

### 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 ( $\mu$ ). 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.

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

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

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

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 better solution, the algorithm always ends in a local minimum.

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

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

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. This 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.

---

```

1 class KMeans:
2     def __init__(self, n_clusters, init="random", max_iter=300):
3         raise NotImplementedError()
4
5     def fit(self, X, y=None):
6         # initialize centroids randomly
7         # this function should update self.centroids coordinates
8         # for given X
9         raise NotImplementedError()
10
11    def predict(self, X):
12        # this function for each X returns the cluster
13        # it belongs to
14        raise NotImplementedError()

```

---

## Hierarchical Clustering

We will look at the specific type of Hierarchical 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

---

```

1 class AgglomerativeClustering:
2     def __init__(self, n_clusters=16, linkage="average"):
3         raise NotImplementedError()
4
5     def fit_predict(self, X, y=None):
6         #this function returns a cluster number to each element of X
7         raise NotImplementedError()

```

---

## Questions

1. Create a subset of MNIST which consists of 2000 randomly sampled instances of “3”s and “7”s.
2. Implement Kmeans with random initialization of the clusters. Perform the algorithm with  $k = 2, 3, \dots, 10$ . Compute  $J$  defined in eq.3 for each  $k$ . Plot  $J$  varying with  $k$ . Describe the effect of  $k$  over  $J$ .
3. Implement AgglomerativeClustering with average distance.
4. For the case  $k = 2$  in the above experiments, give the confusion matrix for both Agglomerative and KMeans clustering.
5. Sample 2000 instances of “3”s and “5”s (rather than “3”s and “7”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

1. Import the data contained in these files and create three datasets with the format  $[(t, x_t)]$  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`.
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).

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.
  - (b) On the same figure, plot the tree 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. You could use the `pandas.plotting.autocorrelation_plot` function.
  - (c) Give an interpretation to this figure.
4. Define a function which takes as arguments  $(y_{t-1}, a, \sigma)$ , 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  $d$ ,
  - $\sigma$  is a scalar: the constant in the autoregressive formula.
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`.

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 15 2021, 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*