CS480/680: Introduction to Machine Learning

Homework 4

Due: 11:59 pm, July 23, 2021, submit on CrowdMark.

Submit your writeup in pdf and all source code in a zip file (with proper documentation). Write a script for each programming exercise so that the TAs can easily run and verify your results. Make sure your code runs!
[Text in square brackets are hints that can be ignored.]

Exercise 1: Gaussian Mixture Model (GMM) (10 pts)

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

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 $S_k \in \mathbb{S}_+^d$
   // $\pi_k \ge 0$, $\sum_{k=1}^K \pi_k = 1$, $S_k$ 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, $\ell$
1  for $iter = 1 : \text{MAXITER}$ do
       // step 2, for each $i = 1, \ldots, n$
2      for $k = 1, \ldots, K$ do
3          $r_{ik} \leftarrow \pi_k |S_k|^{-1/2} \exp[-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_k)^\top S_k^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_k)]$    // 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 $iter > 1$ && $|\ell(iter) - \ell(iter-1)| \le \text{TOL} * |\ell(iter)|$ 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     $S_k \leftarrow \left(\sum_{i=1}^n r_{ik} \mathbf{x}_i \mathbf{x}_i^\top / r_{.k}\right) - \boldsymbol{\mu}_k \boldsymbol{\mu}_k^\top$
```

a) (5 pts) Derive and implement the EM algorithm for the diagonal Gaussian mixture model, where all covariance matrices are constrained to be diagonal. 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. Analyze the space and time complexity of your implementation.

[You might want to review the steps we took in class for a simpler case, and ensure you can derive the updates in Algorithm 1. Then adapt the steps to the simpler diagonal case. The solution should look like $s_j = \frac{\sum_{i=1}^n r_{ik}(x_{ij} - \mu_j)^2}{\sum_{i=1}^n r_{ik}} = \frac{\sum_{i=1}^n r_{ik} x_{ij}^2}{\sum_{i=1}^n r_{ik}} - \mu_j^2$ for the j-th diagonal. Multiplying an $n \times p$ matrix with a $p \times m$ matrix costs $O(mnp)$. Do not maintain a diagonal matrix explicitly; using a vector for its diagonal suffices.]

[Warning: Either in this part or the next one, you may run into issues involving NaNs. You will have to diagnose these issues and fix them.]

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

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

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 $\text{TOL} = 10^{-5}$.

Run your algorithm on `gmm_dataset.csv`, for $k = 1$ to 10. Generate a plot with k on the x-axis and the negative log-likelihood of the data under the final trained model on the y-axis. What do you think the most appropriate choice of k is? Explain and justify how and why you chose this value. [You may want to focus on more than just maximizing the log-likelihood.] For your chosen value of k, report the parameters (mixing weights, mean vectors, and vectors corresponding to the diagonals of the covariance matrices) of your trained model. When reporting them, sort the components in increasing order of mixing weights.

Ans:
Derive the EM algorithm for the diagnoal Gaussian mixture model:
First of all, the probability density function of a given vector x in n-dimension is:

$$p(x) = \frac{1}{(2\pi)^{\frac{n}{2}} |S_k|^{\frac{1}{2}}} \exp[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top S_k^{-1}(\mathbf{x} - \boldsymbol{\mu})]$$

And we know that the Mixture-of-Gaussian distribution can be written as:

$$PM(x) = \sum_{i=1}^{k} \alpha_i * p(x|\mu, S_k)$$

Since the sum of coefficients should be 1 (they represents the probability of chosen the ith component) and the coefficient should larger than 0, we can rewritten the two equation above to get the response:

$$r_{ik} = \pi_k |S_k|^{-1/2} \exp[-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_k)^\top S_k^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)]$$

where $\pi_k \geq 0$, $\sum_{k=1}^{K} \pi_k = 1$. Then, we do the normalization to r_{ik}:

$$r_{ik} = \frac{r_{ik}}{\sum_{j=1}^{K} r_{jk}}$$

To solve $PM(x)$, we can use maximum likelihood estimation:

$$\begin{aligned} LL(D) &= \ln(\prod_{i=1}^{n} PM(x_i)) \\ &= \sum_{i=1}^{n} \ln(\sum_{k=1}^{k} \pi_k * p(x_i|\mu_k, S_k) \end{aligned}$$

To maximize, we will have $\frac{\partial LL(D)}{\partial \mu_k} = 0$:

$$\sum_{i=1}^{n} \frac{\pi_k * p(x_i|\mu_k, S_k)}{\sum_{l=1}^{k} \pi_l * p(x_i|\mu_l, S_l)}(x_i - u_i) = 0$$

Rewrite the equation:

$$\sum_{i=1}^{n} r_{ik}(x_i - \mu_k) = 0$$

$$\begin{aligned}\mu_k &= \frac{\sum_{i=1}^{n} r_{ik} x_i}{\sum_{i=1}^{n} r_{ik}} \\ &= \frac{\sum_{i=1}^{n} r_{ik} x_i}{r_{.k}}\end{aligned}$$

Since we also need to satisfy $\pi_k \geq 0$, $\sum_{i=1}^{k} \pi_k = 1$. Using the Lagrange:

$$\frac{\partial LL(D)}{\partial \pi_k} + \frac{\partial \lambda(\sum_{i=1}^{k} \pi_k - 1)}{\partial \pi_k} = 0$$

$$\sum_{i=1}^{n} \frac{p(x_i|\mu_k, S_k)}{\sum_{l=1}^{k} \pi_l * p(x_i|\mu_l, S_l)} + \lambda = 0$$

We multiply pi_k on both side:

$$\sum_{i=1}^{n} r_{ik} - n\pi_k = 0$$

$$\pi_k = \frac{1}{n} \sum_{i=1}^{n} r_{ik} = \frac{r_{.k}}{n}$$

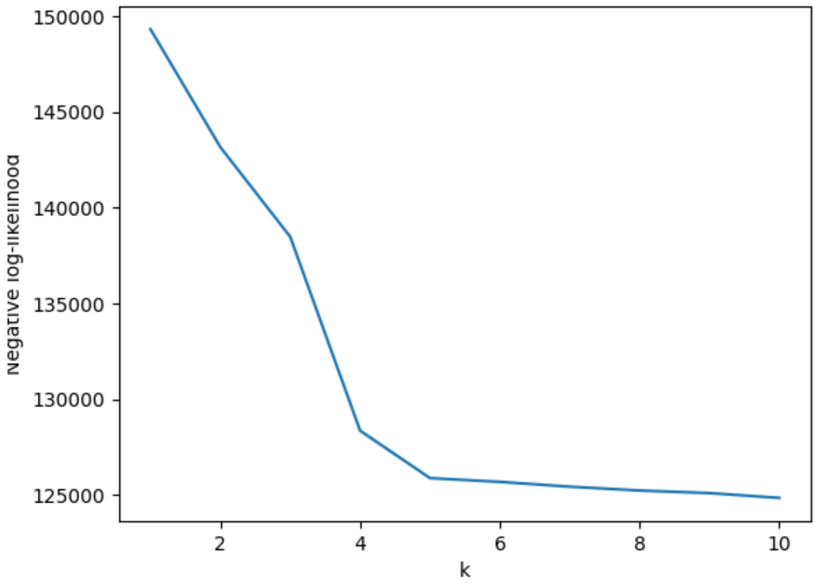
Finally, we get S_k (from how we compute the covarince matrix):

$$S_k = \frac{\sum_{i=1}^{n} r_{ik}(x_i - \mu_k)(x_i - \mu_k)^\top}{\sum_{i=1}^{n} r_{ik}}$$

Simplify this equation, we can get:

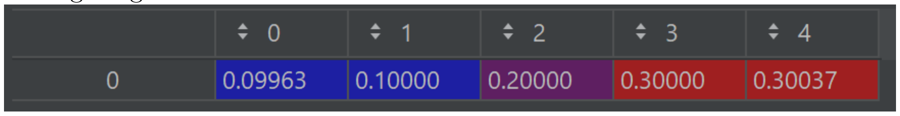
$$S_k = \frac{\sum_{i=1}^{n} r_{ik} x_{ij}^2}{\sum_{i=1}^{n} r_{ik}} - \mu_j^2$$

Space complexity: $n * k + k * d * d + k * d$
Which came from responsibility r_{ik}, S_k and mean, respectively.
Time complexity: $O(2 * n_{iter} * k)$
My code uses dynamic programming to record all the responsibility in compute_log_likelihood(), which use a for loop in range of k. I call this function in estep() to get the log_likelihood. And I call estep() in fit() with in a for loop of n_iter, so $O(n_{iter} * k)$ for now. I also call mstep() in the for loop, which consists of a for loop of k, where I update S_k. Thus, the time complexity is $O(2 * n_{iter} * k)$.



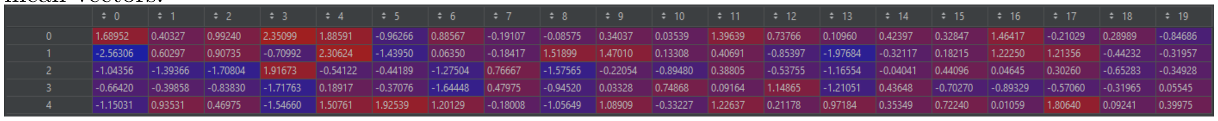
From the graph, my choice of k is 5. Before k = 5, there is a dramatically drop in negative log likelihood. After that, negative log likelihood just gradually drops. Thus, k = 5 is a turning point and we kind of want this in GMM, a balance of loss and the number of clusters exist.

Parameters:(random seed = 24)
mixing weights:



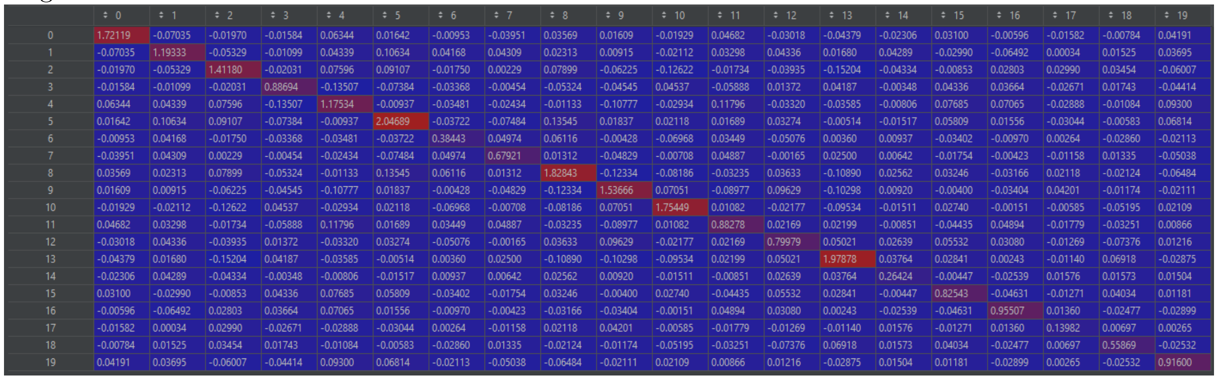
	0	1	2	3	4
0	0.09963	0.10000	0.20000	0.30000	0.30037

mean vectors:

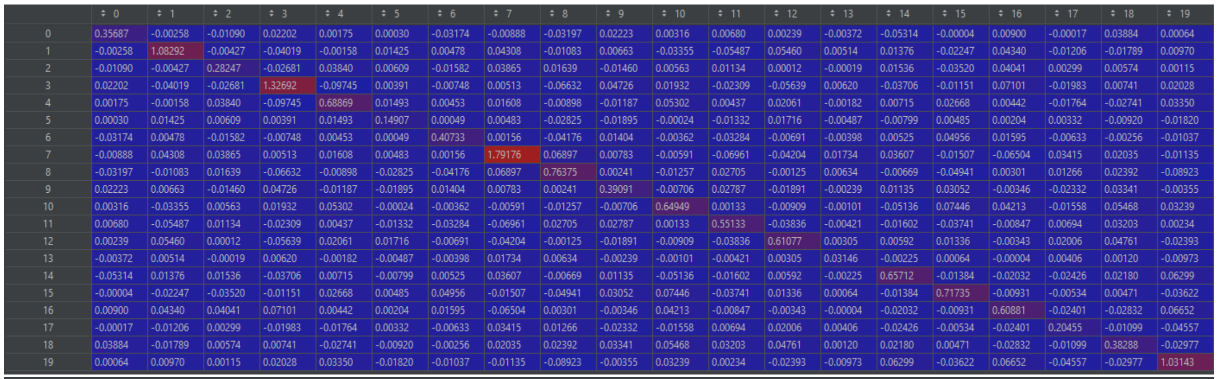


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	1.68952	0.40327	0.99240	2.35099	1.88591	-0.96266	0.88567	-0.19107	-0.08575	0.34037	0.03539	1.39639	0.73766	0.10960	0.42397	0.32847	1.46417	-0.21029	0.28989	-0.84686
1	-2.56306	0.60297	0.90735	-0.70992	2.30624	-1.43950	0.06350	-0.18417	1.51899	1.47010	0.13308	0.40691	-0.85397	-1.97684	-0.32117	0.18215	1.22250	1.21356	-0.44232	-0.31957
2	-1.04356	-1.39366	-1.70804	1.91673	-0.54122	-0.44189	-1.27504	0.76667	-1.57565	-0.22054	-0.89480	0.38805	-0.53755	-1.16554	-0.04041	0.44096	0.04645	0.30260	-0.65283	-0.34928
3	-0.66420	-0.39858	-0.83830	-1.71763	0.18917	-0.37076	-1.64448	0.47975	-0.94520	0.03328	0.74868	0.09164	1.14865	-1.21051	0.43648	-0.70270	-0.89329	-0.57060	-0.31965	0.05545
4	-1.15031	0.93531	0.46975	-1.54660	1.50761	1.92539	1.20129	-0.18008	-1.05649	1.08909	-0.33227	1.22637	0.21178	0.97184	0.35349	0.72240	0.01059	1.80640	0.09241	0.39975

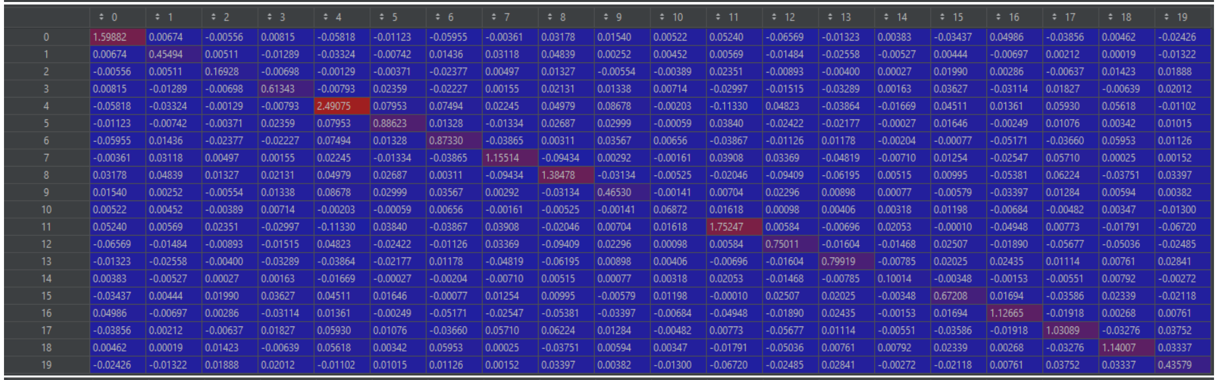
diagnoals of the covariance matrices:



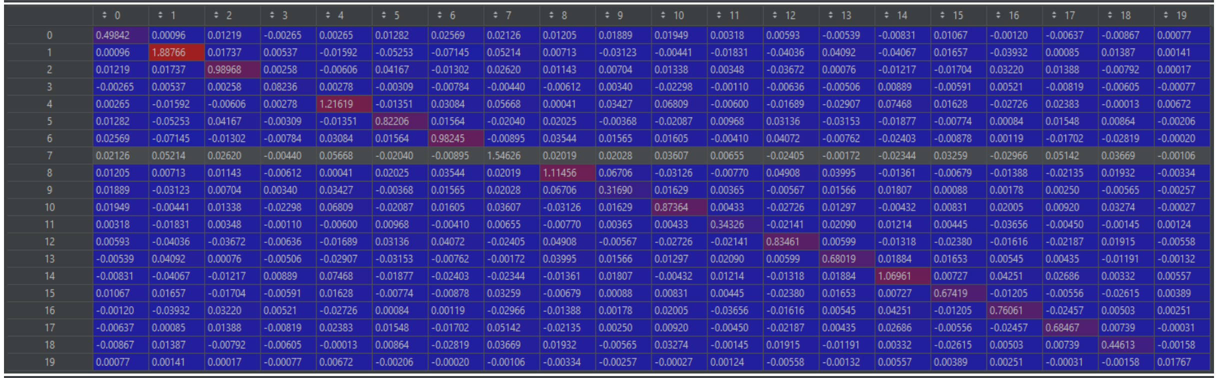
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	1.72119	-0.07035	-0.01970	-0.01584	0.06344	0.01642	-0.00953	-0.03951	0.03569	0.01609	-0.01929	0.04682	-0.03018	-0.04379	-0.02306	0.03100	-0.00596	-0.01582	-0.00784	0.04191
1	-0.07035	1.19333	-0.05329	-0.01099	0.04339	0.10634	0.04168	0.04309	0.02313	0.00915	-0.02112	0.03298	0.04336	0.01680	0.04289	-0.02990	-0.06492	0.00034	0.01525	0.03695
2	-0.01970	-0.05329	1.41180	-0.02031	0.07596	0.09107	-0.01750	0.00229	0.07899	-0.06225	-0.12622	-0.01734	-0.03935	-0.15204	-0.04334	-0.00853	0.02803	0.02990	0.03454	-0.06007
3	-0.01584	-0.01099	-0.02031	0.88694	-0.13507	-0.07384	-0.03368	-0.00454	-0.05324	-0.04545	0.04537	-0.05888	0.01372	0.04187	-0.00348	0.04336	0.03664	-0.02671	0.01743	-0.04414
4	0.06344	0.04339	0.07596	-0.13507	1.17534	-0.00937	-0.03481	-0.02434	-0.01133	-0.10777	-0.02934	0.11796	-0.03320	-0.03585	-0.00806	0.07685	0.07065	-0.02888	-0.01084	0.09300
5	0.01642	0.10634	0.09107	-0.07384	-0.00937	2.04689	-0.03722	-0.07484	0.13545	0.01837	0.02118	0.01689	0.03274	-0.00514	-0.01517	0.05809	0.01556	-0.03044	-0.00583	0.06814
6	-0.00953	0.04168	-0.01750	-0.03368	-0.03481	-0.03722	0.38443	0.04974	0.06116	-0.00428	-0.06968	0.03449	-0.05076	0.00360	0.00937	-0.03402	-0.00970	0.00264	-0.02860	-0.02113
7	-0.03951	0.04309	0.00229	-0.00454	-0.02434	-0.07484	0.04974	0.67921	0.01312	-0.04829	-0.00708	0.04887	-0.00165	0.02500	0.00642	-0.01754	-0.00423	-0.01158	0.01335	-0.05038
8	0.03569	0.02313	0.07899	-0.05324	-0.01133	0.13545	0.06116	0.01312	1.82843	-0.12334	-0.08186	-0.03235	0.03633	-0.10890	0.02562	0.03246	-0.03166	0.02118	-0.02124	-0.06484
9	0.01609	0.00915	-0.06225	-0.04545	-0.10777	0.01837	-0.00428	-0.04829	-0.12334	1.53666	0.07051	-0.08977	0.09629	-0.10298	0.00920	-0.00400	-0.03404	0.04201	-0.01174	-0.02111
10	-0.01929	-0.02112	-0.12622	0.04537	-0.02934	0.02118	-0.06968	-0.00708	-0.08186	0.07051	1.75449	0.01082	-0.02177	-0.09534	-0.01511	0.02740	-0.00151	-0.00585	-0.05195	0.02109
11	0.04682	0.03298	-0.01734	-0.05888	0.11796	0.01689	0.03449	0.04887	-0.03235	-0.08977	0.01082	0.88278	0.02169	0.02199	-0.00851	-0.04435	0.04894	-0.01779	-0.03251	0.00866
12	-0.03018	0.04336	-0.03935	0.01372	-0.03320	0.03274	-0.05076	-0.00165	0.03633	0.09629	-0.02177	0.02169	0.79979	0.05021	0.02639	0.05532	0.03080	-0.01269	-0.07376	0.01216
13	-0.04379	0.01680	-0.15204	0.04187	-0.03585	-0.00514	0.00360	0.02500	-0.10890	-0.10298	-0.09534	0.02199	0.05021	1.97878	0.03764	0.02841	0.00243	-0.01140	0.06918	-0.02875
14	-0.02306	0.04289	-0.04334	-0.00348	-0.00806	-0.01517	0.00937	0.00642	0.02562	0.00920	-0.01511	-0.00851	0.02639	0.03764	0.26424	-0.00447	-0.02539	0.01576	0.01573	0.01504
15	0.03100	-0.02990	-0.00853	0.04336	0.07685	0.05809	-0.03402	-0.01754	0.03246	-0.00400	0.02740	-0.04435	0.05532	0.02841	-0.00447	0.82543	-0.04631	-0.01271	0.04034	0.01181
16	-0.00596	-0.06492	0.02803	0.03664	0.07065	0.01556	-0.00970	-0.00423	-0.03166	-0.03404	-0.00151	0.04894	0.03080	0.00243	-0.02539	-0.04631	0.95507	0.01360	-0.02477	-0.02899
17	-0.01582	0.00034	0.02990	-0.02671	-0.02888	-0.03044	0.00264	-0.01158	0.02118	0.04201	-0.00585	-0.01779	-0.01269	-0.01140	0.01576	-0.01271	0.01360	0.13982	0.00697	0.00265
18	-0.00784	0.01525	0.03454	0.01743	-0.01084	-0.00583	-0.02860	0.01335	-0.02124	-0.01174	-0.05195	-0.03251	-0.07376	0.06918	0.01573	0.04034	-0.02477	0.00697	0.55869	-0.02532
19	0.04191	0.03695	-0.06007	-0.04414	0.09300	0.06814	-0.02113	-0.05038	-0.06484	-0.02111	0.02109	0.00866	0.01216	-0.02875	0.01504	0.01181	-0.02899	0.00265	-0.02532	0.91600



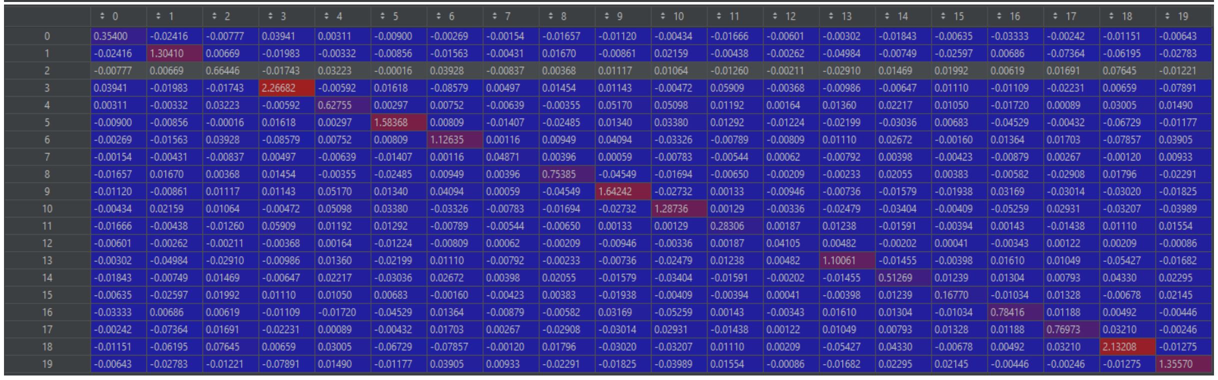
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0.35687	-0.00258	-0.01090	0.02202	0.00175	0.00030	-0.03174	-0.00888	-0.03197	0.02223	0.00316	0.00680	0.00239	-0.00372	-0.05314	-0.00004	0.00900	-0.00017	0.03884	0.00064
1	-0.00258	1.08292	-0.00427	-0.04019	-0.00158	0.01425	0.00478	0.04308	-0.01083	0.00663	-0.03355	-0.05487	0.05460	0.00514	0.01376	-0.02247	0.04340	-0.01206	-0.01789	0.00970
2	-0.01090	-0.00427	0.28247	-0.02681	0.03840	0.00609	-0.01582	0.03865	0.01639	-0.01460	0.00563	0.01134	0.00012	-0.00019	0.01536	-0.03520	0.04041	0.00299	0.00574	0.00115
3	0.02202	-0.04019	-0.02681	1.32692	-0.09745	0.00391	-0.00748	0.00513	-0.06632	0.04726	0.01932	-0.02309	-0.05639	0.00620	-0.03706	-0.01151	0.07101	-0.01983	0.00741	0.02028
4	0.00175	-0.00158	0.03840	-0.09745	0.68869	0.01493	0.00453	0.01608	-0.00898	-0.01187	0.05302	0.00437	0.02061	-0.00182	0.00715	0.02668	0.00442	-0.01764	-0.02741	0.03350
5	0.00030	0.01425	0.00609	0.00391	0.01493	0.14907	0.00049	0.00483	-0.02825	-0.01895	-0.00024	-0.01332	0.01716	-0.00487	-0.00799	0.00485	0.00204	0.00332	-0.00920	-0.01820
6	-0.03174	0.00478	-0.01582	-0.00748	0.00453	0.00049	0.40733	0.00156	-0.04176	0.01404	-0.00362	-0.03284	-0.00691	-0.00398	0.00525	0.04956	0.01595	-0.00633	-0.00256	-0.01037
7	-0.00888	0.04308	0.03865	0.00513	0.01608	0.00483	0.00156	1.79176	0.06897	0.00783	-0.00591	-0.06961	-0.04204	0.01734	0.03607	-0.01507	-0.06504	0.03415	0.02035	-0.01135
8	-0.03197	-0.01083	0.01639	-0.06632	-0.00898	-0.02825	-0.04176	0.06897	0.76375	0.00241	-0.01257	0.02705	-0.00125	0.00634	-0.00669	-0.04941	0.00301	0.01266	0.02392	-0.08923
9	0.02223	0.00663	-0.01460	0.04726	-0.01187	-0.01895	0.01404	0.00783	0.00241	0.39091	-0.00706	0.02787	-0.01891	-0.00239	0.01135	0.03052	-0.00346	-0.02332	0.03341	-0.00355
10	0.00316	-0.03355	0.00563	0.01932	0.05302	-0.00024	-0.00362	-0.00591	-0.01257	-0.00706	0.64949	0.00133	-0.00909	-0.00101	-0.05136	0.07446	0.04213	-0.01558	0.05468	0.03239
11	0.00680	-0.05487	0.01134	-0.02309	0.00437	-0.01332	-0.03284	-0.06961	0.02705	0.02787	0.00133	0.55133	-0.03836	-0.00421	-0.01602	-0.03741	-0.00847	0.00694	0.03203	0.00234
12	0.00239	0.05460	0.00012	-0.05639	0.02061	0.01716	-0.00691	-0.04204	-0.00125	-0.01891	-0.00909	-0.03836	0.61077	0.00305	0.00592	0.01336	-0.00343	0.02006	0.04761	-0.02393
13	-0.00372	0.00514	-0.00019	0.00620	-0.00182	-0.00487	-0.00398	0.01734	0.00634	-0.00239	-0.00101	-0.00421	0.00305	0.03146	-0.00225	0.00064	-0.00004	0.00406	0.00120	-0.00973
14	-0.05314	0.01376	0.01536	-0.03706	0.00715	-0.00799	0.00525	0.03607	-0.00669	0.01135	-0.05136	-0.01602	0.00592	-0.00225	0.65712	-0.01384	-0.02032	-0.02426	0.02180	0.06299
15	-0.00004	-0.02247	-0.03520	-0.01151	0.02668	0.00485	0.04956	-0.01507	-0.04941	0.03052	0.07446	-0.03741	0.01336	0.00064	-0.01384	0.71735	-0.00931	-0.00534	0.00471	-0.03622
16	0.00900	0.04340	0.04041	0.07101	0.00442	0.00204	0.01595	-0.06504	0.00301	-0.00346	0.04213	-0.00847	-0.00343	-0.00004	-0.02032	-0.00931	0.60881	-0.02401	-0.02832	0.06652
17	-0.00017	-0.01206	0.00299	-0.01983	-0.01764	0.00332	-0.00633	0.03415	0.01266	-0.02332	-0.01558	0.00694	0.02006	0.00406	-0.02426	-0.00534	-0.02401	0.20455	-0.01099	-0.04557
18	0.03884	-0.01789	0.00574	0.00741	-0.02741	-0.00920	-0.00256	0.02035	0.02392	0.03341	0.05468	0.03203	0.04761	0.00120	0.02180	0.00471	-0.02832	-0.01099	0.38288	-0.02977
19	0.00064	0.00970	0.00115	0.02028	0.03350	-0.01820	-0.01037	-0.01135	-0.08923	-0.00355	0.03239	0.00234	-0.02393	-0.00973	0.06299	-0.03622	0.06652	-0.04557	-0.02977	1.03143



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	1.59882	0.00674	-0.00556	0.00815	-0.05818	-0.01123	-0.05955	-0.00361	0.03178	0.01540	0.00522	0.05240	-0.06569	-0.01323	0.00383	-0.03437	0.04986	-0.03856	0.00462	-0.02426
1	0.00674	0.45494	0.00511	-0.01289	-0.03324	-0.00742	0.01436	0.03118	0.04839	0.00252	0.00452	0.00569	-0.01484	-0.02558	-0.00527	0.00444	-0.00697	0.00212	0.00019	-0.01322
2	-0.00556	0.00511	0.16928	-0.00698	-0.00129	-0.00371	-0.02377	0.00497	0.01327	-0.00554	-0.00389	0.02351	-0.00893	-0.00400	0.00027	0.01990	0.00286	-0.00637	0.01423	0.01888
3	0.00815	-0.01289	-0.00698	0.61343	-0.00793	0.02359	-0.02227	0.00155	0.02131	0.01338	0.00714	-0.02997	-0.01515	-0.03289	0.00163	0.03627	-0.03114	0.01827	-0.00639	0.02012
4	-0.05818	-0.03324	-0.00129	-0.00793	2.49075	0.07953	0.07494	0.02245	0.04979	0.08678	-0.00203	-0.11330	0.04823	-0.03864	-0.01669	0.04511	0.01361	0.05930	0.05618	-0.01102
5	-0.01123	-0.00742	-0.00371	0.02359	0.07953	0.88623	0.01328	-0.01334	0.02687	0.02999	-0.00059	0.03840	-0.02422	-0.02177	-0.00027	0.01646	-0.00249	0.01076	0.00342	0.01015
6	-0.05955	0.01436	-0.02377	-0.02227	0.07494	0.01328	0.87330	-0.03865	0.00311	0.03567	0.00656	-0.03867	-0.01126	0.01178	-0.00204	-0.00077	-0.05171	-0.03660	0.05953	0.01126
7	-0.00361	0.03118	0.00497	0.00155	0.02245	-0.01334	-0.03865	1.15514	-0.09434	0.00292	-0.00161	0.03908	0.03369	-0.04819	-0.00710	0.01254	-0.02547	0.05710	0.00025	0.00152
8	0.03178	0.04839	0.01327	0.02131	0.04979	0.02687	0.00311	-0.09434	1.38478	-0.03134	-0.00525	-0.02046	-0.09409	-0.06195	0.00515	0.00995	-0.05381	0.06224	-0.03751	0.03397
9	0.01540	0.00252	-0.00554	0.01338	0.08678	0.02999	0.03567	0.00292	-0.03134	0.46530	-0.00141	0.00704	0.02296	0.00898	0.00077	-0.00579	-0.03397	0.01284	0.00594	0.00382
10	0.00522	0.00452	-0.00389	0.00714	-0.00203	-0.00059	0.00656	-0.00161	-0.00525	-0.00141	0.06872	0.01618	0.00098	0.00406	0.00318	0.01198	-0.00684	-0.00482	0.00347	-0.01300
11	0.05240	0.00569	0.02351	-0.02997	-0.11330	0.03840	-0.03867	0.03908	-0.02046	0.00704	0.01618	1.75247	0.00584	-0.00696	0.02053	-0.00010	-0.04948	0.00773	-0.01791	-0.06720
12	-0.06569	-0.01484	-0.00893	-0.01515	0.04823	-0.02422	-0.01126	0.03369	-0.09409	0.02296	0.00098	0.00584	0.75011	-0.01604	-0.01468	0.02507	-0.01890	-0.05677	-0.05036	-0.02485
13	-0.01323	-0.02558	-0.00400	-0.03289	-0.03864	-0.02177	0.01178	-0.04819	-0.06195	0.00898	0.00406	-0.00696	-0.01604	0.79919	-0.00785	0.02025	0.02435	0.01114	0.00761	0.02841
14	0.00383	-0.00527	0.00027	0.00163	-0.01669	-0.00027	-0.00204	-0.00710	0.00515	0.00077	0.00318	0.02053	-0.01468	-0.00785	0.10014	-0.00348	-0.00153	-0.00551	0.00792	-0.00272
15	-0.03437	0.00444	0.01990	0.03627	0.04511	0.01646	-0.00077	0.01254	0.00995	-0.00579	0.01198	-0.00010	0.02507	0.02025	-0.00348	0.67208	0.01694	-0.03586	0.02339	-0.02118
16	0.04986	-0.00697	0.00286	-0.03114	0.01361	-0.00249	-0.05171	-0.02547	-0.05381	-0.03397	-0.00684	-0.04948	-0.01890	0.02435	-0.00153	0.01694	1.12665	-0.01918	0.00268	0.00761
17	-0.03856	0.00212	-0.00637	0.01827	0.05930	0.01076	-0.03660	0.05710	0.06224	0.01284	-0.00482	0.00773	-0.05677	0.01114	-0.00551	-0.03586	-0.01918	1.03089	-0.03276	0.03752
18	0.00462	0.00019	0.01423	-0.00639	0.05618	0.00342	0.05953	0.00025	-0.03751	0.00594	0.00347	-0.01791	-0.05036	0.00761	0.00792	0.02339	0.00268	-0.03276	1.14007	0.03337
19	-0.02426	-0.01322	0.01888	0.02012	-0.01102	0.01015	0.01126	0.00152	0.03397	0.00382	-0.01300	-0.06720	-0.02485	0.02841	-0.00272	-0.02118	0.00761	0.03752	0.03337	0.43579



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0.49842	0.00096	0.01219	-0.00265	0.00265	0.01282	0.02569	0.02126	0.01205	0.01889	0.01949	0.00318	0.00593	-0.00539	-0.00831	0.01067	-0.00120	-0.00637	-0.00867	0.00077
1	0.00096	1.88766	0.01737	0.00537	-0.01592	-0.05253	-0.07145	0.05214	0.00713	-0.03123	-0.00441	-0.01831	-0.04036	0.04092	-0.04067	0.01657	-0.03932	0.00085	0.01387	0.00141
2	0.01219	0.01737	0.98968	0.00258	-0.00606	0.04167	-0.01302	0.02620	0.01143	0.00704	0.01338	0.00348	-0.03672	0.00076	-0.01217	-0.01704	0.03220	0.01388	-0.00792	0.00017
3	-0.00265	0.00537	0.00258	0.08236	0.00278	-0.00309	-0.00784	-0.00440	-0.00612	0.00340	-0.02298	-0.00110	-0.00636	-0.00506	0.00889	-0.00591	0.00521	-0.00819	-0.00605	-0.00077
4	0.00265	-0.01592	-0.00606	0.00278	1.21619	-0.01351	0.03084	0.05668	0.00041	0.03427	0.06809	-0.00600	-0.01689	-0.02907	0.07468	0.01628	-0.02726	0.02383	-0.00013	0.00672
5	0.01282	-0.05253	0.04167	-0.00309	-0.01351	0.82206	0.01564	-0.02040	0.02025	-0.00368	-0.02087	0.00968	0.03136	-0.03153	-0.01877	-0.00774	0.00084	0.01548	0.00864	-0.00206
6	0.02569	-0.07145	-0.01302	-0.00784	0.03084	0.01564	0.98245	-0.00895	0.03544	0.01565	0.01605	-0.00410	0.04072	-0.00762	-0.02403	-0.00878	0.00119	-0.01702	-0.02819	-0.00020
7	0.02126	0.05214	0.02620	-0.00440	0.05668	-0.02040	-0.00895	1.54626	0.02019	0.02028	0.03607	0.00655	-0.02405	-0.00172	-0.02344	0.03259	-0.02966	0.05142	0.03669	-0.00106
8	0.01205	0.00713	0.01143	-0.00612	0.00041	0.02025	0.03544	0.02019	1.11456	0.06706	-0.03126	-0.00770	0.04908	0.03995	-0.01361	-0.00679	-0.01388	-0.02135	0.01932	-0.00334
9	0.01889	-0.03123	0.00704	0.00340	0.03427	-0.00368	0.01565	0.02028	0.06706	0.31690	0.01629	0.00365	-0.00567	0.01566	0.01807	0.00088	0.00178	0.00250	-0.00565	-0.00257
10	0.01949	-0.00441	0.01338	-0.02298	0.06809	-0.02087	0.01605	0.03607	-0.03126	0.01629	0.87364	0.00433	-0.02726	0.01297	-0.00432	0.00831	0.02005	0.00920	0.03274	-0.00027
11	0.00318	-0.01831	0.00348	-0.00110	-0.00600	0.00968	-0.00410	0.00655	-0.00770	0.00365	0.00433	0.34326	-0.02141	0.02090	0.01214	0.00445	-0.03656	-0.00450	-0.00145	0.00124
12	0.00593	-0.04036	-0.03672	-0.00636	-0.01689	0.03136	0.04072	-0.02405	0.04908	-0.00567	-0.02726	-0.02141	0.83461	0.00599	-0.01318	-0.02380	-0.01616	-0.02187	0.01915	-0.00558
13	-0.00539	0.04092	0.00076	-0.00506	-0.02907	-0.03153	-0.00762	-0.00172	0.03995	0.01566	0.01297	0.02090	0.00599	0.68019	0.01884	0.01653	0.00545	0.00435	-0.01191	-0.00132
14	-0.00831	-0.04067	-0.01217	0.00889	0.07468	-0.01877	-0.02403	-0.02344	-0.01361	0.01807	-0.00432	0.01214	-0.01318	0.01884	1.06961	0.00727	0.04251	0.02686	0.00332	0.00557
15	0.01067	0.01657	-0.01704	-0.00591	0.01628	-0.00774	-0.00878	0.03259	-0.00679	0.00088	0.00831	0.00445	-0.02380	0.01653	0.00727	0.67419	-0.01205	-0.00556	-0.02615	0.00389
16	-0.00120	-0.03932	0.03220	0.00521	-0.02726	0.00084	0.00119	-0.02966	-0.01388	0.00178	0.02005	-0.03656	-0.01616	0.00545	0.04251	-0.01205	0.76061	-0.02457	0.00503	0.00251
17	-0.00637	0.00085	0.01388	-0.00819	0.02383	0.01548	-0.01702	0.05142	-0.02135	0.00250	0.00920	-0.00450	-0.02187	0.00435	0.02686	-0.00556	-0.02457	0.68467	0.00739	-0.00031
18	-0.00867	0.01387	-0.00792	-0.00605	-0.00013	0.00864	-0.02819	0.03669	0.01932	-0.00565	0.03274	-0.00145	0.01915	-0.01191	0.00332	-0.02615	0.00503	0.00739	0.44613	-0.00158
19	0.00077	0.00141	0.00017	-0.00077	0.00672	-0.00206	-0.00020	-0.00106	-0.00334	-0.00257	-0.00027	0.00124	-0.00558	-0.00132	0.00557	0.00389	0.00251	-0.00031	-0.00158	0.01767



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0.35400	-0.02416	-0.00777	0.03941	0.00311	-0.00900	-0.00269	-0.00154	-0.01657	-0.01120	-0.00434	-0.01666	-0.00601	-0.00302	-0.01843	-0.00635	-0.03333	-0.00242	-0.01151	-0.00643
1	-0.02416	1.30410	0.00669	-0.01983	-0.00332	-0.00856	-0.01563	-0.00431	0.01670	-0.00861	0.02159	-0.00438	-0.00262	-0.04984	-0.00749	-0.02597	0.00686	-0.07364	-0.06195	-0.02783
2	-0.00777	0.00669	0.66446	-0.01743	0.03223	-0.00016	0.03928	-0.00837	0.00368	0.01117	0.01064	-0.01260	-0.00211	-0.02910	0.01469	0.01992	0.00619	0.01691	0.07645	-0.01221
3	0.03941	-0.01983	-0.01743	2.26682	-0.00592	0.01618	-0.08579	0.00497	0.01454	0.01143	-0.00472	0.05909	-0.00368	-0.00986	-0.00647	0.01110	-0.01109	-0.02231	0.00659	-0.07891
4	0.00311	-0.00332	0.03223	-0.00592	0.62755	0.00297	0.00752	-0.00639	-0.00355	0.05170	0.05098	0.01192	0.00164	0.01360	0.02217	0.01050	-0.01720	0.00089	0.03005	0.01490
5	-0.00900	-0.00856	-0.00016	0.01618	0.00297	1.58368	0.00809	-0.01407	-0.02485	0.01340	0.03380	0.01292	-0.01224	-0.02199	-0.03036	0.00683	-0.04529	-0.00432	-0.06729	-0.01177
6	-0.00269	-0.01563	0.03928	-0.08579	0.00752	0.00809	1.12635	0.00116	0.00949	0.04094	-0.03326	-0.00789	-0.00809	0.01110	0.02672	-0.00160	0.01364	0.01703	-0.07857	0.03905
7	-0.00154	-0.00431	-0.00837	0.00497	-0.00639	-0.01407	0.00116	0.04871	0.00396	0.00059	-0.00783	-0.00544	0.00062	-0.00792	0.00398	-0.00423	-0.00879	0.00267	-0.00120	0.00933
8	-0.01657	0.01670	0.00368	0.01454	-0.00355	-0.02485	0.00949	0.00396	0.75385	-0.04549	-0.01694	-0.00650	-0.00209	-0.00233	0.02055	0.00383	-0.00582	-0.02908	0.01796	-0.02291
9	-0.01120	-0.00861	0.01117	0.01143	0.05170	0.01340	0.04094	0.00059	-0.04549	1.64242	-0.02732	0.00133	-0.00946	-0.00736	-0.01579	-0.01938	0.03169	-0.03014	-0.03020	-0.01825
10	-0.00434	0.02159	0.01064	-0.00472	0.05098	0.03380	-0.03326	-0.00783	-0.01694	-0.02732	1.28736	0.00129	-0.00336	-0.02479	-0.03404	-0.00409	-0.05259	0.02931	-0.03207	-0.03989
11	-0.01666	-0.00438	-0.01260	0.05909	0.01192	0.01292	-0.00789	-0.00544	-0.00650	0.00133	0.00129	0.28306	0.00187	0.01238	-0.01591	-0.00394	0.00143	-0.01438	0.01110	0.01554
12	-0.00601	-0.00262	-0.00211	-0.00368	0.00164	-0.01224	-0.00809	0.00062	-0.00209	-0.00946	-0.00336	0.00187	0.04105	0.00482	-0.00202	0.00041	-0.00343	0.00122	0.00209	-0.00086
13	-0.00302	-0.04984	-0.02910	-0.00986	0.01360	-0.02199	0.01110	-0.00792	-0.00233	-0.00736	-0.02479	0.01238	0.00482	1.10061	-0.01455	-0.00398	0.01610	0.01049	-0.05427	-0.01682
14	-0.01843	-0.00749	0.01469	-0.00647	0.02217	-0.03036	0.02672	0.00398	0.02055	-0.01579	-0.03404	-0.01591	-0.00202	-0.01455	0.51269	0.01239	0.01304	0.00793	0.04330	0.02295
15	-0.00635	-0.02597	0.01992	0.01110	0.01050	0.00683	-0.00160	-0.00423	0.00383	-0.01938	-0.00409	-0.00394	0.00041	-0.00398	0.01239	0.16770	-0.01034	0.01328	-0.00678	0.02145
16	-0.03333	0.00686	0.00619	-0.01109	-0.01720	-0.04529	0.01364	-0.00879	-0.00582	0.03169	-0.05259	0.00143	-0.00343	0.01610	0.01304	-0.01034	0.78416	0.01188	0.00492	-0.00446
17	-0.00242	-0.07364	0.01691	-0.02231	0.00089	-0.00432	0.01703	0.00267	-0.02908	-0.03014	0.02931	-0.01438	0.00122	0.01049	0.00793	0.01328	0.01188	0.76973	0.03210	-0.00246
18	-0.01151	-0.06195	0.07645	0.00659	0.03005	-0.06729	-0.07857	-0.00120	0.01796	-0.03020	-0.03207	0.01110	0.00209	-0.05427	0.04330	-0.00678	0.00492	0.03210	2.13208	-0.01275
19	-0.00643	-0.02783	-0.01221	-0.07891	0.01490	-0.01177	0.03905	0.00933	-0.02291	-0.01825	-0.03989	0.01554	-0.00086	-0.01682	0.02295	0.02145	-0.00446	-0.00246	-0.01275	1.35570

b) (5 pts) Next, we apply (the adapted) Algorithm 1 in part a) to the MNIST dataset. For each of the 10 classes (digits), we can use its (only its) training images to estimate its (class-conditional) distribution by fitting a GMM (with say $K = 5$, 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:

$$\hat{y}(\mathbf{x}) = \arg\max_{c=0,\ldots,9} \underbrace{\Pr(Y=c)\cdot p(X=\mathbf{x}|Y=c)}_{\propto \Pr(Y=c|X=\mathbf{x})}, \tag{2}$$

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 rate on the test set as a function of K (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.]

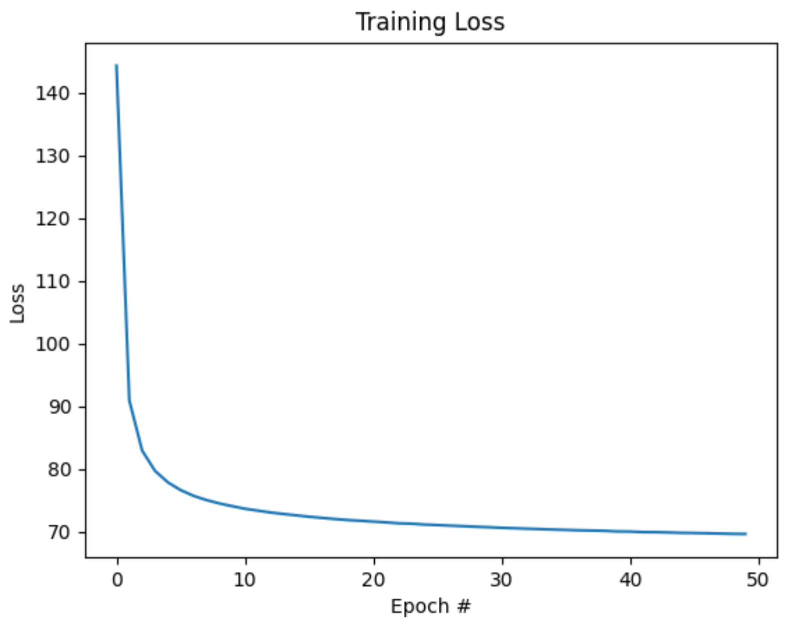
Ans: I have implement PCA of dimension = 20, and the test accuracy is 0.9627 for K=5 (error rate: 0.0373).

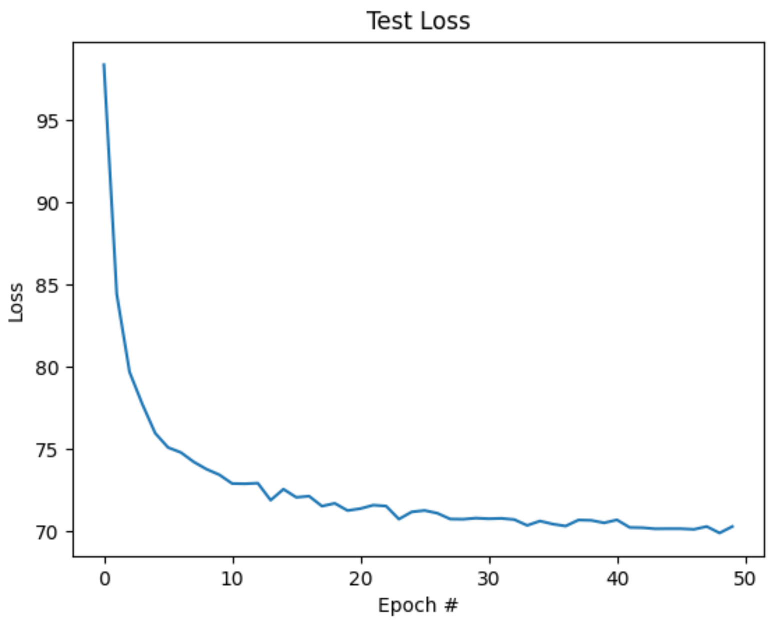
Exercise 2: VAEs and GANs (10 pts)

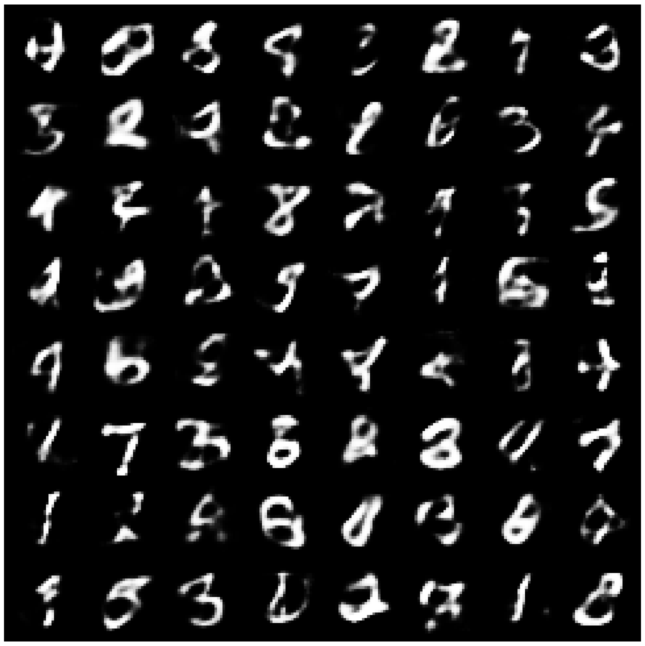
Code provided for this exercise is assuming a PyTorch solution, you may change it as necessary if you prefer to use TensorFlow or JAX. You may find this tutorial for GANs helpful.

a (4.5 pts) Complete the implementation of a VAE in vae.py. In your solution writeup, include the two produced graphs (corresponding to the training and test sets), with the epoch number on x-axis and the loss on the y-axis. Further include the produced samples from every 10th epoch (after epochs 10, 20, 30, 40, and 50).

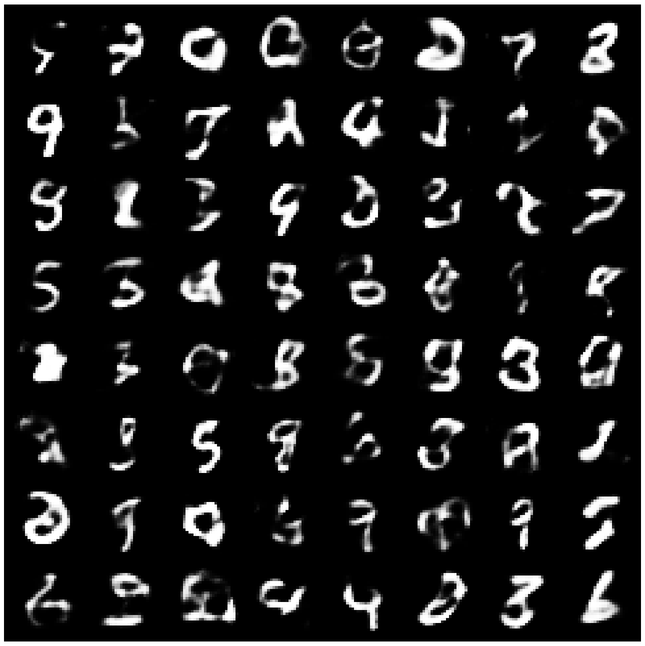
Ans:

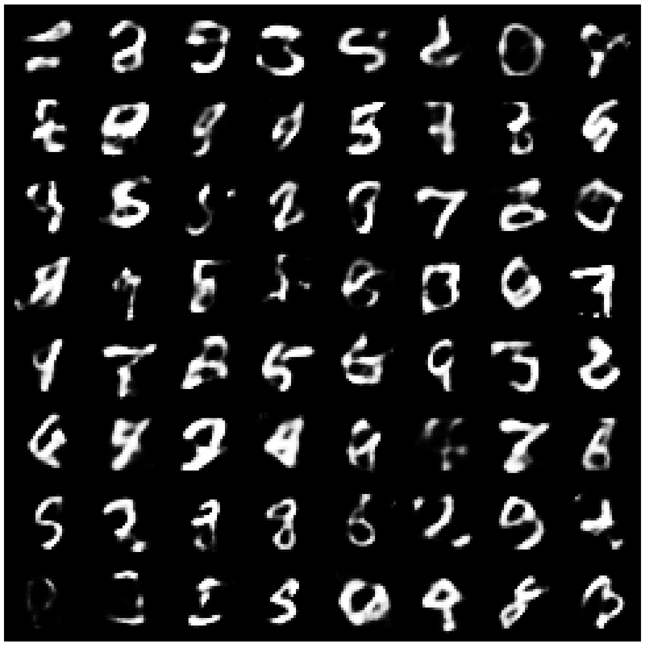


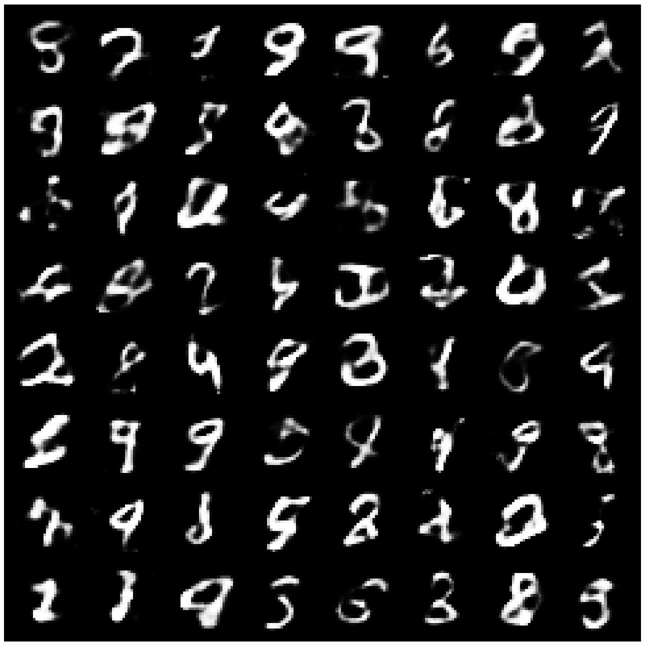




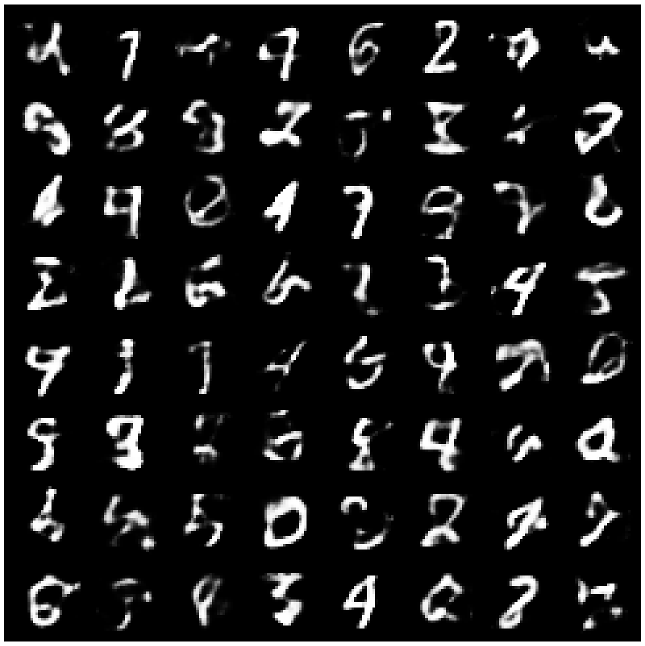
10 epochs


20 epochs


30 epochs



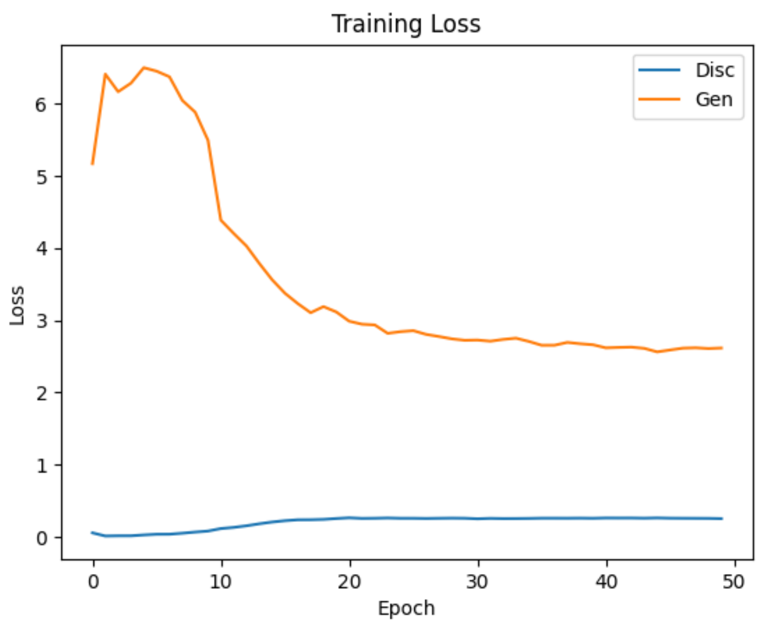
40 epochs

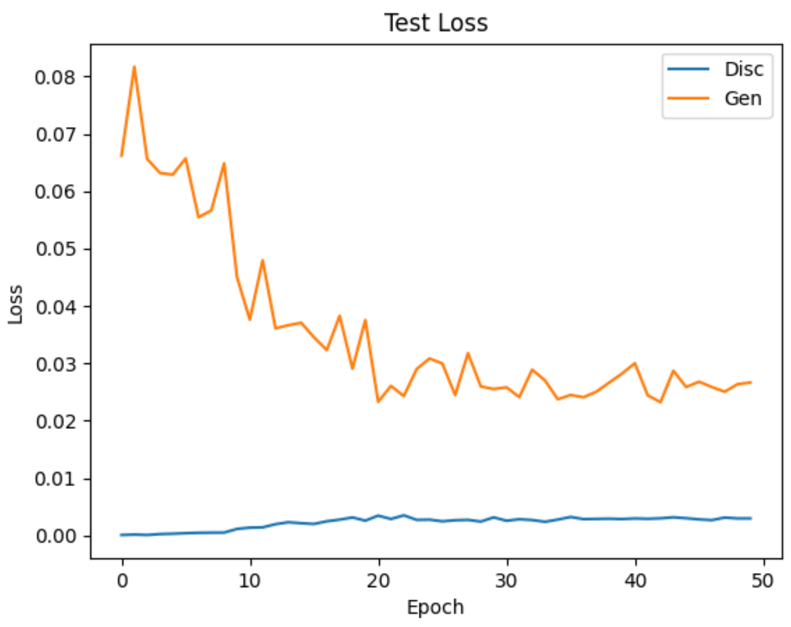


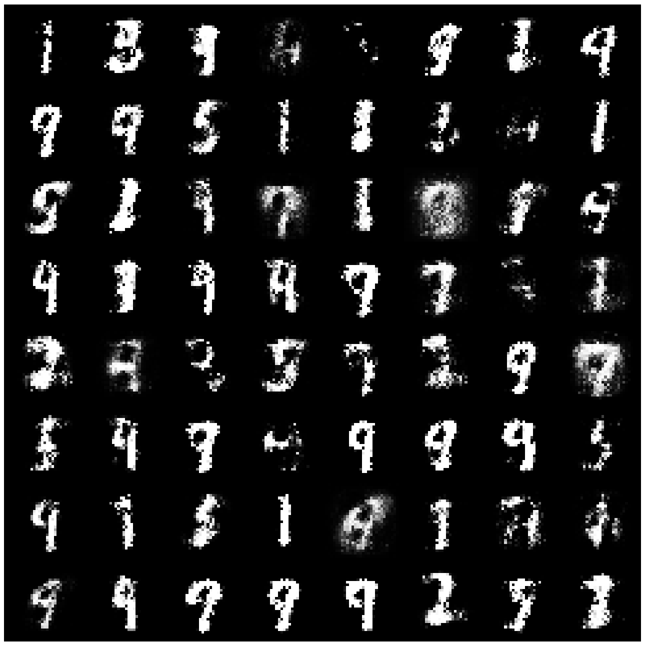
50 epochs

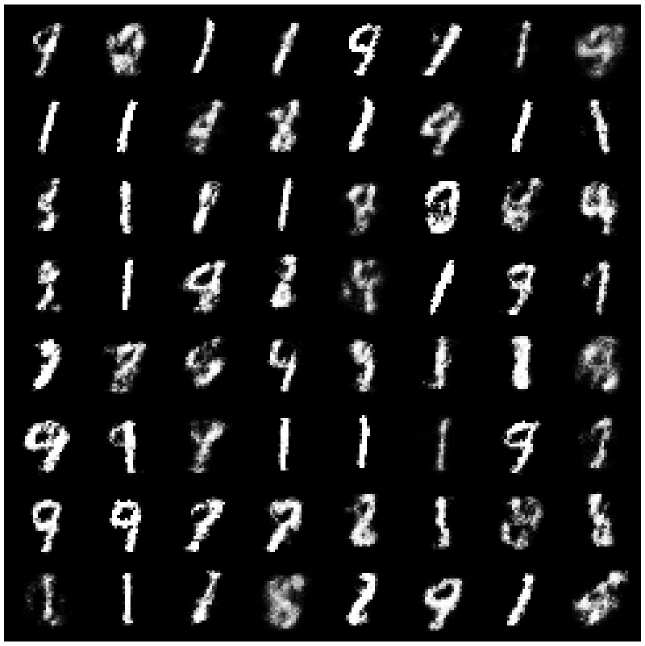
b (4.5 pts) Complete the implementation of a GAN in gan.py. In your solution writeup, include the two produced graphs (corresponding to the training and test sets), with the epoch number on x-axis and the losses (both generator and discriminator) on the y-axis. Further include the produced samples from every 10th epoch (after epochs 10, 20, 30, 40, and 50).

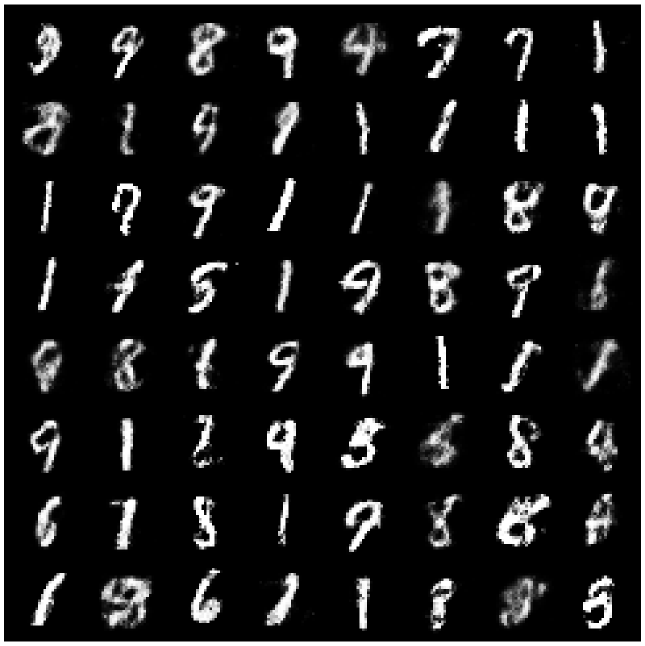
Ans:

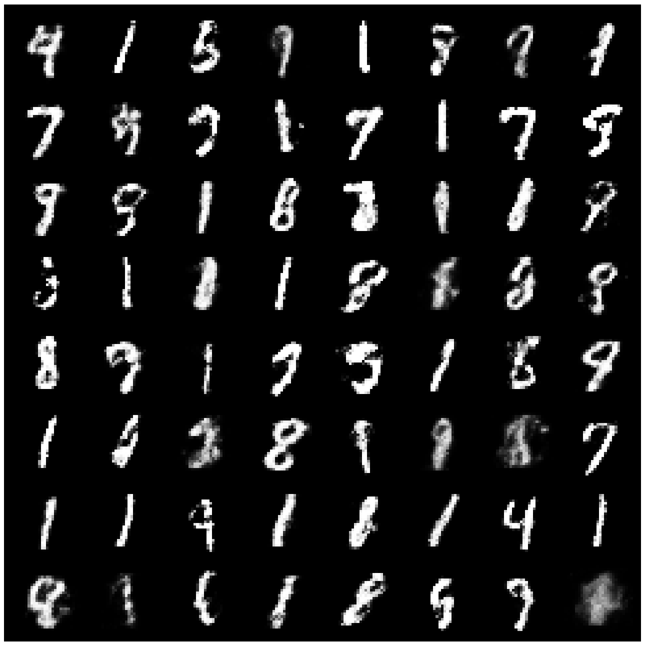

Training Loss
Disc
Gen
Loss
Epoch
0
1
2
3
4
5
6
0
10
20
30
40
50

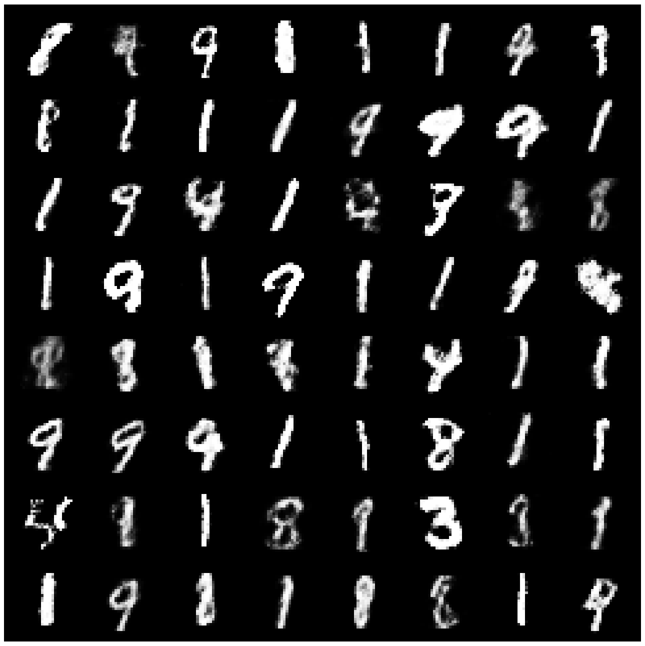

Test Loss
Disc
Gen
Loss
Epoch
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0
10
20
30
40
50


10 epochs


20 epochs


30 epochs


40 epochs


50 epochs

c (1 pt) Describe, compare, and constrast your produced results and experiences with GANs and VAEs in this exercise.

Ans: For VAEs' results, the training loss and test loss decrease rapidly in first few epochs, then gradually decreases (with small fluctuations in test loss) during the rest epochs. Basically, the VAE loss plots show the model is well trained and not suffering from overfitting. While in GAN's loss curves, for training, the generator loss rises at the beginning and starts decreasing at around epoch 10. Again, it rapidly decreases in the next few epochs then gradually decreases. This can be interpreted as that the images generated by generator cannot fool the discriminator at the beginning, as we trained it more, it gradually generate images that can fool the discriminator (which is what we want, faked real images), so the generator loss goes down. For the discriminator loss, it is relatively low at the beginning compared to the generator loss, and only gain very little increase during 50 epochs. This can be interpreted as the discriminator is good at distinguishing the real images from the fake images at first (which should be easy at first). Then our generator keep generated more "realistic" fake images which sometimes fool the discriminator, so the loss goes up but not to very high because our discriminator is still learning and try to be capable doing its job. That's why it is called adversarial learning. The disc and gen loss for test is basically the same trend but with more fluctuations.

Regarding to the images VAE and GAN generated, VAE can generate different numbers, but for many of them, it is kind of hard for a human to tell what the number is. Besides, some of them are even not numbers to me, they are very weird symbols. While for GAN, the numbers it generated are most 1 and 9, as these numbers could be easier to fool the discriminator as I imaged. We can still see 3, 4, 5, 7, 8 as well. There are some blur and meaningless symbol as well, but for the numbers I can recognized, they are more clear compared to VAE's.

For the experience with VAE and GAN, I think VAE is relatively simpler to implement, while for GAN, you have to notice many details, such as how generator loss and discriminator loss is calculated; when calculating discriminator loss, there are two parts: the loss to tell the real is real and the loss to tell fake is fake; when pass the generated data into the discriminator to get the fake loss, we have to add .detach() so that the loss will not back-propagated. We want to do this because we are not training the generator but the discriminator. We have to be more careful to details like those when implementing GAN.