMSE result only using relative_humidity

```
RMSE result using three features; relative_humidity, light, CO2  
0.8820484272152366
```

Figure 10: RMSE result only using three features

In Figure 9 and 10 you can see the RMSE results from only using one feature; *relative_humidity*, and using three features; *relative_humidity*, *light*, and *CO2*. From the Figures it is clear that the RMSE result is better when using three features ≈ 0.882 than when only using a single feature ≈ 1.207 . This means that the general deviation from the ground truth temperature labels from the testing data is approximately 0.882 degrees off when using three features, and 1.207 degrees off when using a single feature. This might be reasoned by the fact, that our model has more values to compare a predicted value to when having three features than just a single feature.

Assignment Statements

Gemini version 1.5 has been used to develop reasonable plots corresponding with the assignment specifications. E.g. in exercise 4 visualizing gradient descent steps and tangent lines.