Assignment 1

Machine Learning 2

Lea Bogensperger, lea.bogensperger@icg.tugraz.at
Benedikt Kantz, benedikt.kantz@student.tugraz.at

April 9, 2024

Deadline: April 30, 2024 at 23:55h.
Submission: Upload your report (`report.pdf`), your implementation (`main.py`) and your figure file (`figures.pdf`) to the TeachCenter. Please do not zip your files. Use the provided file containing the code framework for your implementation.

1 Transformation of Probability Distributions (4P)

Assume we are given a probability density function (pdf) $p_X(x)$. Now for a given non-linear change of variables $z = f(x)$ we can compute the resulting pdf $p_Z(z)$ using

$$p_Z(z) = \sum_{x, f(x)=z} \frac{p_X(x)}{|f'(x)|},$$

where the summation explicitely includes *all* x for which $f(x) = z$. Assume we are given a pdf $p_X(x) = \frac{\exp(-\frac{1}{2}x^2)}{\sqrt{2\pi}}$ and a transformation $f(x) = x^2$.

Tasks

1. Compute $p_Z(z)$ for the transformed random variable Z. Show all steps in your report.
2. State the two properties that a valid pdf must satisfy (see Lecture slides) and verify whether this holds for your transformed pdf $p_Z(z)$. Document your steps of verifying the pdf in your report.

Implementation details

- Since the transformed pdf does not admit a closed-form solution for the indefinite integral, you can use `scipy.integrate.quad` here to numerically approximate an integral.

Implement this task in `task1` of `main.py`.

2 Gaussian Mixture Model (GMM) (15P)

Next, the task is to fit a multivariate gaussian mixture model (GMM) to the FashionMNIST data set [1] such that it can be sampled from or used in a different inverse problem such as denoising or inpainting. The official training data set consists of $\mathbf{x} = \{\mathbf{x}^1, \dots, \mathbf{x}^S\}$ with S training images (see Figure 1 for examples of a sampled subset) and each vectorized image is of size $\mathbf{x}^s \in \mathbb{R}^D$, with $D = M \cdot M$ and $M = 28$.

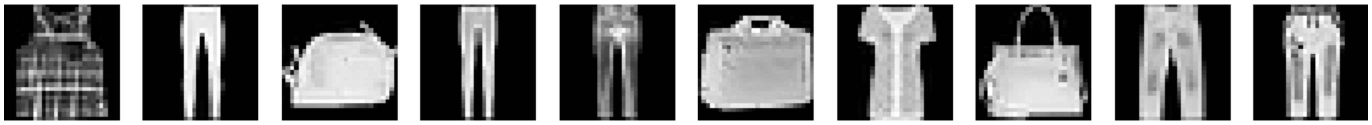

Figure 1: Exemplary training images from a subset of the FashionMNIST data set where $S = 2000$ random images exclusively from the labels $0, 1, 8$ were sampled.

In this setting where we are interested in density estimation, we can use a GMM with K components to represent our data. As we are dealing with training samples each of size $\mathbf{x}^s \in \mathbb{R}^{M \cdot M}$, we require a multivariate GMM to model our prior/data distribution

$$p(\mathbf{x}) = \prod_{s=1}^{S} \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}^s | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{1}$$

where each individual multivariate Gaussian is naturally given by

$$\mathcal{N}(\mathbf{x}^s | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{(2\pi)^{D/2} |\boldsymbol{\Sigma}_k|^{1/2}} \exp\big(-\tfrac{1}{2} (\mathbf{x}^s - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\mathbf{x}^s - \boldsymbol{\mu}_k) \big).$$

The parameters of the GMM can be fitted using an Expectation-Maximization (EM) algorithm to minimize the negative log-likelihood, given in Algorithm 1.

```
Algorithm 1: EM algorithm.
  Set stopping threshold $\epsilon_1$, maximum number of iterations $J$, set iteration counter $j = 1$
  Initialize $\boldsymbol{\mu}_k^0, \boldsymbol{\Sigma}_k^0, \pi_k^0$ for $k = 1, \dots, K$
  while $| -\log p(\boldsymbol{x} | \boldsymbol{\mu}^j, \boldsymbol{\Sigma}^j, \pi^j) + \log p(\boldsymbol{x} | \boldsymbol{\mu}^{j-1}, \boldsymbol{\Sigma}^{j-1}, \pi^{j-1})| \geq \epsilon_1$ do
      $(w_k^s)^j = \frac{\pi_k^{j-1} \mathcal{N}(\mathbf{x}^s | \boldsymbol{\mu}_k^{j-1}, \boldsymbol{\Sigma}_k^{j-1})}{\sum_{k=1}^{K} \pi_k^{j-1} \mathcal{N}(\mathbf{x}^s | \boldsymbol{\mu}_k^{j-1}, \boldsymbol{\Sigma}_k^{j-1})}$   – $\frac{D}{2}$ in exponent too high → cancels out?
      $N_k^j = \sum_{s=1}^{S} (w_k^s)^j$
      $\boldsymbol{\mu}_k^j = \frac{\sum_{s=1}^{S} (w_k^s)^j \mathbf{x}^s}{N_k^j}$
      $\boldsymbol{\Sigma}_k^j = \frac{1}{N_k^j} \sum_{s=1}^{S} (w_k^s)^j (\mathbf{x}^s - \boldsymbol{\mu}_k)(\mathbf{x}^s - \boldsymbol{\mu}_k)^T$
      $\pi_k^j = \frac{N_k^j}{S}$
      $j = j + 1$
  end
  Output: fitted parameters $\{\boldsymbol{\mu}, \boldsymbol{\Sigma}, \pi\} = \{\boldsymbol{\mu}^j, \boldsymbol{\Sigma}^j, \pi^j\}$
```

GMMs are very sensitive to initialization, therefore using the k-means algorithm is a very popular strategy, which is described in Algorithm 2.

```
Algorithm 2: k-means algorithm.
  Set stopping threshold $\epsilon_2$, maximum number of iterations $J$, set iteration counter $j = 1$
  Randomly pick $K$ k-means centroids $\boldsymbol{\mu}_k$ from your data set.
  while stopping criterion not fulfilled do
      $D_k^j = \{\mathbf{x}^s : \|\mathbf{x}^s - \boldsymbol{\mu}_k^j\|_2^2 \leq \|\mathbf{x}^s - \boldsymbol{\mu}_i^j\|_2^2 \quad \forall i,\ 1 \leq i \leq K\}$
      $\boldsymbol{\mu}_k^{j+1} = \frac{1}{|D_k^j|} \sum_{\mathbf{x}^t \in D_k^j} \mathbf{x}^t$
      $j = j + 1$
  end
  Output: $K$ centroids $\boldsymbol{\mu} = \boldsymbol{\mu}^j$
```

Tasks

1. State the dimensions of all GMM parameters depending on K components and the dimensionality M.

2. The loading of the $S = 2000$ training images is already provided for you, see Figure 1 for exemplary images. Note that for simplicity, we only use images containing the labels $0, 1, 8$ – corresponding to T-Shirt/Top (0), Trousers (1) and Bag (8). Choose a reasonable number of GMM components. Initialize your weights using a uniform distribution, and use the identity matrix to initialize the covariance matrices.
3. The means initialization is crucial, as the GMM can easily get trapped in local optima. Therefore, we use a k-means algorithm for the initialization of the means $\boldsymbol{\mu}_k$. Implement this in your code as given in Algorithm 2, where you decide on a reasonable stopping criterion to evaluate the performance of the k-means algorithm and explain this in the report. Note that the k-means algorithm always iterates between the two steps:
   (a) **assignment step**: assign each data sample $\mathbf{x}^s$ to a cluster k with mean $\boldsymbol{\mu}_k$, such that you have sets $\mathbf{D}_k = \{D_{k,1}, \dots, D_{k,L}\}$ with L samples per cluster (they can be different among each cluster),
   (b) **update step**: re-compute the means for the newly assigned cluster samples.
4. Fit your GMM parameters by implementing Algorithm 1. Take a close look at the implementation details for tipps on efficient and numerically stable implementations. As in the previous assignment, report what kind of stopping criterion you used.
5. Plot the means and the covariances of all fitted GMM components (reshape them appropriately!) and include the respective weights π_k in the title of each subplot. Discuss the results.
6. Since we have a generative model that estimates the density of the underlying data, we can actually sample from it. For our GMM this is done by drawing a mixture component k with probabilities π and then sample from this component with the fitted parameters $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$. Generate and plot 10 samples, denote the sampled components in each subtitle.

Implementation details

- For numerical stability when computing the responsibilities, use the log-sum-exp trick to do your computations in the log-domain (note that $\mathbf{y}_k$ is an arbitrary variable here where we sum over dimension k, it is your task to properly adapt this to the setting):

$$\log \sum_{k=1}^{K} \exp(\mathbf{y}_k) = \max_k(\mathbf{y}_k) + \log \sum_{k=1}^{K} \exp(\mathbf{y}_k - \max_k(\mathbf{y}_k)).$$

- When inverting your covariance matrix $\boldsymbol{\Sigma}_k$, add a small offset of $1e-6$ to the main diagonal for stability.
- When you compute $|\boldsymbol{\Sigma}_k|^{1/2}$, it can be advantageous to use a Cholesky decomposition[1] to represent your matrix using a lower-triangular matrix $\mathbf{L}$ – you might also need the trick of adding a small offset to the main diagonal of the covariance matrix first. Make use of the property $|\mathbf{AB}| = |\mathbf{A}||\mathbf{B}|$ when handling matrix determinants.
- For sampling, also use the Cholesky decomposition to sample from the selected mixture component k.
- Note that when plotting means $\boldsymbol{\mu}_k$ it is common to use a `gray` colormap, whereas for the covariances $\boldsymbol{\Sigma}_k$ the convention is to use `viridis` (default).

3 Conditional GMM: Inpainting (6P)

A GMM with fitted parameters can not only be used for sampling, but also for other tasks such as image inpainting (see Figure 2), which can even be computed in closed form. This is based on the principle of conditioning a multivariate Gaussian on a random variable, which can also be extended to a mixture of Gaussians.

[1] You can do this using `numpy.linalg.cholesky`.

x_2

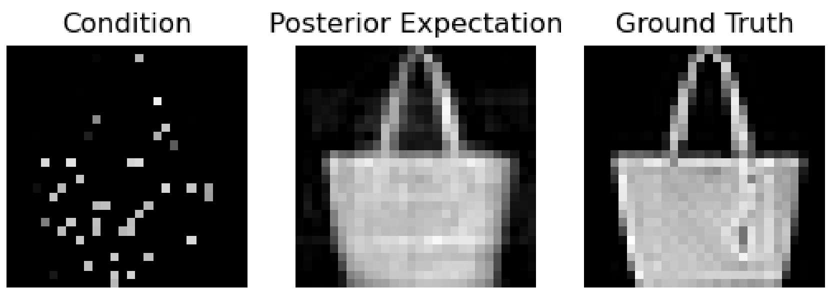

Figure 2: Conditional GMM used for image inpainting using 10% of the original pixel values, where a given corrupted conditioning image can be used to compute the posterior expectation of the conditional GMM.

Let us denote the pixels to condition on as $\mathbf{x}_2$, and the part of the image that has to be restored as $\mathbf{x}_1$. Equivalently, the mean and covariance matrix of a single Gaussian can be partitioned using

$$\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \boldsymbol{\mu}_2)^T, \qquad \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix},$$

respectively. Thus, the conditional mean and covariance for the individual Gaussian can be computed as follows:

$$\boldsymbol{\mu}_{1|2} = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{x}_2 - \boldsymbol{\mu}_2)$$
$$\boldsymbol{\Sigma}_{1|2} = \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}.$$

You task will be to expand the conditioning to a GMM and to apply this on a given subset of the FashionMNIST test set.

Tasks

1. The loading of 10 test images $\mathbf{x}$ is already provided in the code. Generate a corruption mask $\mathbf{m}$ such that it masks random 90% of the pixels when it is applied element-wise for $i = 1, \dots, M^2$ to each image $\mathbf{x}$ by

$$x_i = \begin{cases} x_i & \text{if } m_i = 1, \\ 0, & \text{if } m_i = 0. \end{cases}$$

   Vectorize your images and apply the corruption mask accordingly to mask out a fraction of the images.

2. Show how to analytically compute the posterior of the conditional GMM. Use the product rule as a starting point and document all steps in your report. Show how this is again a proper GMM by introducing $\pi_{k,1|2}$.

3. Show that the expectation of the posterior is given by

$$\mathbb{E}[\mathbf{x}_1|\mathbf{x}_2] = \sum_{k=1}^{K} \pi_{k,1|2}\, \boldsymbol{\mu}_{k,1|2}.$$

   -> i don't even need the posteriors to calculate the posterior expectation!

4. Implement the conditional GMM and compute the posterior computation and merge your result with the conditioned image $\mathbf{x}_2$.

5. For all 10 corrupted test samples, plot the corrupted image together with the restored (posterior expectation) and the ground truth images.

Implementation details

- For computing the conditional means and covariances, make sure you properly index the respective entries of your original means and covariances by using the corruption mask indices. Check all resulting dimensions carefully.

References

[1] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms, 2017.