Artificial Intelligence and Data Analytics for Engineers

Exercise 5

Introduction

Solving exercises is not mandatory, and exercises will not be graded or corrected by the lecture team. However, we strongly advise you to do as many exercises as possible to prepare for the final exam. Some of the tasks will be discussed with a presentation of the solutions during the exercise session on Thursday. If you want to do exercises at home, we suggest installing Anaconda (https://www.anaconda.com/distribution/ - you will want to download the Python 3.7 version).

In case you have any questions, feel free to send us an e-Mail to: aidae@ima-ifu.rwth-aachen.de.

Task 0) Setting up your Python environment

Since you created a Python environment the first week, we only need to enable the environment from now on. This needs to be done **every time** you open a new terminal, for instance after a system restart or if you accidentally closed the Terminal. Open a terminal and enter:

```
source activate aidae
```

Now you are ready to start programming in Python. Remember to ensure that spyder is installed in your environment (`conda install spyder`) and launched from the terminal where you activated the environment.

Task 1) Feature extraction via Bag of Words (BoW)

In this task you will explore the Bag of Words technique, which is used to extract feature vectors from text data.

a) Scikit learn comes with a built-in bag of words extractor. This can be found in `sklearn.feature_extraction.text` and is called the `CountVectorizer`. Look it up in the scikit learn docs and familiarize yourself with its usage.
Then convert the speech.txt, available in Moodle, into a bag of words model. The text file contains the famous "We choose to go to the moon" speech given by John F. Kennedy in 1962. Your objective is to analyze the contents of the speech using the bag of words model and answer the following questions:
   - Which are the most common words in Kennedy's speech?
   - Which words commonly occur together?

   Also make a judgement on the overall matrix: How many entries does it have? How many of these entries are words that occurred just once? What does that tell you about this technique, with regard to further application in machine learning models?

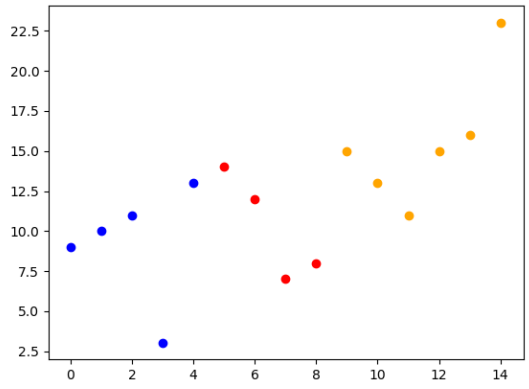
b) Implement your own `bag_of_words(text)` method that takes a string containing multiple sentences as an input and returns a Numpy array of arrays. Each row of the returned array is a BoW feature vector representing a sentence within the original text, while the columns are the occurrence counts of the words. In other words: For each sentence of the input text there should be a row, and for each distinct word within the input there should be a column. Each individual column must represent the same word throughout all rows.

Remember that your BoW needs to handle individual words case insensitive (i.e. the words “Browser” and “browser” count towards the same occurrence count) and that there might be special characters like ‘“’, ‘”’, ‘;’ or, ‘,’ that need to be disregarded.
Hint: One possibility to do this is based on the following procedure: Remove all special characters except for dots. Turn the whole text into lower case. Split into individual sentences based on the dots. Then make a list of all distinct words appearing throughout the data set by iterating over all words in all sentences. Subsequently create a Numpy array of zeros with size (number of sentences, number of words). Finally you count how often the words occur in the individual sentences and fill your array.

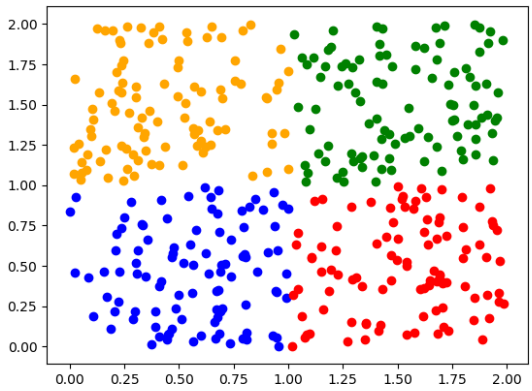
Task 2) Dimensionality reduction

In this task, you will explore Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA). Both of these techniques can be used to reduce the dimensionality of feature vectors, prior to passing them to machine learning algorithms for classification or regression.

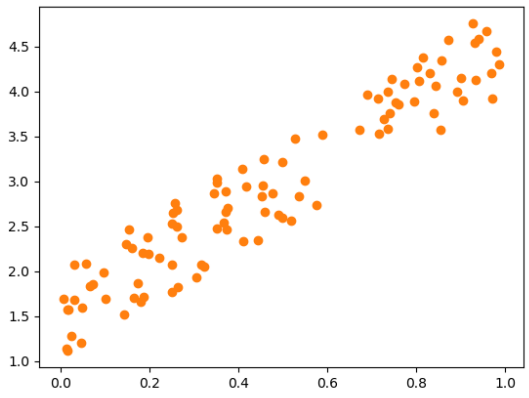
a) Assume you want to reduce the following 2D data sets to 1D. Which of the aforementioned techniques could be applied to which data sets without erasing information obviously required for classification? Justify your answer.



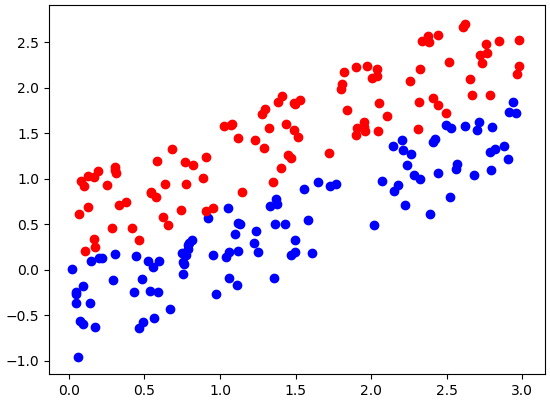
Data set 1



Data set 2



Data set 3



Data set 4

b) Scikit learn comes with several demo data sets. We will use the Iris data set to demonstrate the real-world application of **LDA**. For each row - representing an individual flower – the data set contains four features. These features are the length of parts of the flowers. Additionally, each row is associated with a class label of one of three different types of flowers. We want to project this into a 2-dimensional sub space. Use the LDA implementation of Scikit learn to perform this projection. Then visualize the result in a Matplotlib scatter plot, such that different classes correspond to different point colors.
Hint: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html#sklearn.datasets.load_iris

c) Contrary to our recommendation in the lecture not to implement **PCA** yourself, you will implement your own PCA algorithm in this task (for learning purposes). Create a new Python function `custom_pca(X)`, taking a numpy array of row vectors and returning another numpy array containing the PCA transformed row vectors.
Hint: The Scikit learn library's `sklearn.preprocessing` module contains the `StandardScaler().fit_transform(X)` method, which can be used to center the data to the coordinate system (and normalize its standard deviation). Furthermore, have a look at `numpy.linalg.eig` and `numpy.cov`.