Hyperiondev

TASK

Principal Component Analysis

Visit our website

Introduction

WELCOME TO THE PRINCIPAL COMPONENT ANALYSIS TASK!

So far the problems we have looked at have had only a handful of input variables. In practice, the number of variables in a machine learning task is usually higher. As the number of variables grows, the data becomes harder to work with. Relationships between variables become harder to see, training slows down, and the chance of overfitting increases. It is, therefore, useful to know a bit about how to reduce the number of variables. This task focuses on one way to achieve this: Principal Component Analysis.

Get in touch

Connect for support

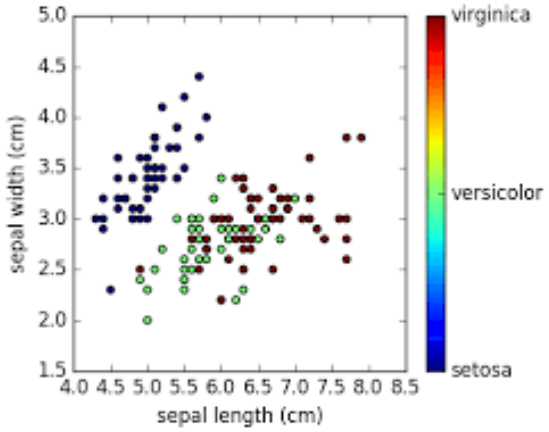
Remember that with our courses, you're not alone! You can contact an expert code reviewer to get support on any aspect of your course.

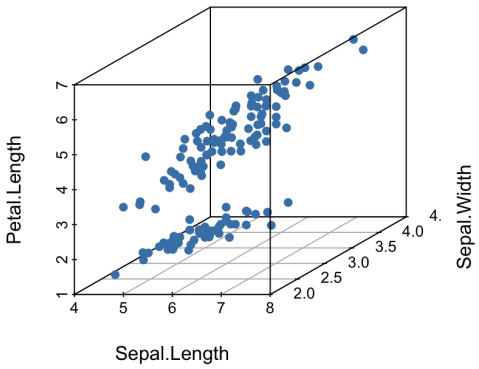
The best way to get help is to login to Discord at **https://discord.com/invite/hyperdev** where our specialist team is ready to support you.

Our team is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

DIMENSIONALITY REDUCTION

If you remember our earlier discussions of distances between points, you'll remember that data points can be represented as a **vector**, an array of points in a feature space, e.g. [4, 3] for the sepal length and width of an instance in the Iris dataset. A subset of the Iris dataset containing only sepal lengths and widths can be visualised in a 2-dimensional space. One that also included the petal length would be 3-dimensional.





In the examples of previous tasks, there were usually only up to three input variables to a problem. Usually, it was easy to see how the input variable might impact the dependent variable. Suppose now that we need to categorise items based on dozens of features, or hundreds, or, as in vision and language problems, thousands. With 10 variables we would need 45 plots to examine the relationship between each pair of variables. Inspecting all of these takes time, and it would be hard to find a pattern that applies to all of the data. This means we need to have methods for dealing with multi-dimensionality in our repertoire.

In many cases, we will find our approach benefits from performing **dimensionality reduction.** Some reasons why dimensionality reduction is useful include:

- Reducing the dimensionality simplifies the dataset, facilitating description and visualisation.
- Reducing the dimensionality of the dataset reduces the size of the distance on which clustering algorithms must calculate the dissimilarity between the observations. This may improve their efficiency and positively affect their performance.
- Reducing the dimensionality of the dataset reduces the risk of overfitting.

Feature selection

One way of dealing with multi-dimensional data is to try to remove features which we know have little impact on the outcome variable of our problem. This may involve manually looking at the input variables and deciding which one of them can reasonably be expected to have any impact on the outcome.

If inspection of the variables' relevance isn't possible or doesn't help you narrow down which features to remove, there are also quantitative measures that can inform you of the strength of an input variable's relationship with an outcome. Scikit-learn has implemented simple methods for applying univariate tests that score input features on their relevance.

INTER-FEATURE CORRELATION

Let's say that you are trying to make a classification in the Iris dataset. You have your four features (petal and sepal widths and lengths), and your output prediction (which species it belongs to). We know that all four features are valuable in making the prediction. However, we also know that, generally speaking, the longer a petal is, the wider it is (and vice-versa). In addition, larger petals generally mean larger sepals, especially if it is just a slightly bigger plant.

What this means is that there is some correlation in the features themselves. To a certain degree, you can predict the size of a petal by looking at the size of a sepal. In other words, you could remove that data and still have a pretty good prediction. This is an example of **redundancy** in the data.

To diagnose redundancy, follow these steps:

1. Remove the target data from the dataframe
2. Use df.corr() to get the correlation coefficients of each feature with each other feature
3. Plot a heatmap of the correlation coefficients

A heatmap is essentially a 2D grid. When looking at cell (0,1) in the grid, the value there shows the correlation between features 0 and 1. To analyse the correlation coefficient heatmap, look at each value. Each value in this matrix lies between -1 and 1. If it is:

- **Close to 1**: This denotes a high positive correlation (i.e. the higher one value gets, the higher the other value gets). This needs to be removed in the data.

- **Close to 0**: There is little to no correlation between features. This is the ideal case.
- **Close to -1**: There is a high negative correlation between features (i.e. the lower one value gets, the higher the other gets). Like the case of it being close to 1, this is bad and needs to be removed.

How do we remove these correlations? One way may be to remove features that are correlated mostly with other features. However, this isn't very practical, as these features still hold some valuable information in them. There is a way to extract this valuable information out of the feature and leave the less valuable stuff behind. This is done by creating **principal components** out of the features using Principal Component Analysis.

PRINCIPAL COMPONENT ANALYSIS

Principal component analysis (PCA) is a dimensionality reduction technique that finds, as the name says, the principal components of a set of input variables. Principal components are directions in the feature space of a dataset that reflect where the data are most spread out. Considering a toy example with vectors [[4, 3, 1], [4, 2, 5], [4.1, 5, 6]], the outcome of PCA would reflect that the toy instances have nearly the same value for X_1 (namely; 4, 4 and 4.1). This means feature X_1 does not help us put the data into different categories. But the dimension of X_2 and X_3 the data points are more spread out.

How it works

PCA works by computing the eigenvectors of the dataset. For any given dataset there are as many eigenvectors as there are features, and each eigenvector has a corresponding eigenvalue. The eigenvectors are the directions in which the data is most spread out, and the eigenvalue describes how spread out the data is (how much variance there is) in that direction. PCA then selects the eigenvectors with the highest eigenvalues and projects the data points into a space that has only those eigenvectors as its dimensions.

Note that principal components (eigenvectors with least variance) of a dataset are not simply a list of the most informative features. Interpreting the results is therefore not simply a matter of interpreting why certain features got removed. Instead, it is often useful to also think about *correlations* between variables. For example, PCA might successfully reduce the dimensions of a dataset that contains the height, weight, and IQ of a group of adults from 3 to 2 dimensions, since height and weight are more strongly correlated to each other than to IQ.

Extra resource

An example of an important application of PCA is described in a paper by Khaled Labib and V. Rao Vemuri. Consider the following:

"With the widespread use of computer networks and telecommunication devices, network security has become a primary concern for the developers and users of these networks. As a result, the problem of intrusion detection has grasped the attention of both research and corporate institutions with the aim of developing and deploying effective Intrusion Detection Systems (IDS) that are capable of protecting critical system components against intruders."

"Network traffic data collected for intrusion analysis is typically high-dimensional making it difficult to both analyze and visualize. Principal Component Analysis is used to reduce the dimensionality of the feature vectors extracted from the data to enable simpler analysis and visualization of the traffic."

To read more about the Application of Principal Component Analysis to the Detection and Visualization of Computer Network Attacks, read the full **paper** by Khaled Labib and V. Rao Vemuri

Instructions

- For a very good explanation of how and why PCA works, please read **this blog post**.

- Read the **PCA.ipynb** notebook for an example of Principal Component Analysis in action. (Note, you may need to `pip install lxml`. If it is not installed, reopen the notebook after installing it).

- See sklearn's **Iris data PCA tutorial** for another example.

Compulsory Task

Follow these steps, using the **PCA.ipynb** for guidance::

- Load the Iris dataset.
- Create a plot of the data using all features to show the positive and negative correlations between them.
- Scale the data, then apply PCA to it. Use `n_components = 3`.
- Plot the transformed version of the data.
- Compare the two plots and comment on what redundancy is likely to have been removed from the data.

If you are having any difficulties, please feel free to contact our specialist team **on Discord** for support.

Completed the task(s)?

Ask an expert to review your work!

Review work

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved, or think we've done a good job?

Click here to share your thoughts anonymously.