TP // Machine Learning – 2A
2020-2021

Practical Session

MACHINE LEARNING / DEEP LEARNING

Supervisor: Pierre-Alain MOELLIC

- **Team**: 2 students
- **Validation**: A report (see recommendations below) + Python codes.
- Project must be sent by email to pierre-alain.moellic@emse.fr
  - All files in a ZIP file (no password)
  - **DO NOT ATTACH** the zip file, use seafile for that : https://seafile.emse.fr/

Deadline: 24/01/2021, 23h59

Table des matières

Initial comment

For this Practical Session we will mainly use two Machine Learning (ML) libraries working on PYTHON:

- **SCIKIT-LEARN** (for short: sklearn)
- **TENSORFLOW** with **KERAS**

The documentation of SCIKIT-LEARN (hereafter "sklearn") and TENSORFLOW/KERAS are complete and understandable:

- https://scikit-learn.org/stable/
- https://www.tensorflow.org/tutorials/

Most of all, you usually need 2 or 3 lines of code to apply a ML algorithm. Outside the official documentations, you can find numerous codes using sklearn and Tensorflow.

Moreover, additional resources (JUPYTER NOTEBOOK) are linked to each ML course. These resources are highly relevant with this practical session.

Please, be careful and think before copy & paste.

This Practical Session **is not** a Python exam. For the evaluation of your work, we do not care of your coding abilities (of course, *prettier is better*). The most important is how you reach the objectives, understand what you are doing and play with ML algorithms. You can do more than what is asking (bonus point).

Report recommendations

In your report, you must present and explain what you achieved and your results. Note that, more than the "success", we will be interested in evaluating:

- your understanding of the data and the problem(s)/tasks
- your methodologies
- your critics / reviews according to your results and methods
- your propositions of improvements / perspectives

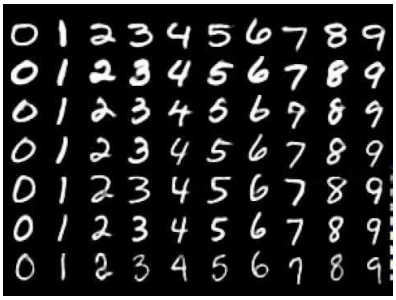
Your report will be evaluated, IF AND ONLY IF the following requirements are met:

- **Report could be in French or English.**
- **PDF and only PDF**
- **Max report size: 25 pages (excluding front page and table of contents).**
- **Use a spell checker (you have been warned… malus points are possible).**
- **Do not write in left-aligned text, use "JUSTIFY" text.**
- **Do not send python files by email, they are usually banned (e.g. by emse).**

1. The datasets: MNIST

Description

The MNIST dataset is the most popular Machine Learning dataset. It represents 70 000 black-and-white 28x28 pixels images with a handwritten digit (‘0’ to ‘9’). The goal is to classify the images among these 10 classes.



Load the dataset

Download MNIST on CAMPUS: MNIST.zip. The archive contains two numpy arrays with the images (MNIST_X_28x28.py, a 70000x28x28 structure) and the associated labels (MNIST_y.npy, a 70000-size array).

Tips

Here is an example for displaying a picture with matplotlib:

```
######## Plot a sample picture
nb_sample = 123
plt.imshow(X[nb_sample])
img_title='Classe ' + str(y[nb_sample])
plt.title(img_title)
plt.show()
plt.clf()
```

A first objective is to check how the dataset is organized.

WORK: What are the shape of the data? Display samples from the dataset.

WORK: Is the dataset well balanced?

WORK: Use the sklearn method `train_test_split` to split the dataset in one train set and one test set:

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

Note that with this method, you can also fix the size of the training set (useful when X is too big…).

2. Unsupervised Machine Learning

2.1. Dimensionality reduction

Dataset

We use the MNIST dataset.

Objectives

Pictures from MNIST have 784 pixels (28x28 grayscale picture), i.e. $x^{(i)} \in X \subset \mathbb{R}^{784}$ we are in a (relatively) high-dimensional space. It is difficult to "see" how the data are distributed and if some intrinsic characteristics between the features exist, that could help further analysis / tasks.

WORK: Perform a Principal Component Analysis (PCA) with sklearn.

https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

WORK: Try different `n_components`.

WORK: Try to display some MNIST pictures with different `n_components`.

An interesting feature is `PCA.explained_variance_ratio_`.

WORK: Explain these values according to your understanding of PCA and use these values to fit `n_components`.

2.2. Data clustering

Dataset

We use the MNIST dataset.

Objectives

Our goal is to cluster X. We will use classical approaches: K-MEANS and EM with Gaussian mixture (see Tips below for documentation links).

WORK: Split X (and y) in a train and test sets with the sklearn method: `split_train_test`.

WORK: With sklearn, perform K-MEANS. Play with the parameter K as well as the initialization (KMEANS++, random, or fixed array).

WORK: For the correct K, evaluate how good is this partition (with the knowledge of y).

WORK: Using the PCA performed in section 2.1. apply K-MEANS with K=10 and `n_components` = 2. Display the partition and comment.

WORK: Briefly explain what is the main difference between K-MEANS and EM with Gaussian Mixture.

WORK: Do the same job with the EM-clustering using the good K parameter (10 for MNIST). Comment your results.

Tips

Most of the Machine Learning algorithms proposed in SKLEARN are called with 3 major steps:

1. Call and initialize the model.
2. Apply (**`fit`**) your algorithm on your (training) data
3. Evaluate the model on test data (**`predict`** or **`score`**)

Sometimes, the first and second steps are processed in a single line. For a better understanding of what you are doing, separate these steps.

Here is the documentation to perform K-Means with sklearn:

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

NB: On the sklearn documentation, you could find several examples of KMeans applied on the IRIS dataset. This Practical Session is not a *Copy & Paste* exercise: please understand everything you type.

A toy example is provided at the end of the documentation, look how it works:

```
>>> from sklearn.cluster import KMeans
>>> import numpy as np
>>> X = np.array([[1, 2], [1, 4], [1, 0],
...               [10, 2], [10, 4], [10, 0]])
>>> kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
>>> kmeans.labels_
array([1, 1, 1, 0, 0, 0], dtype=int32)
>>> kmeans.predict([[0, 0], [12, 3]])
array([1, 0], dtype=int32)
>>> kmeans.cluster_centers_
array([[10.,  2.],
       [ 1.,  2.]])
```

With sklearn, EM algorithm is as simple as K-MEANS to apply:
https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html

- my_em = mixture.GaussianMixture(n_components=3,covariance_type='full')
- my_em.fit(X)
- Y_prediction = my_em.predict(X)

3. Supervised Machine Learning

3.1. Decision Tree, SVM and Logistic Regression

Dataset

We use the MNIST dataset.

With sklearn, and the `train_test_split` method, you already split the database into a train and test datasets.

WORK: Why this train/test separation is compulsory?

Objectives

The main objective is to perform a supervised classification task using AT LEAST TWO classical methods among: (1) Decision Tree, (2) Support Vector Machine (SVM), (3) (multiclass) Logistic Regression, (4) Naïve Bayes Classifier.

NB: "at least" means that you can also do more than 2, according to your time and motivation…

WORK: What is the major difference between Naïve Bayes Classifier and Support Vector Machine (or Logistic Regression)? (if you forgot the course #2, a clue: what are we trying to predict?)

WORK: With sklearn, perform a classification using your favorite methods. With the documentation, check how to modify the parameters and comment how it influences the results.

For example, if you chose SVM, change the kernel between 'linear' and 'rbf' (Gaussian kernel). You can also play with 'C' parameter to switch from hard-margin SVM to soft-margin SVM[1]…

WORK: With the `score` method, compute the accuracy of the model on the training and the test datasets. Why do we need to analyze the performance of the model at training and testing time?

WORK: In section 2.1. you applied a PCA to X so that the projected set – hereafter X^{red} – lies on a "reduced" space. Among the supervised methods you chose, select one method and apply your code to X^{red}. Does the PCA influence the performance of the classification (according to the intensity of the reduction)?

Tips

Here is the documentation to perform SVM, Decision Tree, Logistic Regression and Gaussian Naïve Bayes classifier with sklearn:

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

A completely dumb example:

```
>>> from sklearn import svm
>>> X = [[0, 0], [1, 1]]
>>> y = [0, 1]
>>> clf = svm.SVC()
>>> clf.fit(X, y)
SVC()
```

[1] https://scikit-learn.org/stable/auto_examples/svm/plot_svm_margin.html

```
>>> clf.predict([[2., 2.]])
array([1])
```

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

3.2. Deep Learning

3.2.1. Dataset

The dataset is MNIST.

Note that – if necessary – we can reduce the complexity by building a "short version" with only the three or four first classes ('0', '1', '2', '3'). For example:

- X_train_short=X_train[y_train<4]
- y_train_short=y_train[y_train<4]

3.2.2. Objective and tool

Same as in 3.1.: we want to perform a supervised image classification task.

For a model based on neural networks, the development platform will be TENSORFLOW/KERAS.

The documentation and tutorials are here: https://www.tensorflow.org/tutorials/. Advice: use the "Functionnal API" rather than the "Sequential API".

Tensorflow with Keras enables simple model design in a pure "Lego style": you just need to define each layer and connect one to the following (remember "feedforward network"?):

```
input_layer = Input(shape=(nb_features,))
x = Dense(123)(inputs)
x = ReLU()(x)
x = Dropout(0.25)
x = Dense(54)(x)
x = ReLU()(x)
x = Dense(num_classes)(x)
output_layer = Softmax()(x)
model=Model(input_layer, output_layer)
```

Then, you need to `compile` with some important neural networks characteristics, `fit` with your data (here you can define the validation set) and `evaluate` on X^{test}.

3.2.3. MultiLayer Perceptron (MLP)

Your first deep neural network is a Multy Layer Perceptron (MLP), i.e. a feedforward network only composed of fully connected layers. For your first attempt, use only why hidden layer.

WORK: What is the size of the input tensor? What is the size of the output layer?

WORK: How many epochs do you use? What does it mean? What is the batch_size? What does it means? Why do we define a validation set (for example: `validation_split=0.2`)? What kind of parameters do we set with the `compile` method? And with the `fit` method? For each one, explain why it is an important parameter and what is the influence on the training process?

WORK: Comment the training and the results.

WORK: Is there any overfitting? Why? If yes, what could be the causes? How to fix this issue?

WORK: If you do not observe overfitting, how can you make your model overfit? Try and demonstrate the overfitting.

WORK: According to this first performance, change the architecture of the MLP (change parameters, add/remove layers…) as well as hyper-parameters, explain why, what are the influence on the results…?

3.2.4. Convolutional Neural Network (CNN)

WORK: Why this type of neural networks is (probably) the more relevant for our task? Is the input tensor size the same as for MLP? Why?

WORK: Build a first CNN model with only one convolutional block. Comment the training and the results.

WORK: Change the architecture of the CNN (change parameters, add layers…) as well as hyper-parameters, explain the influence on the results.