Analyse et Traitement de l'Information

TP4: Clustering approaches and autoregression.

The MNIST dataset of handwritten digits [1] consists of 28×28 grayscale images. To download the dataset use this code:

```
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split

images, labels = fetch_openml('mnist_784', return_X_y=True, as_frame=False)
train_images, test_images, train_labels, test_labels = \
            train_test_split(images, labels, random_state=42)
```

To filter images of the specific class you can use the following method:

```
import numpy as np

def select_with_label(images, labels, desired_labels):
    mask = np.isin(labels, desired_labels)
    return images[mask], labels[mask]

images_of_two, labels_of_two = \
        select_with_label(train_images, train_labels, desired_labels=['2'])
images_of_odd, labels_of_odd = \
        select_with_label(train_images, train_labels, \
        desired_labels=['1', '3', '5', '7', '9'])
```

You can copy-paste from this link.

[1] http://yann.lecun.com/exdb/mnist/

1 Clustering

Classification is a *supervised* problem, where labeled (training) data is used to label the unlabeled (testing) data.

Clustering is an *unsupervised* problem, where the objective is to group a given (unlabeled) dataset $\{\mathbf{x}\}_{i=1}^{n}$ into k clusters, so that similar samples appear in the same cluster, and dissimilar samples are in different clusters.

K-Means

The k-means algorithm is used to partition a given set of observations into k clusters where k is chosen in advance. The algorithm starts with a random set of k center-points (μ). During each update step, all observations x are assigned to their nearest center-point (see equation 1). In the standard algorithm, only one assignment to one center is possible. If multiple centers have the same distance to the observation, a random one would be chosen.

$$S_i^{(t)} = \left\{ x_p : \left\| x_p - \mu_i^{(t)} \right\|^2 \leq \left\| x_p - \mu_j^{(t)} \right\|^2 \ \forall j, 1 \leq j \leq k \right\} \tag{1}$$

Afterwards, the center-points are repositioned by calculating the mean of the assigned observations to the respective center-points (see 1).

$$\mu_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j \tag{2}$$

The update process reoccurs until all observations remain at the assigned center-points and therefore the center-points would not be updated anymore.

This means that the k-means algorithm tries to optimize the objective function 3. As there is only a finite number of possible assignments for the amount of centroids and observations available and each iteration has to result in a better solution, the algorithm always ends in a local minimum.

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2 \tag{3}$$

$$\text{with } r_{nk} = \begin{cases} 1 & x_n \in S_k \\ 0 & \text{otherwise} \end{cases}$$

The main problem of k-means is its dependency on the initially chosen centroids. The centroids could end up in splitting common data points whilst other, separated

points get grouped together if some of the centroids are more attracted by outliers. These points will get pulled to the same group of data points.

The most common approach is to perform multiple clusterings with different start positions. Afterwards, the clustering, which occurred most is considered as correct.

```python
class KMeans:
    def __init__(self, n_clusters, init="random", max_iter=300):
        raise NotImplementedError()

    def fit(self, X, y=None):
        # initialize centroids randomly
        # this function should update self.centroids coordinates
        # for given X
        raise NotImplementedError()

    def predict(self, X):
        # this function for each X returns the cluster
        # it belongs to
        raise NotImplementedError()
```

Questions 1

1.1 Create a subset of MNIST which consists of 2000 randomly sampled instances of "3"s and "7"s.

1.2 Implement Kmeans from scratch (do not use sklearn) with random initialization of the clusters. Perform the algorithm with k = 2, 3, . . . , 10. Compute J defined in eq.3 for each k. Plot J varying with k. Describe the effect of k over J.

Gaussian Mixture Models

Gaussian Mixtures are based on K independent Gaussian distributions that are used to model K separate clusters.

The probability for a point $\mathbf{x}$ to belong to a Gaussian with $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ is

$$N(\mathbf{x} : \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1.}{\sqrt{(2\pi)|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right)$$

In the GMM case, each Gaussian (cluster) will have a different weight π_k, where $1 \leq k \leq K$ and $\sum_{k=1}^{K} \pi_k = 1.0$. For each point, we will estimate the probability to belong to a certain cluster, but we do not assign the cluster directly (can be done using argmax over probabilities or over distances to the centers).

Probability for a point x_i to belong to a Gaussian Mixture Model is

$$P(\mathbf{x}_i) = \sum_{k=1}^{K} \pi_k N(\mathbf{x_i} : \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

The probability of the whole point set to belong to the clusters modeled as GMM is

$$P(\mathbf{X}) = \sum_{i=1}^{N} P(\mathbf{x}_i)$$

Before talking about EM algorithm, let us understand what is the likelihood function. The likelihood function is just a probability function of the parameter values (these parameters are usually written as θ). if we knew the true parameters θ, this would be a probability density (so the formula is the same, but what we call variables is different)

In other words,

$$\mathcal{L}(\theta \mid x) = f_\theta(x).$$

For the GMM, $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ (means and covariance matrixes) and the likelihood is

$$P(\mathbf{X}) = \sum_{i=1}^{N} \sum_{k=1}^{K} \pi_k N(\mathbf{x_i} : \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = f_{\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}}(\mathbf{X})$$

Expectation-Maximization Algorithm

Expectation-Maximization algorithm is an iterative algorithm. The EM algorithm seeks to find the MLE of the marginal likelihood by iteratively applying these two steps:

Expectation step (E step):

Define $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)})$ as the expected value of the log-likelihood function of $\boldsymbol{\theta}$, with respect to the current conditional distribution of unobserved latent data $\mathbf{Z}$ given observed data $\mathbf{X}$ and the current estimates of the parameters $\boldsymbol{\theta}^{(t)}$:

$$Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}) = \mathrm{E}_{\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{(t)}} \left[\log L(\boldsymbol{\theta}; \mathbf{X}, \mathbf{Z})\right]$$

Maximization step (M step): Find the parameters that maximize this quantity:

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)})$$

For the GMM, as expectation step, we will compute $P(x)$. As maximization step we will update θ values:

$$\hat{\boldsymbol{\pi}} = \frac{1}{N} \frac{\sum_{i=1}^{N} \boldsymbol{\pi} N(x_i : \boldsymbol{\mu}, \boldsymbol{\Sigma})}{P_\theta(\mathbf{x}_i)} \tag{4}$$

Same for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$.

```python
class GMM:
    def __init__(self, n_components, max_iter = 100):
        self.n_componets = n_components
        self.max_iter = max_iter
        # pi list contains the fraction of the dataset for every cluster
        self.pi = [1/self.n_componets for comp in range(self.n_componets)]


    def fit_predict(self, X, y=None):
        #this function returns a cluster number to each element of X
        raise NotImplementedError()
```

Questions 2

2.1 Use a subset of MNIST consisting of 2000 randomly sampled instances of "3"s and "7"s from the previous task. Implement GMM model from scratch (do not use sklearn) with random initialization of the parameters. Perform the algorithm with k = 2, 3, . . . , 10.

2.2 Build confusion matrices using sklearn for K-Means and GMM.

2.3 Describe the differences between K-Means and GMM (deterministic/probabilistic, parameters, functions to optimize, updates).

2.4 Describe how K-Means can be seen from the perspective of the algorithm EM (take distance instead of probability function).

Hierarchical Clustering

We will take a look at another useful type of Clustering – Agglomerative Clustering. The idea is to unite the clusters with the shortest distance between them. This method has a high computational complexity so it is applied only **for small datasets**.

Linkage is a method which will be used to compute the distance between the clusters.

- *average* - the distance between two clusters is the average distance between all possible pairs of points where one belongs to one cluster and another one to another
- *single* - the distance between two clusters is the minimal distance between two points where one belongs to one cluster and another one to another
- *complete* - the distance between two clusters is the maximal distance between two points where one belongs to one cluster and another one to another

```
class AgglomerativeClustering:
    def __init__(self, n_clusters=16, linkage="complete"):
        raise NotImplementedError()

    def fit_predict(self, X, y=None):
        #this function returns a cluster number to each element of X
        raise NotImplementedError()
```

Question 3

3.1 Implement AgglomerativeClustering with complete distance using **sklearn**. Explain the algorithm in the report.

3.2 Take 200 instances of “3”s and “7”s from the MNIST dataset. For the case k = 2, give the confusion matrix for both Agglomerative and KMeans clustering. (you can use sklearn implementation of KMeans for this task)

3.3 Sample 200 instances of “3”s and “5”s. Perform both clustering approaches with k = 2. Build the confusion matrix. Based on your results, explain the difference between these two clustering problems: (1)“3” vs. “7”; (2) “3” vs. “5”.

2 Time series prediction

You will find two csv data files in the data folder:

- `data_mintemp.csv` contains daily minimum temperature data with two columns: ”Date” and ”Temp”.
- `data_cons.csv` contains daily electrical consumption, wind and solar electricity production with three columns: `"Date"`, `"Consumption"`, `"Wind+Solar"`.

Questions 4

4.1 Import the data contained in these files and create three datasets with the for- mat $[(t, x_t)]$ where t represents all possible times for the dataset (”Date”), and x_t could be either the corresponding `"Temp"`, `"Consumption"`, or `"Wind+Solar"`. Hint: you could use `pandas.read_csv` method with the right parameters for `path`, `header`, `index_col`, and `usecols`.

4.2 For each of these datasets, perform the following tasks (you could create a function for these tasks and apply it on each dataset):

(a) For $k \in \{7, 30, 365\}$, create a new feature of the dataset named `"SMA_k"` which corresponds to the simple moving average with a window of size k defined as the simple mean of the k previous values:

$$\text{SMA-k}_t = \frac{\sum_{i=0}^{k-1} x_{t-i}}{k}$$

The dataset will now have 5 columns: `"Date"`, `"value"`, `"SMA_7"`, `"SMA_30"`, and `"SMA_365"`.

(b) Plot the data along with their three simple moving averages and give an interpretation.

(c) Create a new feature of the dataset named `"value-365"` (`"value"` being the correct considered data) which corresponds to the serie $[y_t] = [x_t - \text{SMA-365}_t]_t$

(d) Create a new feature of the dataset named `"SMA_30(value-365)"`, which corresponds to the simple moving average of the previous series `"value-365"`, with a window of size 30, this new serie is $[z_t]_t$.

(e) Create a new feature of the dataset named `"value-365-30"` (`"value"` being the correct considered data) which corresponds to the series $[y_t - z_t]_t$. The dataset will now have 8 columns.

(f) Plot in the same figure the series `"value"`, `"SMA_365"`, `"SMA_30(value-365)"`, and `"value-365-30"`. Explain the mathematical link between these series and give a clear interpretation to them (you should use the same terminology as in slides ATI.06).

4.3 For data, with only the original two columns, perform the following tasks: the first dataset only, which contains the daily minimum temperature

(a) For k $\in \{365, 182, 91\}$, create a new feature column which is the serie of temperature lagged by k days. You could use the pandas function shift, and get the autocorrelation $\text{Cor}(x_t, x_{t-k})$ between the original and lagged series. You can use the `pandas.plotting.autocorrelation_plot` function.

(b) On the same figure, plot the three previous autocorrelation points and the global autocorrelation curve (for any possible k): the x-axis corresponds to k and the y-axis to the correlation.

(c) Give an interpretation to this figure.

4.4 Define a function which takes as arguments (y_{t-1}, a, σ), and returns the predicted value y_t (scalar) defined in slide 17 of ATI.6 where:

- y_{t-1} is a vector of length d: the d previous values of the serie,
- a is a vector of length d: the Yule-Walker coefficients of deepness d,
- σ is a scalar: the constant in the autoregressive formula.

4.5 Define a function which takes as arguments (`time_series`, `deep`, `n_pred`, `n_train`) and returns the predicted time series (of length `n_pred`), where:

- `time_series` is a time series (i.e. a pandas dataframe or series) of length more than `n_train` + `n_pred`,

- `deep` is the deepness value for autoregression formula and for the Yule-Walker coefficients (it must be smaller than `n_train`),
- `n_pred` is the number of values of the series to predict: with indexes `n_train` $\rightarrow$ `n_train` + `n_pred` - 1,
- `n_train` is the number of the first values of the series to use for the training: with indexes 0 $\rightarrow$ `n_train` - 1.

You can use the previous function and the function

`statsmodels.api.regression.yule_walker`.

4.6 6. For the first dataset only, which contains the daily minimum temperature data, for each `deep` $\in \{7, 30, 365\}$ predict the last year of data

(`length(time series)` - 365 and `n_pred = 365`) and find the Mean Square Error (MSE) with the real values of the last year of the series. Comment on you results.

Submission

Please archive your report and codes in "Prénom Nom.zip" (replace "Prénom" and "Nom" with your real name), and upload to "`Upload TP4 - Clustering approaches and autoregression.`" on `https://moodle.unige.ch` before Monday, November 14 2022, 23:59 PM. Note, the assessment is mainly based on your report, which should include your answers to all questions and the experimental results. *Importance is given on the mathematical explanations of your works and your codes should be commented*

Please use the template from the first TP.