Course 67564 2021/22

Bayesian Machine Learning
Exercise 6: Gibbs Sampling and Gaussian Mixture Models

Prof. Yair Weiss *TA: Roy Friedman*

Deadline: January 16, 2022

1 Theoretical

1.1 EM Initialization

Recall that the EM algorithm for finding an approximate ML solution for GMMs is governed by the following equations:

$$\text{E-step:} \qquad r_{ik}^{(t)} \triangleq \frac{\pi_k^{(t-1)} \mathcal{N}\left(x_i \,|\, \mu_k^{(t-1)},\, \Sigma_k^{(t-1)}\right)}{\sum_{k'} \pi_{k'}^{(t-1)} \mathcal{N}\left(x_i \,|\, \mu_{k'}^{(t-1)},\, \Sigma_{k'}^{(t-1)}\right)}$$

$$\begin{aligned} & \pi_k^{(t)} = \frac{1}{N} \sum_i r_{ik}^{(t)} \\ \text{M-step:} \qquad & \mu_k^{(t)} = \frac{\sum_i r_{ik}^{(t)} x_i}{\sum_i r_{ik}^{(t)}} \\ & \Sigma_k^{(t)} = \frac{\sum_i r_{ik}^{(t)} \left(x_i - \mu_k^{(t)}\right)\left(x_i - \mu_k^{(t)}\right)^T}{\sum_i r_{ik}^{(t)}} \end{aligned} \tag{1.1}$$

1. Which solution does the EM algorithm converge to if we initialize the algorithm with $\mu_1^{(0)} = \mu_2^{(0)} = \cdots = \mu_K^{(0)}$ and $\Sigma_1^{(0)} = \Sigma_2^{(0)} = \cdots = \Sigma_K^{(0)}$?

1.2 Gibbs Sampling

Consider the following distribution:

$$p(x, y) = \frac{1}{Z} \mathcal{N}(x \,|\, \mu_x,\, \Sigma_x)\, \mathcal{N}(y \,|\, \mu_y,\, \Sigma_y)\, e^{-\frac{\beta}{2}\|x-y\|^2} \tag{1.2}$$

where Z is a normalization constant and x and y are vectors.

2. Find the conditional distributions needed for the Gibbs sampling algorithm under the given distribution $p(x, y)$

2 Robust Regression

Consider the problem of linear regression with outliers (also called *robust regression*), as defined in class, where we assume that:

$$p(y_i|\theta, x_i, k_i) = \begin{cases} \mathcal{N}\left(y_i|\, h(x_i)^T \theta,\, \sigma^2\right) & k_i = 0 \\ \mathcal{N}\left(y_i|\, \mu_o,\, \sigma_o^2\right) & k_i = 1 \end{cases} \tag{2.1}$$

where $h(\cdot)$ are some basis functions, x_i is the input data, σ^2 is the sample noise, μ_o is the mean of the outliers, σ_o^2 is the variance of the outliers and k_i dictates whether the data point is an inlier (a standard data point) in which case $k_i = 0$ or an outlier, in which case $k_i = 1$. As we have seen in class, if we assume a Gaussian prior over θ:

$$\theta \sim \mathcal{N}(\mu, \Sigma) \tag{2.2}$$

then we can sample from $\theta|\mathcal{D}$ directly, but to do this we will need to create a GMM with 2^N Gaussians, where N is the number of data points. This is impractical when we have even a humble amount of data points.

Instead, we will use Gibbs sampling in order to generate samples from $p(\theta, k|\mathcal{D})$, where $k \in \mathbb{R}^N$ is a vector whose coordinates indicate whether each point is an outlier, in which case k_i =1, and otherwise $k_i = 0$. The Gibbs sampling algorithm is comprised of two alternating steps that are repeated until convergence:

$$(1) \quad k^{(t)} \sim p\left(k|\, \theta^{(t-1)}, \mathcal{D}\right) \tag{2.3}$$

$$(2) \quad \theta^{(t)} \sim p\left(\theta|\, k^{(t)}, \mathcal{D}\right) \tag{2.4}$$

Let's write $\mathcal{D}_{\text{in}}^{(t)} \triangleq \{(x_i, y_i)\}_{i:\, k_i^{(t)}=0}$, which will make the second step easy to describe:

$$(2) \quad p\left(\theta|\, k^{(t)}, \mathcal{D}\right) = p\left(\theta|\, \mathcal{D}_{\text{in}}^{(t)}\right) \tag{2.5}$$

where $p\left(\theta\,|\,\mathcal{D}_{\text{in}}^{(t)}\right)$ is just the posterior for linear regression we have seen throughout the whole course, applied only to the data points which $k^{(t)}$ decides are not outliers. Notice that once we decide which points are outliers, we can also use kernels and the dual space in order to sample the parameters (this will be like assuming that our parameters are α and the basis functions are defined by the kernel).

The first step isn't much harder. For each point we will ask whether it is an outlier or not:

$$(1) \quad \forall i\; p\left(k_i = 1|\, \theta^{(t)}, \mathcal{D}\right) = \frac{p(k_i = 1)\,\mathcal{N}\left(y_i|\, \mu_o,\, \sigma_o^2\right)}{p(k_i = 1)\,\mathcal{N}\left(y_i|\, \mu_o,\, \sigma_o^2\right) + p(k_i = 0)\,\mathcal{N}\left(y_i|\, h(x_i)^T\, \theta^{(t)},\, \sigma^2\right)} \tag{2.6}$$

We will use robust regression on two data sets. The first data set is of 1D points on a line, where the outliers are really easy to see. To create this data use the `outlier_data()` function from `ex6_utils.py`, which creates N data points with explicitly defined outliers. The second data set we will use is the same dogs vs. frogs from the previous exercise. To fit this data, we will use kernel regression in step (2), instead of the regular Bayesian linear regression - notice that the algorithm itself is exactly the same! This data can be loaded using the function `load_dogs_vs_frogs()` from `ex6_utils.py`, along with an implementation of the polynomial kernel in the function `poly_kernel()`.

1. Implement the Gibbs sampling algorithm for robust regression. To do so, you may use the implemented class `BayesianLinearRegression` (which can receive kernel functions as well as basis functions) from `ex6_utils.py`
2. Sample $N = 50$ points from the `outlier_data()` function and use the Gibbs sampling algorithm for robust regression in order to fit the model using $T = [0, 1, 5, 10, 25]$ Gibbs sampling iterations. The model you should use is a 1st order polynomial with $\theta \sim \mathcal{N}(0, I)$, sample noise $\sigma^2 = 0.15$, outlier probability is $p(k_i = 1) = 0.1$, $\mu_o = 4$ and $\sigma_o^2 = 2$. How many iterations (on average) are needed for convergence? Plot the fit found after $T = [0, 1, 5, 10, 25]$ iterations together with the data points
3. Sample $N = 1000$ points from each class of the dogs vs. frogs data set. Use the Gibbs sampling algorithm for robust regression in order to fit the model using $T = 50$ iterations, with a polynomial kernel of degree 2, sample noise $\sigma^2 = 0.001$, $\mu_o = 0$, $\sigma_o^2 = .5$ and $p(k_i = 1) = 0.01$. Plot all of the images that have been deemed to be outliers

3 Bayesian Gaussian Mixture Models

In the setting of the Bayesian GMM, we assume that we have a prior for the mixture probabilities, over the means and over the covariances. Specifically, we will assume that:

$$\theta = \left\{\pi, \{\mu_k\}_{k=1}^K, \{\Sigma_k\}_{k=1}^K\right\} \tag{3.1}$$

where $\pi \in \mathbb{R}^K$ such that $\forall k\ \pi_k \geq 0$ and $\sum_k \pi_k = 1$, $\mu_k \in \mathbb{R}^d$ and $\Sigma_k \in \mathbb{R}^{d\times d}$ are positive definite matrices. For the Bayesian GMM, we will assume the following priors over the parameters:

$$\pi \sim \text{Dirichlet}\left(\alpha_1, \cdots, \alpha_K\right) \qquad \alpha_i > 0 \tag{3.2}$$

$$\mu_k \sim \mathcal{N}\left(\mu_0, I\sigma_0^2\right) \qquad \mu_0 \in \mathbb{R}^d,\ \sigma_0 > 0 \tag{3.3}$$

$$\Sigma_k \sim \mathcal{W}^{-1}\left(\nu + d - 1, \nu \cdot I\beta\right) \qquad \beta, \nu > 0 \tag{3.4}$$

Recall we saw in class how to use the Gibbs sampling algorithm to sample a Bayesian GMM. For this algorithm, we defined a new (auxiliary) vector $z \in \{1, \cdots, K\}^N$ where each coordinate z_i indicates which Gaussian the i-th data point belongs to. Using this new vector z, the Gibbs sampling algorithm is defined by the following steps:

$$\begin{aligned}
&(1)\ \forall i \in [N] \quad z_i|\theta, \mathcal{D} \sim \text{Categorical}\left(q_{i1}, \cdots, q_{iK}\right) \qquad q_{ik} \triangleq \frac{\pi_k \mathcal{N}\left(x_i \,|\, \mu_k, \Sigma_k\right)}{\sum_{k'} \pi_{k'} \mathcal{N}\left(x_i \,|\, \mu_{k'}, \Sigma_{k'}\right)} \\
&(2)\ \pi|z, \mathcal{D} \sim \text{Dirichlet}\left(N_1 + \alpha_1, \cdots, N_K + \alpha_K\right) \qquad N_k \triangleq \sum_i \mathbb{1}\left[z_i = k\right] \\
&(3)\ \forall k \in [K] \quad \mu_k|z, \mathcal{D}, \Sigma_k \sim \mathcal{N}\left(\left(N_k\Sigma_k^{-1} + I\frac{1}{\sigma_0^2}\right)^{-1}\left(\Sigma_k^{-1}\sum_{i:\, z_i=k} x_i + \frac{1}{\sigma_0^2}\mu_0\right), \left(N_k\Sigma_k^{-1} + I\frac{1}{\sigma_0^2}\right)^{-1}\right)
\end{aligned} \tag{3.5}$$

which we will repeat for T iterations. To sample from the Dirichlet distribution in step (2) you can use the function `np.random.dirichlet`.

The only missing piece is how to sample Σ_k given the rest of the parameters (and the data). In this exercise you will implement two versions of the algorithm - one where Σ_k is fixed and in the other it will be learned.

3.1 Fixed Covariance

As part of this exercise, we will fit a GMM to images. Because fitting a covariance matrix to high-dimensional data is computationally very expensive, we will use a fixed isotropic covariance when fitting to images. In practice, this means that the covariance of each Gaussian will be fixed and set to $\Sigma_k = I\beta$ for some $\beta > 0$. This also allows for much more computationally efficient updates for steps (1) and (3) as well:

$$\Sigma_k = I\beta \quad \Leftrightarrow \quad \log \mathcal{N}\left(x_i \,|\, \mu_k, \Sigma_k\right) = -\frac{1}{2}\left[d\log\left(2\pi \cdot \beta\right) + \frac{1}{\beta}\|x_i - \mu_k\|^2\right] \tag{3.6}$$

$$\Sigma_k = I\beta \quad \Leftrightarrow \quad \mu_k|z, \mathcal{D}, \Sigma_k \sim \mathcal{N}\left(\frac{\frac{1}{\beta}\sum_{i:\, z_i=k} x_i + \frac{1}{\sigma_0^2}\mu_0}{\frac{N_k}{\beta} + \frac{1}{\sigma_0^2}}, \frac{1}{\frac{N_k}{\beta} + \frac{1}{\sigma_0^2}} I\right) \tag{3.7}$$

3.2 Learning the Covariance

While learning the covariance *is* a computationally expensive operation, the resulting algorithm is much more expressive, which is why it is worth exploring as well. However, to do so we need to sample from the inverse-Wishart distribution ($\mathcal{W}^{-1}$ that we saw in the previous exercise) which is kind of troublesome, so instead we will take the MMSE at each iteration:

$$(4)\ \ \Sigma_k \leftarrow \mathbb{E}\left[\Sigma_k|z, \mathcal{D}, \mu_k\right] = \frac{\nu \cdot I\beta + \sum_{i:\, z_i=k}\left(x_i - \mu_k\right)\left(x_i - \mu_k\right)^T}{\nu + N_k} \tag{3.8}$$

3.3 Quality of Clustering

The act of fitting a GMM to data can be thought of clustering; each point is assigned a cluster by finding under which Gaussian is most likely to have generated it. While in 2D it is straightforward to check how good the clustering algorithm performs (we can just look at the points), in higher dimensions it is not exactly trivial. In this

exercise we will be clustering images, which are of a very high dimension. While we can visually check how good of a job the GMM did manually, this would involve looking at all of the images, which isn't really practical. Instead, since we do have access to the ground truth labels, we can try and think of other metrics that will tell us whether the GMM clusters the images well or not.

One of the most popular metrics for clustering algorithms is called the *cluster purity*. Specifically, suppose that a set of images $\{x_1, \cdots, x_M\}$ are clustered together and suppose that their ground truth labels are $\{y_1, \cdots, y_M\}$. Then the cluster purity is given by:

$$\text{purity}\left(y_1, \cdots, y_M\right) = \frac{1}{M} \max_k \left|\{i \,|\, y_i = k\}\right| \equiv \frac{1}{M} \max_k \left[\sum_i \mathbb{1}\left[y_i = k\right]\right] \tag{3.9}$$

Intuitively, the purity measures how much of the cluster is composed of the label that appears the most in the cluster. In the file `ex6_utils.py` you are supplied with the function `cluster_purity()`, which calculates the purity of a cluster (see `purity_example()` for an example of how to use this function).

3.4 Clustering Data

In `ex6_utils.py` there is a function named `gmm_data()`, which receives as an input the number of points N and number of clusters K to use and returns N 2D points from K clusters. In the following questions you should use this function in order to generate the 2D data that you will fit your GMMs to. You are also supplied with the function `load_MNIST()` which loads images from the MNIST data set (as well as their labels); we will use the MNIST data set to fit a GMM over images.

4. Implement a Bayesian GMM model that receives as an input the prior parameters. Remember to add a flag that indicates whether or not the covariance should be learned (in which case step (4) from equation 3.8 is added to the Gibbs sampling algorithm)

5. Sample 1000 points from a GMM with 5 clusters using the `gmm_data()` function. Use $T = 100$ iterations of the Gibbs sampling algorithm described by steps $(1) - (4)$ in order to sample 5 different GMMs, using $K = 50$ clusters, $\alpha_k = 0.01$, $\mu_0 = 0$, $\sigma_0^2 = \frac{1}{2}$, $\nu = 5$ and $\beta = \frac{1}{2}$. For each of the sampled GMMs, plot a histogram of the mixture probabilities π found using the Gibbs sampling algorithm (in descending order) as well as the found Gaussians on top of the scattered data[1]. How many clusters (on average) had a probability of $\pi_k > 10^{-4}$?

6. Fit a Bayesian GMM with fixed covariances $\Sigma_k = I\beta$ to the MNIST data set, using $K = 500$ clusters, $\alpha_k = 1$, $\mu_0 = \frac{1}{2}\mathbf{1}$, $\sigma_0 = \frac{1}{10}I$ and $\beta = \frac{1}{4}$

7. For each image, find which cluster it is most likely to belong to (cluster the images according to the GMM). Plot the 25 images from the 5 clusters with the top purity values and from the clusters with the bottom 5 purity values

4 Submission Guidelines

Submit a single zip file named "`ex6_<YOUR ID>.zip`". This file should contain your code, along with an "`ex6.pdf`" file in which you should add the figures/text for the practical part. Please write readable code, as the code will also be checked manually (and you may find it useful in the following exercises). In the submitted code, please make sure that you write a basic main function in a file named "`ex6.py`" that will run (without errors) and produce all of the results that you showed in the pdf of answers that you submitted. The only packages you should use are `numpy`, `scipy` and `matplotlib`. You may also reuse code from your previous exercise in order to answer the questions in this exercise, if needed.

In general, it is better if you type your homework, but if you prefer handwriting your answers, please make sure that the text is readable when you scan it.

Part of your assignment will be graded by submitting your answers through Moodle, at this link. In each of the questions, write the answer to the corresponding question for grading. These answers will be graded automatically, so write only numeric values where needed.

[1] You can use the function `plot_2D_gmm()` in order to plot the found GMM and the points

5 Supplementary Code

In the file `ex6_utils.py` you can find an example of how to load the supplied data as well as a few helper functions. In addition, the same file has implementations of the `BayesianLinearRegression` class as well as the `polynomial_basis_functions()` function for polynomial basis functions and the `poly_kernel()` implementation of the polynomial kernel. You can use this code as you see fit, and change any part of it that you want, just be sure to submit it as well if you change it. Finally, we have also supplied an outline code which you can use to get started in `ex6.py`. You don't have to use the format we outlined, but your code must run without errors and you must submit the plots required in the exercise description.

Good luck!