Homework 7: Collaborative Filtering & Variational Autoencoders

UC Irvine CS275P: Machine Learning with Generative Models

Homework due at 11:59pm on June 5, 2025

Full Name:

UCINetID (e.g., ucinetid@uci.edu):

Question 0: (5 points)

Canvas includes detailed guidelines on how homework solutions should be formatted for submission to Gradescope. To receive full credit on this question, be sure that you read and follow these guidelines carefully! In particular, please take care that:

- There are two Gradescope assignments for Homework 7. For the first assignment, submit a PDF containing your answers for questions 1-2. For the second assignment, submit your Python code for question 1, which will be checked by an autograder. *For this assignment, you do not need to submit your code for question 2.*
- For the PDF submission, you must fill in this PDF template with your answers. All pages must remain in their original order when uploading your assignment to Gradescope. *Do not rearrange, add, or remove any pages from the provided PDF template.*
- For multiple-choice questions, ensure that you *completely fill in the bubble next to your selected answer*, and provide an explanation or justification to support your answer.
- Enter your final answers (to a precision of 3 decimal places) in the answer boxes, and write your explanations and/or math justifications in the space provided below each question. *Always show the work needed to produce your answers, not just the final result.* Math may be handwritten, but if possible, please type any sentences.
- For question 1, use the provided Python file. After adding your code to the template, submit that file separately to the second Gradescope assignment. In addition to uploading your Python code, be sure you provide answers and explanations in the designated areas of the PDF. *Do not include code for question 1 in the PDF.*
- If any question asks you to create a plot, insert an image showing the plot in the PDF.
- Check that your answers (especially scanned math) are readable within Gradescope, and that content has not been cut-off by the page margins.

Question 1: (30 points)

The MovieLens dataset (http://movielens.org) contains ratings for M movies, recorded as integers between 1 and 5, for a community of N users. Most users have only rated a few of the entire set of possible movies, so the data is stored as a sparse $M \times N$ matrix X, where M is the number of movies and N is the number of users. For this assignment we have extracted a small subset of the overall database, containing $M = 500$ movie titles and $N = 943$ users. The training and test data represent the same users, but each matrix has a different set of *observed*, non-zero user-movie rating pairs. You will learn factor analysis and PCA models from the training ratings, and use them to predict test ratings.

Let x_{ij} be the rating that user i gives to movie j. Not all user-movie pairs are observed in training data: let $r_{ij} = 1$ if x_{ij} is observed, and $r_{ij} = 0$ otherwise. The set of observed ratings for user i is then $x_i^o = \{x_{ij} \mid r_{ij} = 1\}$. Factor analysis explains ratings x_i for each user i via a K-dimensional latent vector $z_i \in \mathbb{R}^K$:

$$p(z_i) = \text{Norm}(z_i \mid 0, I_K), \qquad p(x_i \mid z_i) = \text{Norm}(x_i \mid Wz_i + m, V).$$

Here, W is a $M \times K$ *factor-loading* matrix, m is an $M \times 1$ mean vector, and V is an $M \times M$ diagonal covariance matrix. Given a matrix of partially observed ratings x_i^o, the EM algorithm will find maximum likelihood estimates of parameters W, m, V, and corresponding marginals for the hidden variables z. We have provided an implementation of the EM algorithm for learning factor analysis models from sparse observations like these.

In this question, we evaluate the empirical performance of several methods for predicting movie ratings. We compute the test root mean square error (RMSE) using only the ratings that were not observed in the training set, and the following formula:

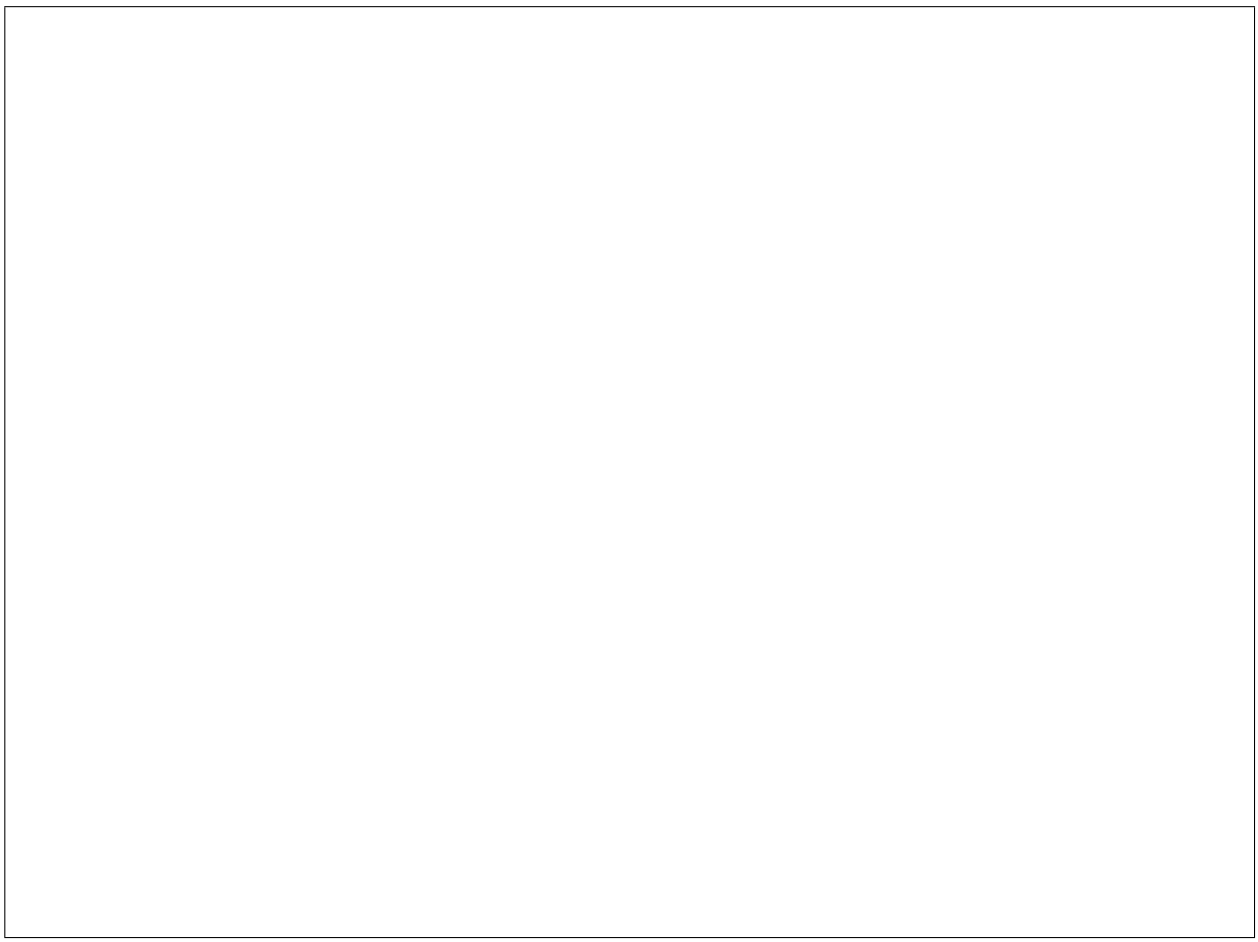
$$\text{RMSE} = \sqrt{\frac{1}{N_h} \sum_{i=1}^{N} \sum_{j \in H_i} (x_{ij} - \hat{x}_{ij})^2}.$$

Here, H_i is the set of test movies for user i, N_h is the total number of ratings in the test dataset, and $\hat{x}_{ij}$ is the rating predicted by the model under evaluation.

a) As a simple baseline, for each movie in the corpus, compute the average of the observed training ratings x^o across all users. Then for each test item, simply predict the mean rating of the corresponding movie. Calculate the test RMSE for this method.

Test RMSE = [ ]

b) *Next, we consider a simple heuristic method for dimensionality reduction with sparse data. First, fill in the missing entries of the training movie rating matrix using the mean predictions from part (a). Apply principal component analysis (PCA) to this matrix using the* `scikit-learn` *Python implementation (see demonstration code for details). Find low-dimensional representations z_i for each user by using the top $K = \{1, 2, \ldots, 15\}$ principal components, and use these to reconstruct the missing ratings x_i^h. Plot RMSE versus K. Is this reconstruction better than the baseline?*



Briefly discuss how PCA predictions compare to the baseline from part (a):

c) For the heuristic dimensionality reduction method of part (b), what should the performance approach as $K \to M$, the number of movies? Briefly justify your answer.

d) Run the provided EM factor analysis code to estimate low-dimensional representations z_i, the factor matrix W, mean vector m, and variances V. For each $K \in \{1, 2, \ldots, 15\}$, run EM for 100 training iterations. After training, use these estimated quantities to reconstruct the missing ratings. Plot RMSE versus K. How do the two methods (Factor Analysis and PCA) compare? What choice of K leads to the best performance?

Best Factor Analysis performance achieved for $K =$

Briefly discuss how the performance of Factor Analysis compares to PCA:

Question 2: (50 points)

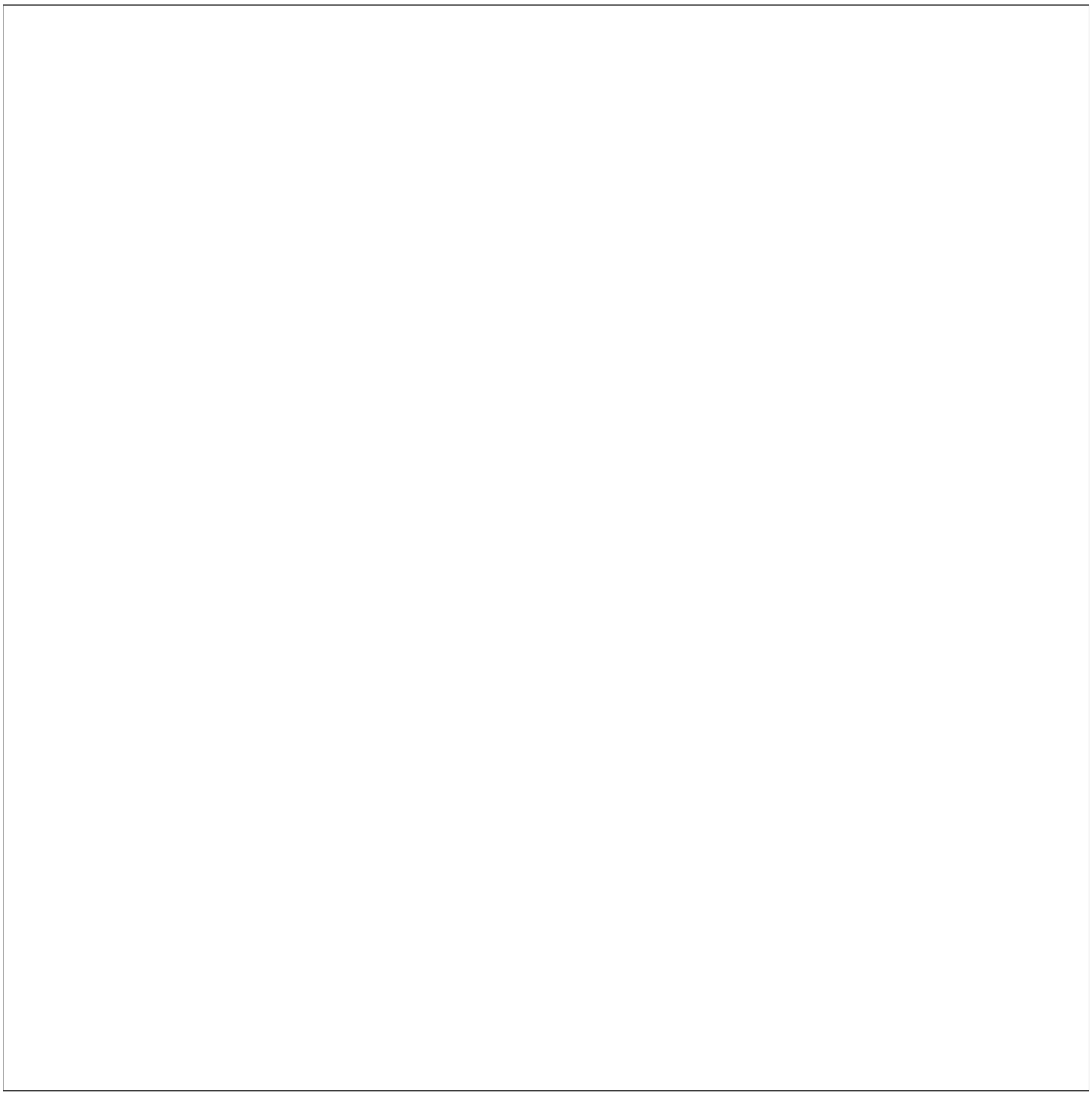
In this question, we will learn generative models that represent data x_i of dimension M by low-dimensional vectors $z_i \in \mathbb{R}^K$, where $K \ll M$. In our experiments, the data are images of handwritten digits with M pixels, from the MNIST dataset. We will learn *variational autoencoders* (VAEs) that model images x_i as a non-linear function of their embedding z_i:

$$p(z_i) = \text{Norm}(z_i \mid 0, I_K), \qquad p(x_i \mid z_i) = \prod_{j=1}^{M} \text{Bernoulli}(x_{ij} \mid \mu_j(z_i; \theta)).$$

Here x_{ij} is the value of pixel j in image i, whose mean μ_j is determined by a deep neural network with parameters θ. As a baseline, we will also consider *probabilistic PCA* (PPCA) models that assume the data mean is a linear function of z_i. PPCA is a special case of the factor analysis model from Question 1 where $V = \sigma^2 I_M$.

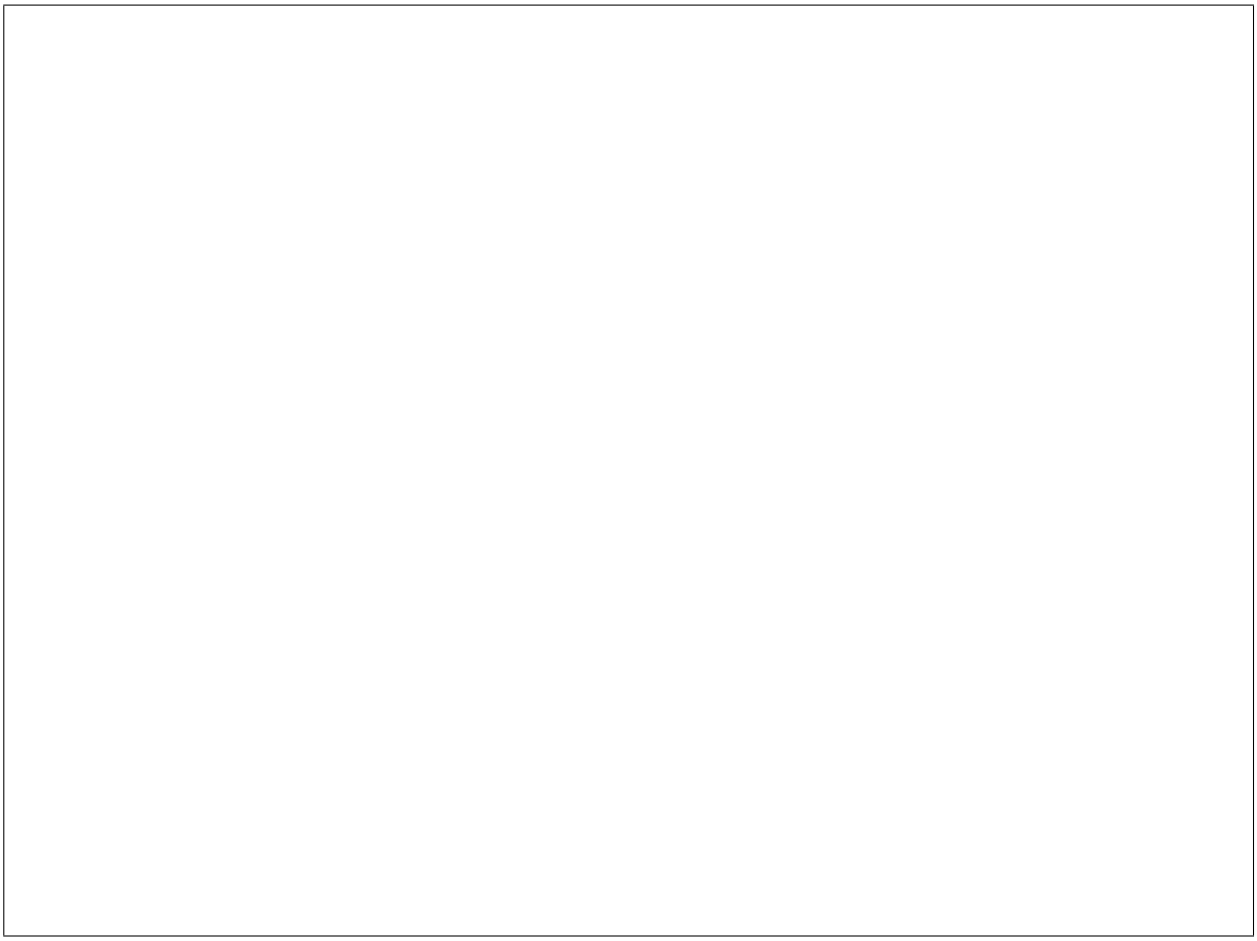
a) As a simple baseline, train two PPCA models with latent space dimensions $K = 2, 50$. Use the maximum likelihood PPCA training algorithm implemented in `ppca.py`*. (You do not need to report anything for this part.)*

b) For each of the two PPCA models, plot the means of the reconstructions of 7 images from the MNIST dataset.



c) Generate and plot 25 samples from each of the two PPCA models. You can generate each sample by first drawing a "code" z_i from the Gaussian prior in the latent space, and then using the `decode(...)` *function to transform z_i into an image $x_i = Wz_i + m$.*

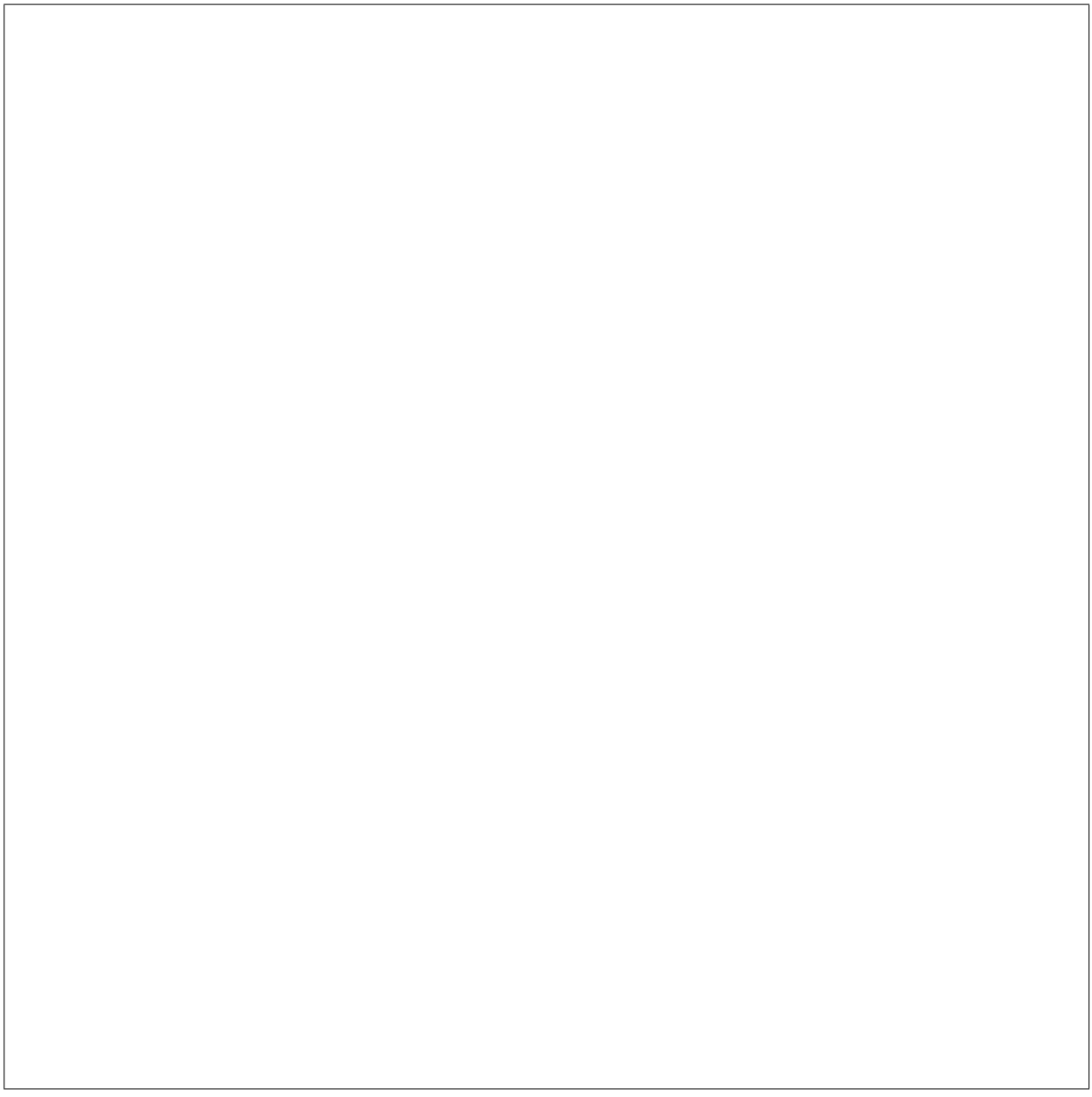
d) Train two VAE models with latent space dimensions $K = 2, 50$. For each VAE, run the stochastic gradient training algorithm for 50 epochs using the Adam optimizer, with a learning rate of 0.001 and a batch size of 128. For each VAE model, plot the average training loss per epoch.



e) For each of the two VAE models, plot the means of the reconstructions of 7 images from the MNIST dataset. Briefly discuss how the VAE reconstructions compare to the PPCA reconstructions from part (b).

Brief discussion of how VAE reconstructions compare to PPCA reconstructions:

f) Generate and plot 25 samples from each of the two VAE models. You can generate each sample by first drawing a "code" z_i from the Gaussian prior in the latent space, and then using the `decode(...)` *function to transform z_i into an image $x_i = \mu(z_i; \theta)$. Briefly discuss how the VAE samples compare to the PPCA samples from part (c).*



Brief discussion of how VAE samples compare to PPCA samples:

g) For the PPCA and VAE models with $K = 2$ latent dimensions, use `plot_2d_clusterings` *to create a plot of the latent encodings for each model, color-coded based on the digit label. Compare how well the PPCA and VAE models cluster digits in the latent space.*

Brief discussion of how PPCA embeddings compare to VAE embeddings: