Machine Learning 1, SS21
Homework 4
Maximum Likelihood Estimation. K-means. Expectation-Maximization Algorithm

Horst Petschenig
Ceca Kraisnikovic

Tutor:	Sofiane Correa de Sa, correadesa@student.tugraz.at
Points to achieve:	25 pts
Bonus points:	2* pts
Info hour:	will be announced via TeachCenter
Deadline:	25.06.2021 23:55
Hand-in procedure:	Use the **cover sheet** that you can find in Teach Center. Submit your **python files and a report** in Teach Center. Do not zip them. Do not upload the data folder.
Course info:	TeachCenter

Contents

General remarks

Your submission will be graded based on:

- Correctness (Is your code doing what it should be doing? Is your derivation correct?)
- The depth of your interpretations (Usually, only a couple of lines are needed.)
- The quality of your plots (Is everything clearly visible in the print-out? Are axes labeled? . . .)
- Your submission should run with Python 3.5+.

1 Maximum Likelihood Estimation [5 points]

We say that $X \in \{0, 1, ..., \}$ has a Poisson distribution with rate parameter $\lambda > 0$, written $X \sim \text{Poi}(\lambda)$, if its probability mass function is

$$\text{Poi}(\lambda) = \frac{\lambda^x}{x!} e^{-\lambda}$$

The term $e^{-\lambda}$ is just a normalization constant required to ensure the distribution sums to 1. The Poisson distribution is often used as a model for counts of rare events like radioactive decay and accidents.

The gamma distribution is a flexible distribution for positive real valued random variables, $x > 0$. It is defined in terms of two parameters, called the shape α and the rate β

$$\text{Gam}(\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x},$$

where $\Gamma(x)$ is the gamma function

$$\Gamma(x) = \int_0^\infty u^{x-1} e^{-u} du.$$

Tasks:

1. Let us assume a dataset with i.i.d. observations $X = [x_1, \ldots, x_N]$ and a Poisson distribution. Derive the maximum likelihood estimate for the rate parameter λ using

   $$p(X|\lambda) = \prod_{i=1}^{N} \frac{\lambda^{x_i}}{x_i!} e^{-\lambda}$$

   by optimizing the log-likelihood (compute the derivative, set to zero, solve for λ).

2. In order to take a more Bayesian approach, we will assume a gamma prior $p(\lambda) = \text{Gam}(\lambda|\alpha, \beta) \propto \lambda^{\alpha-1} e^{-\lambda\beta}$. The gamma distribution is the conjugate prior of the Poisson distribution; the result is a gamma posterior distribution. Derive the maximum posterior estimate for the rate parameter λ using $p(\lambda|X) \propto p(X|\lambda)p(\lambda)$ by optimizing the log-posterior. *Hint:* Rewrite $p(X|\lambda)p(\lambda)$ in form of a Gamma distribution $ae^b\lambda^c$ where you will have to figure out a, b and c; then compute the log-posterior.

2 K-means. Expectation-Maximization Algorithm [20 points]

In this task, we will implement K-means and Expectation-Maximization algorithms, and evaluate them using the Mouse data set. Functions for loading the data set, plotting the original data, and plotting the clusters are already provided.

2.1 K-means Algorithm [7 points]

Suppose we have a data set $x_1, ..., x_N$, consisting of N observations of a D-dimensional Euclidean variable x. We aim at partitioning the data set into K different clusters. The number of clusters, K, is either given or chosen by us. Each cluster has a center (centroid) μ_k, $k = 1, .., K$, $\mu_k \in \mathbb{R}^D$, i.e., the centroids are D-dimensional vectors, and there are K such vectors, one for each cluster.

Cost function

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

represents the sum of the squares of the distances of each data point to its assigned vector μ_k, $r_{nk} \in \{0, 1\}, k = 1, \ldots, K$ and $n = 1, \ldots, N$, represents a binary indicator variable describing which of the K clusters the data point x_n is assigned to.

Formally,

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j ||x_n - \mu_j||^2 \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

This means, to each data point a one-hot vector of length K is assigned – there is exactly one 1 in the vector at the index of the assigned cluster, the remaining entries of the vector are zeros.

Centroids (means of clusters) are calculated as follows:

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}} \tag{3}$$

Tasks:

1. In the code implement function **euclidean_distance()** that calculates the Euclidean distance between two vectors x and y as: $d = ||x - y||_2 = \sqrt{\sum_i (x_i - y_i)^2}$.
2. In the code, in the function **cost()** implement the equation J from Equ. 1. Note that it uses the squared Euclidean distance.
3. In the code, implement function **closest_centroid()**. This function takes a single sample (data point), calculates distances to all centroids, and returns an index – the index to the closest centroid. Note that this expression uses the squared Euclidean distance. This implementation would correspond to the *arg min* part of Equ. 2.
4. In the code, in function **assign_samples_to_clusters()** implement Equ. 2. Note that in this function you should use the function **closest_centroid()** that returned the index of the closest centroid. In principle, this function assigns one-hot vectors to each sample.
5. In the code, in function **recompute_centroids()** implement Equ. 3.
6. In the code, in function **kmeans()** add function calls for assigning samples to clusters, then evaluate the loss function, recompute the centroids, and calculate the cost function again. (We have to evaluate the cost function two times, so implement all 4 TODOs provided in the code, in the order specified there.)
7. In **main.py/task_kmeans()**, choose the appropriate number of clusters K and maximum iterations *max_iter*. There is already a function call to **kmeans()** provided.
8. In **main.py** implement function **plot_cost()** that plots the cost function over iteration. Include the plot in the report.
9. There is already a function call to plot the Mickey Mouse after clustering. Include the plot in the report. Compare it with the plot of the original data (in one or two sentences).
10. Set $K = 6$. Include the plot of Mickey Mouse in the report. Compare it with the plot of the original data (in one or two sentences).

2.2 Expectation-Maximization (EM) Algorithm [10 points]

A Gaussian Mixture Model (GMM) with K components is given as:

$$p(x_n|\Theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k), \tag{4}$$

where Θ contains the parameters π_k that represent the component weights (constraint $\sum_{k=1}^{K} \pi_k = 1$ must be satisfied), μ_k the means, and Σ_k the covariance matrices.

Steps of the EM algorithm are:

- Initialize the parameters π, μ, Σ.
- **Expectation step (E-step):** For each sample x_n, calculate the probability $p(k|x_n, \Theta)$ that the sample was caused by the k-th component of the GMM:

$$\gamma_{nk} = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} \tag{5}$$

  This corresponds to assigning responsibilities to each sample (soft classification).

  (For easier understanding: The numerator is calculated for cluster k, whereas the denominator calculates the same but for all clusters, and then these parts, for each cluster, are summed.)

- **Maximization step (M-step):** Calculate the effective number of samples for the k-th component:

$$N_k = \sum_{n=1}^{N} \gamma_{nk}, \tag{6}$$

  and update all the components of Θ, namely, the parameters :

$$\mu_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} x_n, \tag{7}$$

$$\Sigma_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} (x_n - \mu_k^{\text{new}})(x_n - \mu_k^{\text{new}})^T, \tag{8}$$

$$\pi_k^{\text{new}} = \frac{N_k}{N} \tag{9}$$

- Evaluate the log likelihood cost function

$$\log p(X|\Theta^{\text{new}}) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k^{\text{new}} \mathcal{N}(x_n|\mu_k^{\text{new}}, \Sigma_k^{\text{new}}) \tag{10}$$

  Check if the log-likelihood cost function converged or not. If yes, terminate the algorithm, otherwise, repeat the E-M steps and evaluate the log-likelihood again.

Tasks:

1. In the code, in function **calculate_responsibilities()** implement the E-Step of EM algorithm, namely, Equ. 5 that calculates responsibilities (posterior probabilities) for each sample.
2. In the code, in function **update_parameters()** implement the M-Step of EM algorithm, that is, Equ. 7, and 9. Equ. 8 is already implemented.
3. In the code, in function **em()** implement the steps of EM algorithms that are missing. Initialization of the GMM is provided.
4. In **main.py/task_em()**, set K to the value that you used in K-means (in order to compare it with the results from the previous task), and choose *max_iter*, then call function **em**. Use the function **plot_cost** that you implemented previously to plot the log-likelihood (cost function). Include the plot in the report.
5. There is already a function call to plot the Mickey Mouse after clustering. Include the plot in the report. Compare it with the plot of original data (in one or two sentences). Include in the report the initial value of π and the final value (after the algorithm converged).

2.3 Summary and comparison of two algorithms [3 points + 2* points]

Tasks:

1. Which algorithm works better for the Mouse data set? Why? Explain by comparing the plots of the original data, after K-means clustering, and after EM clustering.
2. Are there noisy data points (outliers) in the original data? Is K-means robust to noise? Is EM robust to noise?
3. What is **the main** difference between K-means and EM algorithms?

Bonus tasks [2* points]:

- Components π_k in GMM (used in the EM algorithm) can also be thought of as **prior probabilities** over mixing components (component weights). Which concrete distribution is given by initial values for π (you can check the code, there we have $\pi_k = \frac{1}{K}$)? Can you relate the final values of π_k with clusters (e.g., the maximum value in π is for which cluster)?
- In the EM algorithm, the covariance matrices are updated in each iteration. K-means does not have this parameter. What are the assumed (implicit) covariance matrices for clusters in K-means? What shape of clusters does K-means tend to produce? (Probably, the shapes of clusters, dependent on the covariance matrices, should be also visible in the plot of Mickey Mouse after clustering with K-means.)