ERHVERVS AKADEMI AARHUS

Machine Learning E2019.

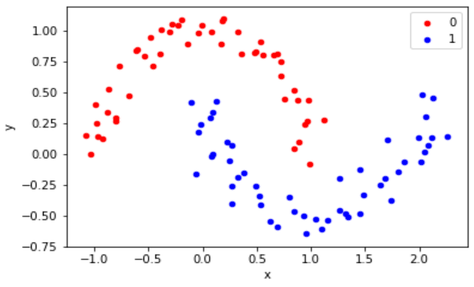
Simon Laub. Dec 9th, 2019.

The Mandatory Exam Assignment. Autumn 2019.

Complete as many of the following exercises as you are able to.
Credit will be given for the efficiency and effectiveness of your solutions and for clear written explanations of the solutions and (important) your own thinking in making the solutions, including the decisions, limitations and interpretations you have made in order to come up with your solution. You must hand-in individuel reports, group hand-ins are not allowed in this exam. Your hand-in can be written in danish or in english. You can do the exercises in any order you want – they are not dependent on each other.
Your report must be handed in on ***Wiseflow on friday the 20th of December 2019 at 16.00 O'clock*** *at the latest.* ***If the report consists of multiple files (report, scripts) then it must be zipped into one file.***

Exercise 1.

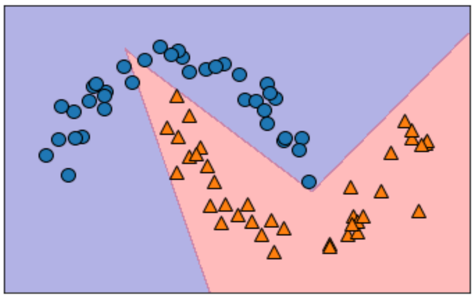
We have previously worked with the (sklearn) dataset "make_moons".



Andreas C. Mueller (author of the book “Introduction to ML with Python”) has made his own python code available on Github that can separate the two moons with the help of a neural network. An adaption of Andreas Muellers code can be found on Canvas (moons_dataset_nn.py). Still, as we have seen earlier in this course - Sklearn’s “make_moons” function can, of course, also generate the moons dataset (with fewer lines of code). And you are free to use the sklearn functionality in the following exercise, if you prefer.

A)

Run the code and adjust the number of hidden layers, the number of units in each hidden layer, alpha (the regularization term, default 0.0001). What do you find? What solution will you recommend?



A plot of a neural net solution that uses two hidden layers.

B)

Compare the result with the neural nets by classification using logistic regression. Is logistic regression possible here? What about k-means? Then compare with (see exercises earlier in the course) decisions trees and random forest techniques. Finally, you might want to try with an SVM (Support Vector Machine) with different kernels (linear, rbf, polynomial, 3 – 5 degrees).

What do you find? If possible, then include plots of some of the classifications to support your findings. Here, as in the other exercises, a .py file or files with your code is expected to be included in the .zip file

Exercise 2

In the class you should have worked with a limited version of real data from Titanic detailing whether passengers survived or died and other data. We want to predict who survives and who dies based on other passenger data.

In this exercise you will look at a larger dataset from the Titanic and use machine learning on this dataset.

The dataset can be downloaded in .csv format from Canvas in the exam assignment folder under modules. It consists of a header and 800 passenger records (And yes...Multiple versions of the Titanic passenger exist, and an alternative version with 887 records, Titanic_alternative.csv, is also uploaded to Canvas. But: **Don't** use this alternative file until you have completed this entire exercise with the 800 passenger excel file. The alternative file is only meant for those who want to dig a little deeper here, after having answered all of the questions in this exercise, or run into problems with the 800 passenger excel file).

The dataset can can be imported into a text program or a .csv reader (I recommend using openoffice for this which is really good at displaying the data from .csv files).

Here is a small explanation of the data (the name and the passengerId should be self-explanatory):

Variable in data	**Definition**	**Key**
Survival	Survived or not	1 = survived, 0 = died
pclass	Passenger class	1 = 1st class, 2 = second class, 3 = third class
Sex	Sex	Male or female
age	Age in years	

sibsp	# of siblings / spouses aboard the Titanic	
parch	# of parents / children aboard the Titanic	
Ticket	A ticket number	
fare	The price paid for the ticket	
cabin	The cabin numer	
embarked	Port city where the passenger embarked	C = Cherbourg, Q = Queenstown, S = Southampton

There are several sections below marked with the capital letters. In each section there are several questions. You job is to answer as many questions correctly as you can.

You should also handin a .py file or files with your code included in the .zip file

A) Getting to know the data and the problem.

What kind of machine learning problem are we talking about in this exercise? Be as precise as possible.

How many different features (not counting labels (y-values) as a feature here) are there in this dataset?

How many of these features would the Y-data (the labels) consist of?

B) Cleaning the data

You have to decide on which of the features for the X training dataset you want to use and then later what to do with missing data for the features you keep.

Think about if there are some features that will most likely have no impact on the survival and discuss why this could be the case. Of course it can be “dangerous” to just remove features, as they might be of even some limited value to a training model, but in this dataset there are features which can be safely removed.

You can back up claims also be looking to see if there is some relationsship or not between the feature and the survival rate. An easy way to see if there is a relationship between two variables is to check for correlation or doing a graph like a scatterplot to see it visually (this was also part of a previous exercise from class). Although, no correlation is not an absolute guarantee that a feature can be removed – remember there could be multi-dimensional relationships between 3 or 4 features that you cannot see.

The feature(s) removed will then not be used in the training and also not in the test set.

Some of the features also have missing data. For instance the age field has missing data. You need to discuss how you will handle missing data? (Think back to earlier discussions we have had in class about similar problems).

You also need to consider that some the data is not numerical data, so that data would need to be converted into numbers (in both training and test set). This is easy to do with pandas – here is an example: `xtrain['Sex'] = xtrain['Sex'].replace(['female'],1.0 )`

This changes all the 'female' labels into 1.0 instead of test for the sex (of course the male would also need a value).

After cleaning (and scaling your data – again see slides or previous exercise on this) then you should split your data into two sets. One for training and then one for evaluating performance.

There are 800 samples in this dataset.

How many would you put into the training set and how many would you put into the test set? Give reasons for your decision.

C) Choosing a model and doing training

We have worked with a few models, such as randomforests, decision trees, neural networks etc.

Choose at least 1 model for doing training and explain your reason for this choice. Train the model using the training data.

D) Evaluating performance on the test set.

In the lecture 5 in the exercises and on the slides (and in the book also) it was discussed how to evaluate performance. Give the precision and recall rates for your model.

Also give the confusion matrix and use the values in the confusion matrix to calcualate the percentage of how many of the samples in the set you got right?

Look at the numbers in the confusion matrix also – what is your model best at and what is it worst at?

E) Experiments

Try to do a few experiments – either with the parameters of your chosen model or by choosing another model and comparing prediction performance with your old model. Document your experiments and results.

Exercise 3.

A)

We have worked with the Iris dataset several times by now. PCA (Principal Component Analysis) can be used to to reduce the dimensionality of the Iris dataset to 2. Which might make the dataset easier to visualize? We have also seen that k-means can be used to find clusters in a dataset.

Start this exercise by revisiting the iris dataset, make a dimensionality reduction with PCA, and use k-means to find clusters in the dataset. Make plots, and explain how many clusters we can reasonably find in the dimensionality reduced Iris dataset.

B)

PCA can be even more useful with larger datasets. Explain why.

Lets take a look at the "Olivetti faces" dataset (This dataset contains a set of face images taken between April 1992 and April 1994 at AT&T Laboratories Cambridge. There are ten different images of each of 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions etc.) See the file, faces_pca_svm.py, on Canvas.

The python program print out (“eigenfaces”) ordered by their importance. Look in the code. Apparently, first something about lighting conditions, and then certain identifying features. Things like nose, eyes, eyebrows? You might want to print out more of the pictures?

Using a SVM, the program then make a classification of the dataset. Finally we run through some pictures from the test set. For a human looking at the data it might **not** be completely trivial to verify the original classification – you might want to look very closely at the original classification in order to be convinced? But, apparently, only 1 picture from the first 25 pictures in the testset is misclassified. Looking at the first 36 (a subplot of size 6x6) gives 3 errors, and so on.

Code like:

```
from sklearn import metrics
y_pred = clf.predict(X_test_pca)
print(metrics.classification_report(y_test, y_pred))
```

might give a more complete report.

Can you improve on this result with other settings on the SVM? What would you try? Is the PCA dimensionality reduction necessary, is it useful? Can you think of other ways to classify that might work (Explain in words, or with code).

Lets say you are only looking for one person in the dataset, how could you change the dataset in order to be able to tell whether a new picture is this person or not? How could you code this?