Assignment 3 - Q1

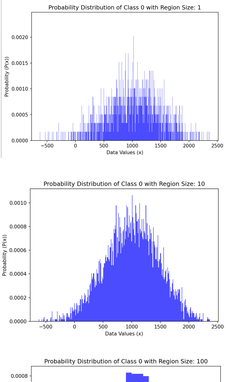
November 4, 2023 9:19 PM

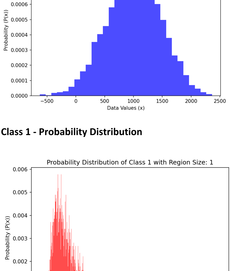
Exercise 1: Non-Parametric Estimation (35 pts)

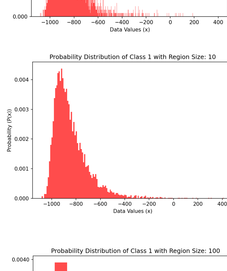
In this exercise you will be using the MNIST dataset for image classification. To work with this dataset, you will first need to flatten your images from 28×28 to 784×1 vectors. Next, use the PCA in scikit learn to convert the 784×1 vectors to 2×1 vectors. Note that the dataset consists of the training and the test sets. Use the training set for implementing the classifiers in the exercise. Also, use only the first two classes in the dataset i.e. the two classes representing numbers 0 and 1.

1. (10 pts) Use histogram-based estimation with region sizes of {1,10,100} to estimate the probability distribution of your dataset. Note that you will need to estimate the distribution of each class separately. Plot your probability distributions for all the region sizes. Which region size seems to be the best for estimating the probabilities? Explain.

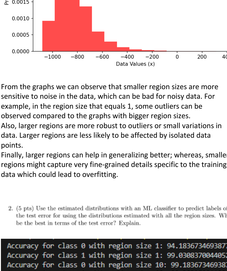
Class 0 - Probability Distribution

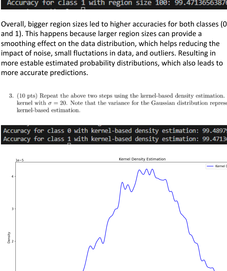


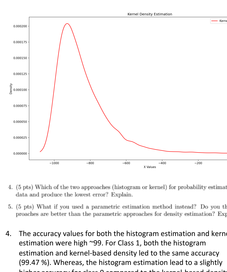




Class 1 - Probability Distribution







From the graphs we can observe that smaller region sizes are more sensitive to noise in the data, which can be bad for noisy data. For example, in the region size that equals 1, some outliers can be observed compared to the graphs with bigger region sizes.
Also, larger regions are more robust to outliers or small variations in data. Larger regions are less likely to be affected by isolated data points.
Finally, larger regions can help in generalizing better; whereas, smaller regions might capture very fine-grained details specific to the training data which could lead to overfitting.

2. (5 pts) Use the estimated distributions with an ML classifier to predict labels of your test data. Report the test error for using the distributions estimated with all the region sizes. Which region size seems to be the best in terms of the test error? Explain.

```
Accuracy for class 0 with region size 1: 94.18367346938776
Accuracy for class 1 with region size 1: 99.03883700440529
Accuracy for class 0 with region size 10: 99.18367346938776
Accuracy for class 1 with region size 10: 99.38325991189427
Accuracy for class 0 with region size 100: 99.69387755102041
Accuracy for class 1 with region size 100: 99.47136563876651
```

Overall, bigger region sizes led to higher accuracies for both classes (0 and 1). This happens because larger region sizes can provide a smoothing effect on the data distribution, which helps reducing the impact of noise, small fluctations in data, and outliers. Resulting in more estable estimated probability distributions, which also leads to more accurate predictions.

3. (10 pts) Repeat the above two steps using the kernel-based density estimation. You can use a Gaussian kernel with $\sigma = 20$. Note that the variance for the Gaussian distribution represents the scaling factor in kernel-based estimation.

```
Accuracy for class 0 with kernel-based density estimation: 99.48079591836735
Accuracy for class 1 with kernel-based density estimation: 99.47136563876651
```


4. (5 pts) Which of the two approaches (histogram or kernel) for probability estimations best represent your data and produce the lowest error? Explain.
5. (5 pts) What if you used a parametric estimation method instead? Do you think non-parametric approaches are better than the parametric approaches for density estimation? Explain.

4. The accuracy values for both the histogram estimation and kernel estimation were high ~99. For Class 1, both the histogram estimation and kernel-based density led to the same accuracy (99.47 %). Whereas, the histogram estimation lead to a slightly higher accuracy for class 0 compared to the kernel-based density estimation. These accuracies were only higher for estimations with bigger region sizes. Compared to the other accuracies obtained with the histogram estimation approach, the kernel-based density approach led to better accuracies.

   Overall, histograms help in representing and providing a discrete representation of the data distribution, which can be useful for discrete data such as the frequency of pixel values in images. However, they might not capture smooth variations within the digits. On the other hand, using kernels such as the Gaussian kernel can provide a smooth and continuous representation of the data distribution, helping in capturing both local and global features. This is desirable for handwritten digits which are characterized for having fine details and broader patterns. Kernels might help in providing a detailed representation of the variations in pixel values within the digits.

5.

```
Accuracy for class 0 with parametric estimation: 99.89795918367346
Accuracy for class 1 with parametric estimation: 98.94273127753304
```

   Parametric density estimation methods assume the distribution of the data, estimating parameters of that distribution. This can lead to accurate results when the assumed distribution closely matches the actual distribution. For data that doesn't match the distribution, it will lead to low accuracies and higher errors. From the accuracies above, using parametric estimation with a gaussian pdf led to a high accuracy for class 0. Whereas, it led to a lower accuracy for class 1. The high accuracy is expected for class 0 since it does have a gaussian distribution as observed in the histograms above. However, using this method results in lower accuracy for class 1 which does not have this gaussian distribution.

   Non-parametric approaches reduces the risk of high errors since those methods do not make strong assumptions about the data distribution, causing more flexibility to capture complex data distributions.

   For data from the MNIST dataset, working with non-parametric approaches is more suitable because they do not impose the distribution of the data; therefore, they can capture more complex data like handwritten digit patterns.

Assignment 3 - Q2

November 6, 2023 2:58 AM

Exercise 2: K-means clustering (40 pts)

NOTE: You are allowed to compare your results with the Scikit implementation of k-means, but you cannot use the Scikit implementation as your own solution.
In this exercise you will be using the MNIST dataset as in the previous exercise. You will also be using 784×1 directly without any PCA. You can use the training data for this exercise. Use all the classes in the dataset.

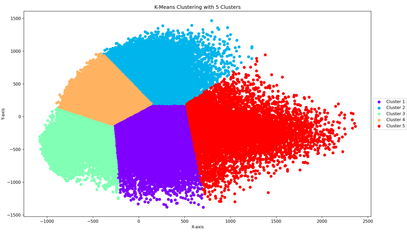
1. (10 pts) Implement the k-means clustering algorithm with euclidean distance as the distance metric.

Check code in python

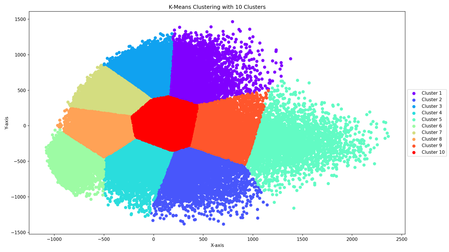
2. (10 pts) Apply your k-means implementation to the MNIST dataset. Use k={5,10,20,40}. Also, you will not be using class labels during the clustering process as it is an unsupervised learning problem.

Plots for 2D data for 60000 points

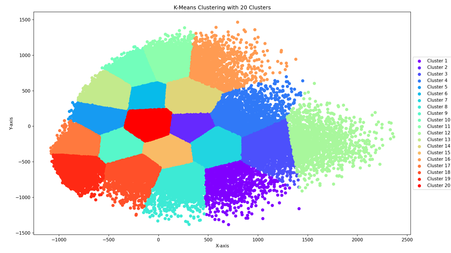
k = 5



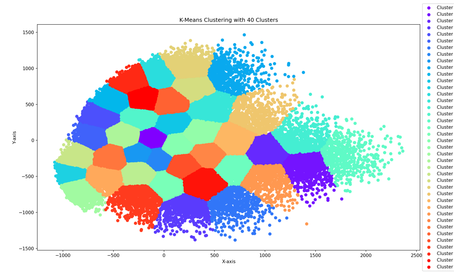
k= 10



k = 20



k = 40



3. (7 pts) As we do not use class labels for clustering we cannot use test error to evaluate the k-means algorithm. Instead, we can test our clustering implementation using cluster consistency: all data points in a cluster should belong to the same class. For example, if your cluster number 1 has all the data points belonging to class "5" then this cluster has perfect consistency. You can find the cluster consistency Q_i for each cluster i as follows:

$$Q_i = \frac{m_i}{N_i} \tag{1}$$

where, N_i is the total number of data points in cluster i, and m_i represents the total number of data points belonging to the most common class in cluster i. You can find m_i by first counting the total number of data points belonging to each of the 10 classes in cluster i, and then taking the max of this list. To find the overall clustering consistency:

$$Q = \frac{1}{k}\sum_{i=1}^{k} Q_i \tag{2}$$

Report the cluster consistency of your k-means clustering on the MNIST dataset for all four values of k.

For all the MNIST dataset:

```
Consistency using own K-mean function
The consistency for k= 5 is:0.1121072982486198 3
The consistency for k= 10 is:0.1128130290487700 2
The consistency for k= 20 is:0.1137755642355919 9
The consistency for k= 40 is:0.1178237634812073
```

```
Consistency using KMeans library from sklearn
The consistency for k= 5 is:0.11216888277149076
The consistency for k= 10 is:0.11257538707091508
The consistency for k= 20 is:0.1139602349490247
The consistency for k= 40 is:0.1175023289853527
```

Overall, the consistency values are low ~0.11-0.12, this suggest that the clustering performed by the K-Means (own implementation) algorithm and the Kmeans (sklearn library) algorithm are not effective in preserving the class structures within the clusters. Low consistency values indicate that the data points are not being grouped in a way that reflects the inherent distribution of classes in a meaningful name. This can be related to the dependency on the initialization of the clusters centers which is random. Running the algortihms with different initializations may lead to different results.

However, the results for both algorithms led to similar consistency results, meaning that my k-means implementation was done properly. The consistecy values only vary by ~ 0.0001.

Using 100 samples from the MNIST dataset led to higher consistency results as shown below:

```
Consistency using own K-mean function
The consistency for k= 5 is:0.2588437001594896
The consistency for k= 10 is:0.330450937950938
The consistency for k= 20 is:0.4359432344322345
The consistency for k= 40 is:0.6425
```

```
Consistency using KMeans library from sklearn
The consistency for k= 5 is:0.2313640307061359 8
The consistency for k= 10 is:0.2969658119658119 5
The consistency for k= 20 is:0.4403571428571428 4
The consistency for k= 40 is:0.6345833333333333
```

4. (8 pts) Which k value produces the best results? Explain. Can the results from cluster consistency be misleading? Explain. [HINT Intuitively, what k value should produce the best results on the MNIST dataset?]

From the above results, it can be observed that the choice the clusters number (k) can potentially influence the consistency, which improves for higher values of k, being the highest accuracy/ consistency value for k =40.

Having, too few clusters can lead to mixed clusters with different classes. On the other hand, having too many clusters can lead to clusters with less data points.

Intuitevely, the best results should be given for k = 10 since the MNIST dataset consists of 10 classes of handwritten digits. With k =10, each cluster could potentially represent a distinct digit class, making the results more interpretable. Having less clusters or more clusters will not lead to a good clustering of the MNIST dataset.

Assignment 3 - Q3

Tuesday, November 7, 2023 5:09 PM

Exercise 3: Gaussian Mixture Model (GMM) (35 pts)

Notation: For a matrix A, $|A|$ denotes its determinant. For a diagonal matrix $\mathrm{diag}(\mathbf{s})$, $|\mathrm{diag}(\mathbf{s})| = \prod_i s_i$.

Note: For 2.1 and 2.2, you only need to implement (or analyze) your algorithm without testing it on any dataset. Implementations in 2.1 and 2.2 will be tested in 2.3. Therefore, any questions related to the testing of your implementation in 2.1 and 2.2 on a dataset, can be answered in 2.3.

Algorithm 1: EM for GMM

Input: $X \in \mathbb{R}^{n\times d}$, $K \in \mathbb{N}$, initialization for *model*
// *model* includes $\pi \in \mathbb{R}^K_+$ and for each $1 \le k \le K$, $\boldsymbol{\mu}_k \in \mathbb{R}^d$ and $S_k \in \mathbb{S}^d_+$
// $\pi_k \geq 0$, $\sum_{k=1}^K \pi_k = 1$, S_k symmetric and positive definite.
// random initialization suffices for full credit.
// Alternatively, can initialize r by randomly assigning each data to one of the K components
Output: *model*, ℓ

1 **for** *iter* $= 1 :$ MAXITER **do**
  // step 2, for each $i = 1, \ldots, n$
2 **for** $k = 1, \ldots, K$ **do**
3 $r_{ik} \leftarrow \pi_k |S_k|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_k)^\top S_k^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)\right]$ // compute responsibility
  // for each $i = 1, \ldots, n$
4 $r_i \leftarrow \sum_{k=1}^K r_{ik}$
  // for each $k = 1, \ldots, K$ and $i = 1, \ldots, n$
5 $r_{ik} \leftarrow r_{ik}/r_i$ // normalize
  // compute negative log-likelihood
6 $\ell[iter] = -\sum_{i=1}^n \log(r_i)$
7 **if** $iter > 1$ && $|\ell[iter] - \ell[iter-1]| \leq \mathrm{TOL} * |\ell[iter]|$ **then**
8 **break**
  // step 1, for each $k = 1, \ldots, K$
9 $r_k \leftarrow \sum_{i=1}^n r_{ik}$
10 $\pi_k \leftarrow r_k/n$
11 $\boldsymbol{\mu}_k = \sum_{i=1}^n r_{ik}\mathbf{x}_i/r_k$
12 $S_k = \sum_{i=1}^n r_{ik}\mathbf{x}_i\mathbf{x}_i^\top/r_k - \boldsymbol{\mu}_k\boldsymbol{\mu}_k^\top$

1. (20 pts) Derive and implement the EM algorithm for the diagonal Gaussian mixture model (dGMM), where all covariance matrices are constrained to be diagonal. Note that the above algorithm does not make any assumptions about the covariance matrices, however, in our class, we assumed the covariances to be identity matrices. Algorithm 1 recaps all the essential steps and serves as a hint rather than a verbatim instruction. In particular, you must change the highlighted steps accordingly (with each S_k being a diagonal matrix), along with formal explanations.

   [You might want to review the steps we took in class to get the updates in Algorithm 1 and adapt them to a bit more complex case here. The solution should look like $s_j \leftarrow \frac{\sum_{i=1}^n r_{ik}(x_{ij}-\mu_{kj})^2}{\sum_{i=1}^n r_{ik}} = \frac{\sum_{i=1}^n r_{ik}x_{ij}^2}{\sum_{i=1}^n r_{ik}} - \mu_{kj}^2$ for the j-th diagonal. Multiplying an $n \times p$ matrix with a $p \times n$ matrix costs $O(mnp)$. Do not maintain a diagonal matrix explicitly; using a vector for its diagonal suffices.]

   To stop the algorithm, set a maximum number of iterations (say MAXITER $= 500$) and also monitor the change of the negative log-likelihood ℓ to

$$\ell = -\sum_{i=1}^n \log\left[\sum_{k=1}^K \pi_k (2\pi)^{-d/2} |S_k|^{-1/2} \exp\left[-\tfrac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_k)^\top S_k^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)\right]\right], \tag{3}$$

   where $\mathbf{x}_i$ is the i-th column of $X^\top$. As a debug tool, note that ℓ should decrease from step to step, and we can stop the algorithm if the decrease is smaller than a predefined threshold, say $\mathrm{TOL} = 10^{-5}$.

2. (15 pts) Next, we apply (the adapted) Algorithm 1 in Ex 2.1 to the MNIST dataset (that you already experimented on before). For each of the 10 classes (digits), we can use its (and only its) training images to estimate its (class-conditional) distribution by fitting a GMM (with any $K = 5$, roughly corresponding to 5 styles of writing this digit). This gives us the density estimate $p(\mathbf{x}|y)$ where $\mathbf{x}$ is an image (of some digit) and y is the class (digit). We can now classify the test set using the Bayes classifier:

$$\hat{y}(\mathbf{x}) = \arg\max_{c=0,\ldots,9} \underbrace{\Pr(Y = c) \cdot p(X = \mathbf{x}|Y = c)}_{\propto \Pr(Y=c|X=\mathbf{x})}, \tag{4}$$

   where the probabilities $\Pr(Y = c)$ can be estimated using the training set, e.g., the proportion of the c-th class in the training set, and the density $p(X = \mathbf{x}|Y = c)$ is estimated using GMM for each class c separately. Report your error rates on the test set as a function of K. Use $5, 10, 20, 30$ as the values for K. Note that if time is a concern, using $K = 5$ will receive full credit.

   [Optional. Reduce dimension by PCA may boost accuracy quite a bit. Your running time should be on the order of minutes (for one K), if you do not introduce extra for-loops in Algorithm 1.]

   [In case you are wondering, our classification procedure above belongs to the so-called plug-in estimators (plug the estimated densities to the known optimal Bayes classifier). However, note that estimating the density $p(X = \mathbf{x}|Y = c)$ is actually harder than classification. Solving a problem (e.g. classification) through some intermediate harder problem (e.g. density estimation) is almost always a bad idea. Extra: 5 pts Do you think this will be true for harder classification problems (such as indoor scene classification)? Explain.]

From notes, we know:

Given Σ, sample x from a Gaussian distribution $\mathcal{N}(x|\mu_k, \Sigma)$

$\Rightarrow$ $p(x|z=k) = \mathcal{N}(x|\mu_k, \Sigma) = \pi_k$

Using Bayes rule: $p(z=k|x) = \frac{p(x|z=k)\,p(z=k)}{p(x)}$ $\rightarrow$ probability of x come from kth cluster.

Now, we want to not assume $z=k$; thus, we use the expected value:

$\mathbb{E}\left[\mathbb{I}[z=k]|x_i\right] = p\left(z_i=k|x_i\right)$

r_k^i

if we do this, we can calculate using ML

$\log p(D) = \sum_{i=1}^N \sum_{k=1}^K r_k^i \left(\log \mathcal{N}(x_i|\mu_k, I) + \log \pi_k\right)$

this replace previous observation with expected value.

$\log p(D) = \sum_{i=1}^N \log p(x_i)$

$p(x_i)$

$p(x_i) = \sum_{k=1}^K \pi_k$

$\Rightarrow \log p(D) = \sum_{i=1}^N \sum_{k=1}^K r_k^i \left(\log \mathcal{N}(x_i|\mu_k, I) + \log \pi_k\right)$

Σ_k is identity matrix

$\Rightarrow \log p(D) = \sum_{i=1}^N \sum_{k=1}^K r_k^i \log \pi_k + r_k^i \log\left(N(x_i|\mu_k, \Sigma)\right)$

- Maximizing this:

for any i, we can also ignore $r_k \log(\pi_k)$ since it'll be zero.

$\Rightarrow \sum_{i=1}^N r_k^i \log\left(\mathcal{N}(x_i|\mu_k, I)\right)$

Maximizing $\sum_{i=1}^N r_k^i \log\left(N(x_i|\mu_k, I)\right)$ is the same as maximizing:

Maximizing / Deriving for μ_k:

$\sum_{i=1}^N r_k^i \left(-\frac{1}{2}\log(\det(\Sigma)) - \frac{1}{2}(x_i-\mu_k)^T \Sigma^{-1}(x_i-\mu_k)\right)$

$\Rightarrow -\frac{1}{2}\sum_{i=1}^N r_k^i \log(\det(\Sigma)) - \frac{1}{2}\sum_{i=1}^N r_k^i (x_i-\mu_k)^T \Sigma^{-1}(x_i-\mu_k)$

$\det(\Sigma) = 1$ where $\log(1) = 0$

identity matrix

$\Rightarrow -\frac{1}{2}\sum_{i=1}^N r_k^i \log(1) - \frac{1}{2}\sum_{i=1}^N r_k^i (x_i-\mu_k)^T I (x_i-\mu_k)$

$\Rightarrow -\frac{1}{2}\sum_{i=1}^N r_k^i (x_i-\mu_k)^T I (x_i-\mu_k)$

Taking the derivative with respect to μ_k:

$\frac{\partial}{\partial \mu_k}\left(\sum_{i=1}^N r_k^i (x_i-\mu_k)^T I (x_i-\mu_k)\right)$

$= \sum_{i=1}^N r_k^i I (\mu_k - x_i) = 0$

$\Rightarrow \sum_{i=1}^N r_k^i \mu_k = \sum_{i=1}^N r_k^i x_i$

$$\mu_k = \frac{\sum_{i=1}^N r_k^i x_i}{\sum_{i=1}^N r_k^i}$$

Deriving for Σ_k

Before assuming Σ_k is I

$$(*) \quad -\frac{1}{2}\sum_{i=1}^N r_k^i \log\left(\det(\Sigma_k)\right) - \frac{1}{2}\sum_{i=1}^N r_k^i (x_i-\mu_k)^T \Sigma_k^{-1}(x_i-\mu_k)$$

We know $(A^{-1}) = \det(A)^{-1}$

$\Rightarrow -\frac{1}{2}\sum_{i=1}^N r_k^i \log\left(\det(\Sigma_k)\right) = \frac{1}{2}\sum_{i=1}^N r_k^i \log\left(\det(\Sigma_k^{-1})\right)$

$\Rightarrow$ We can write (*) as:

$$\frac{1}{2}\sum_{i=1}^N r_k^i \log\left(\det(\Sigma_k^{-1})\right) - \frac{1}{2}\sum_{i=1}^N r_k^i (x_i-\mu_k)^T \Sigma_k^{-1}(x_i-\mu_k)$$

Taking the derivative with respect to Σ_k^{-1}, we have:

$$\frac{\partial}{\partial \Sigma_k^{-1}}\left(\frac{1}{2}\sum_{i=1}^N r_k^i \log\left(\det(\Sigma_k^{-1})\right) - \frac{1}{2}\sum_{i=1}^N r_k^i (x_i-\mu_k)^T \Sigma_k^{-1}(x_i-\mu_k)\right)$$

$$\Rightarrow \frac{1}{2}\sum_{i=1}^N r_k^i \Sigma_k - \frac{1}{2}\sum_{i=1}^N r_k^i (x_i-\mu_k)(x_i-\mu_k)^T = 0$$

$$\Sigma_k = \frac{\sum_{i=1}^N r_k^i (x_i-\mu_k)(x_i-\mu_k)^T}{\sum_{i=1}^N r_k^i}$$

Assume $(x_i-\mu_k)$ is symmetric $\Rightarrow (x_i-\mu_k)^T = (x_i-\mu_k)$

$$\Rightarrow \Sigma_k = \frac{\sum_{i=1}^N r_k^i (x_i-\mu_k)^2}{\sum_{i=1}^N r_k^i}$$

Replacing $\mu_k = \frac{\sum_{i=1}^N r_k^i x_i}{\sum_{i=1}^N r_k^i}$

$$\Rightarrow \Sigma_k = \frac{\sum_{i=1}^N r_k^i \left(x_i - \frac{\sum_{i=1}^N r_k^i x_i}{\sum_{i=1}^N r_k^i}\right)^2}{\sum_{i=1}^N r_k^i}$$

$$\Sigma_k = \frac{\sum_{i=1}^N r_k^i x_i^2 - 2\sum_{i=1}^N r_k^i x_i^2 + \left(\frac{\sum_{i=1}^N r_k^i x_i}{\sum_{i=1}^N r_k^i}\right)^2}{\sum_{i=1}^N r_k^i}$$

$$\Sigma_k = \frac{\sum_{i=1}^N r_k^i x_i^2}{\sum_{i=1}^N r_k^i} - \mu_k^2$$

$$\Sigma_k = \frac{\sum_{i=1}^N r_k^i x_i^2}{\sum_{i=1}^N r_k^i} - \mu_k^2$$