CS552: Generative AI

Homework 1

Will Buchta

1/28/25

Problem 1

1. **Image Autoregression [20 pts]**

In this exercise you will generate images by autoregressively sampling from a list of conditional probability tables (CPTs). This process will produce the pixels of a 5×5-pixel *binary* image $\mathbf{x} \in \{0,1\}^{25}$, where $\mathbf{x} = \begin{bmatrix} x_1 & \dots & x_5 \\ & \ddots & \\ x_{21} & \dots & x_{25} \end{bmatrix}$. These CPTs have been pre-computed from an image dataset. To get started, first call `gunzip image_cpts.pkl.gz` (the expanded file is about 500MB) and then call `pickle.load(open("image_cpts.pkl", "rb"))` to load the CPTs. Each CPT i represents $p(x_i \mid x_1, \dots, x_{i-1})$, represented as a multidimensional numpy array with indices `[x1][x2]...[xi]`. (Python indices start at 0; hence, CPT $i = 1$ is stored at index 0 of the list in `image_cpts.pkl`.) Your tasks:

(a) Use the CPTs and `np.random.rand` to sample 100 images from the joint distribution $p(\mathbf{x})$. To sample each image, first sample pixel $x_1 \sim p(x_1)$. Then, sample $x_2 \sim p(x_2 \mid x_1)$, i.e., the conditional distribution of pixel x_2 given the value of x_1 has already been drawn. Proceed for each x_i until you generate the entire image. After sampling 100 images in this way, display them in a 10×10 "collage" of images. If you sample correctly, the contents of these images should be recognizable objects. Include the collage in the PDF you submit. **[8 pts]**

See the collage I created below. I inverted the colors of the characters to make them easier to see. As seen, the CPT represents letters of the alphabet.

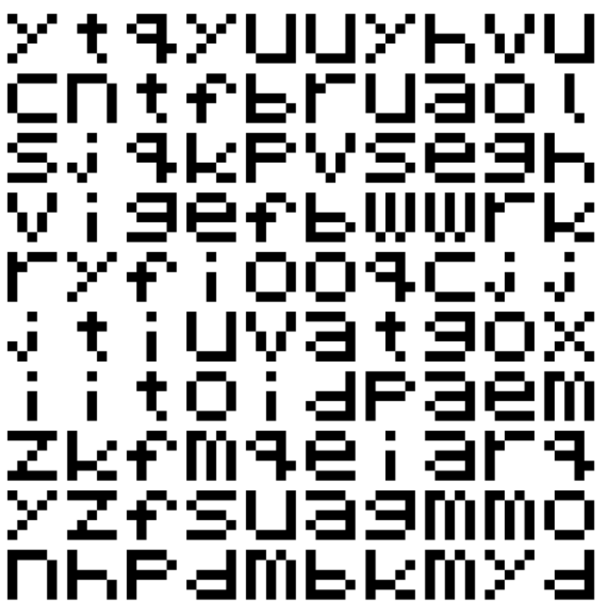


Figure 1: CPT Table Produced Collage

b)

Figure 2a: LSTM Produced Collage

I first created a tensor dataset with 26 datapoints, each datapoint being a vector of length 25, being a unique alphabet character. To create X and Y samples to train the model, I padded each vector to have 0 at the start. This way, the x[0:24] inclusive represent the inputs to the LSTM, and x{1:25] represent the ideal outputs. The training was quite noisy, and after 400 epochs, I achieved a loss of 0.163, down from the start of ~0.7.

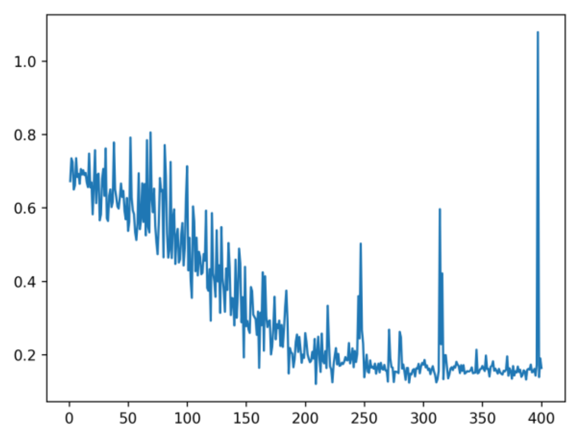


Figure 2b: Training of LSTM

Problem 2

a) See the function "fit_ppca" in problem2.py
b) I created the function x_to_z to help project x samples to the latent space. For d=2, I produced the following scatter plot (Figure 3). Out of curiosity, I also created a 3D scatter plot for d=3 (Figure 4)

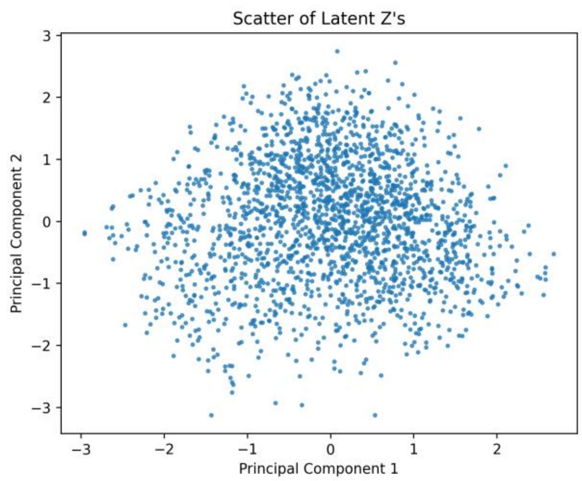


Figure 3: 2D projection of faces using PPCA

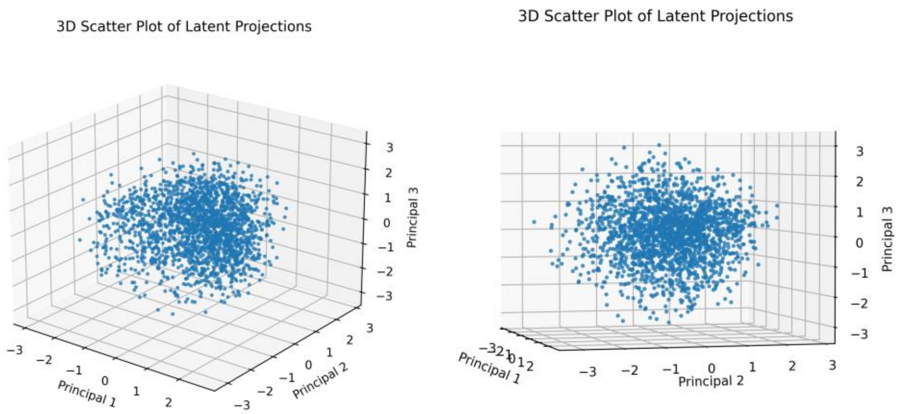


Figure 4: 3D projection of faces using PPCA

c) "For d ∈ {16, 32, 64}, use the trained model to "reconstruct" 25 randomly selected images from the provided dataset" → See figures 5, 6, and 7 below. The quality of the reconstruction increases greatly with increasing latent dimension d. This is trivial, as more information can be stored with a larger latent dimension.

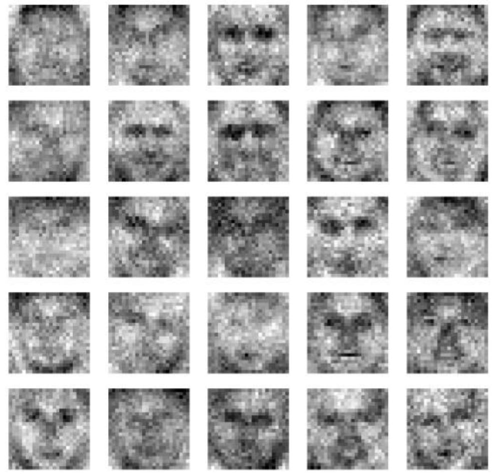


Figure 5: Reconstructed faces, d=16

Figure 6: Reconstructed faces, d=32

Figure 7: Reconstructed faces, d=64

d) “Generate 100 new faces from scratch (just choose a value of d that gives good visual results) by sampling from p(z) and then computing E[x | z]. Create a 10 × 10 collage of these images and include it in your PDF”.

Figure 8: Randomly generated faces, latent dim = 64

e) See Figure 9 below. Each row represents a different latent dimension of the face that is being perturbed, and each column represents a different amount the dimension is being perturbed by. The first column represents the PPCA’d image, with no alterations to the latent vector. It is hard to see, but it looks like the dimension in the 4th row affects how “smiley” the face is.

Figure 9: Randomly perturbed faces

Problem 3

a) MLE Derivative derivation:

3a) Gradient of MLE $\qquad \sigma(x) = \frac{1}{1+e^{-x}}$

Chain Rule: $\frac{d}{dx} f(g(x)) = f'(g(x))g'(x) \qquad \frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x)) \qquad \frac{d}{dx}\log(x) = \frac{1}{x}$

$\frac{d}{dx} f(g(h(x))) = f'(g(h(x)) g'(h(x)) h'(x)$

$$\sum_{i<j} e_{ij} \log\left(\sigma\left(a x^{(i)T} x^{(j)} + b\right)\right) + (1-e_{ij})\log\left(1-\sigma\left(a x^{(i)T} x^{(j)} + b\right)\right) \qquad e_{ij} \in \{0, 1\}$$

$$\frac{\delta}{\delta a}\left[e_{ij} \log\left(\sigma\left(a x^{(i)T} x^{(j)} + b\right)\right) + (1-e_{ij})\log\left(1-\sigma\left(a x^{(i)T} x^{(j)} + b\right)\right)\right]$$

$$= e_{ij} \frac{\delta L}{\delta a} \overset{f}{\log}\left(\overset{g}{\sigma}\left(\overset{h}{a} x^{(i)T} x^{(j)} + b\right)\right) + (1-e_{ij})\frac{\delta}{\delta a}\log\left(1-\sigma\left(a x^{(i)T} x^{(j)} + b\right)\right)$$

$$= e_{ij} \frac{\overset{g'(h(x))}{\sigma(a x^{(i)T} x^{(j)} + b)\left(1-\sigma(a x^{(i)T} x^{(j)} + b)\right)}}{\underset{f'(g(h(x)))}{\sigma(a x^{(i)T} x^{(j)} + b)}} * \overset{h'(x)}{x^{(i)T} x^{(j)}} + (1-e_{ij}) \frac{-\sigma(a x^{(i)T} x^{(j)} + b)\cancel{\left(1-\sigma(a x^{(i)T} x^{(j)} + b)\right)}}{\cancel{1-\sigma(a x^{(i)T} x^{(j)} + b)}} * x^{(i)T} x^{(j)}$$

$$= x^{(i)T} x^{(j)} \left(e_{ij}\left(1-\sigma(a x^{(i)T} x^{(j)} + b)\right) - (1-e_{ij})\,\sigma(a x^{(i)T} x^{(j)} + b)\right)$$

When $e_{ij} = 0$: $\frac{\delta L}{\delta a} = -\sigma(a x^{(i)T} x^{(j)} + b)\, x^{(i)T} x^{(j)}$

$e_{ij} = 1$: $\frac{\delta L}{\delta a} = 1-\sigma(a x^{(i)T} x^{(j)} + b)\, x^{(i)T} x^{(j)}$

$$\boxed{\frac{\delta L}{\delta a} = e_{ij} - \sigma(a x^{(i)T} x^{(j)} + b)\, x^{(i)T} x^{(j)}}$$

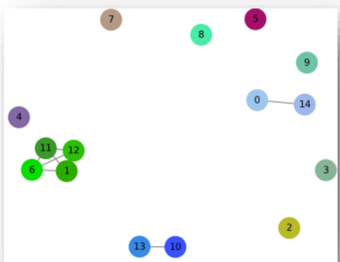
$\frac{\delta L}{\delta b}$: same derivation, no $x^{(i)T} x^{(j)}$ term

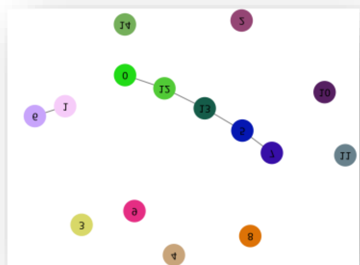
$$\boxed{\frac{\delta L}{\delta b} = e_{ij} - \sigma(a x^{(i)T} x^{(j)} + b)}$$

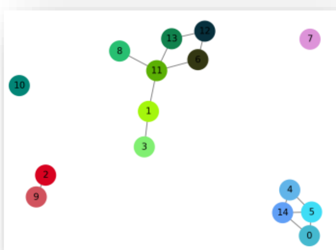
b) Training run for MLE (note that it converges pretty quickly) → see problem3.py

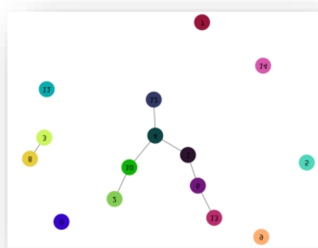
```
C:\Users\bucht\OneDrive - Worcester Polytechnic Institute (wpi.edu)\CS Courses\CS552_GAI\hw1>python problem3.py
Epoch   1 MLE: -16.875125 a: 3.091292 b: -3.415255
Epoch   2 MLE: -16.382760 a: 4.082604 b: -4.117396
Epoch   3 MLE: -16.585647 a: 4.668903 b: -4.558403
Epoch   4 MLE: -16.877045 a: 5.075740 b: -4.871851
Epoch   5 MLE: -17.158169 a: 5.379587 b: -5.109087
Epoch   6 MLE: -17.408950 a: 5.616377 b: -5.295545
Epoch   7 MLE: -17.627445 a: 5.806081 b: -5.445814
Epoch   8 MLE: -17.816378 a: 5.961030 b: -5.569086
Epoch   9 MLE: -17.979464 a: 6.089399 b: -5.671549
Epoch  10 MLE: -18.120317 a: 6.196902 b: -5.757579
Epoch  11 MLE: -18.242143 a: 6.287691 b: -5.830385
Epoch  12 MLE: -18.347694 a: 6.364883 b: -5.892390
Epoch  13 MLE: -18.439306 a: 6.430870 b: -5.945468
Epoch  14 MLE: -18.518955 a: 6.487533 b: -5.991099
Epoch  15 MLE: -18.588310 a: 6.536369 b: -6.030464
Epoch  16 MLE: -18.648785 a: 6.578590 b: -6.064524
Epoch  17 MLE: -18.701584 a: 6.615187 b: -6.094069
Epoch  18 MLE: -18.747732 a: 6.646980 b: -6.119750
Epoch  19 MLE: -18.788106 a: 6.674652 b: -6.142113
Epoch  20 MLE: -18.823459 a: 6.698775 b: -6.161617
```

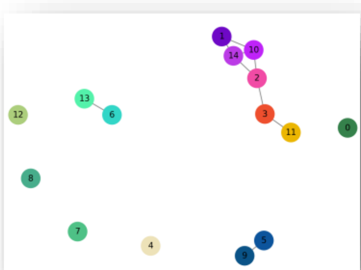
c) Using the trained parameters of a and b, I generated 5 random graphs. See below:











Problem 4

Solution for W:

4) MLE Derivation

$$P(y\,|\,X) = \mathcal{N}(y = X^T w, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - x^T w)^2}{2\sigma^2}\right)$$

log likelihood $L(w, \sigma^2) = \log \prod P(y^{(i)} \,|\, x^{(i)}, w, \sigma^2)$

using log rules:

$$= \sum_1^n \log P(y^{(i)} \,|\, x^{(i)}, w, \sigma^2)$$

$$= \sum_1^n \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - x_i^T w)^2}{2\sigma^2}\right)\right)$$

$$= \sum_1^n \left[\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + \log\left(\exp\left(-\frac{(y_i - x_i^T w)^2}{2\sigma^2}\right)\right)\right]$$

$$= \sum_1^n \left[\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \frac{(y_i - x_i^T w)^2}{2\sigma^2}\right]$$

$$= \sum_1^n \log\left((2\pi\sigma^2)^{-\frac{1}{2}}\right) - \sum_1^n \frac{(y_i - x_i^T w)^2}{2\sigma^2}$$

$$\frac{\delta}{\delta w}\left[\sum_1^n \cancelto{0}{\log\left((2\pi\sigma^2)^{-\frac{1}{2}}\right)} - \sum_1^n \frac{(y_i - x_i^T w)^2}{2\sigma^2}\right] = -\frac{\delta}{\delta w} \sum_{i=1}^n \frac{(y_i - x_i^T w)^2}{2\sigma^2}$$

$$= \frac{-1}{2\sigma} \sum_{i=1}^n \frac{\delta}{\delta w}(y_i - x_i^T w)^2 = \frac{-1}{\sigma^2} \sum_{i=1}^n (-x_i)(y_i - x_i^T w)$$

Transpose comes out during chain rule application

$$\boxed{\frac{\delta}{\delta w} = \frac{1}{\sigma^2} \sum_{i=1}^n x_i \,(y_i - x_i^T w)}$$

$$\frac{\delta L}{\delta w} = 0 = \frac{1}{\sigma^2} \sum_{i=1}^n x_i \,(y - x_i^T w)$$

$$0 = \sum_{i=1}^n x_i \,(y - x_i^T w) = \sum_{i=1}^n (x_i y - x_i x_i^T w)$$

$$0 = \sum_{i=1}^n x_i y - \sum_{i=1}^n x_i x_i^T w$$

$$\sum_{i=1}^n x_i y = \sum_{i=1}^n x_i x_i^T w \rightarrow \sum_{i=1}^n x_i y = w \sum_{i=1}^n x_i x_i^T$$

$$\boxed{w = \frac{\sum_{i=1}^n x_i y}{\sum_{i=1}^n x_i x_i^T}}$$

Solution for variance:

$$\frac{\delta}{\delta\sigma}\left[\sum_{1}^{n}\log\left(\left(2\pi\sigma^2\right)^{-\frac{1}{2}}\right)-\sum_{i=1}^{n}\frac{(y_i-x_i^Tw)^2}{2\sigma^2}\right]$$

$$=\frac{\delta}{\delta\sigma}\left[\frac{-1}{2}\sum_{i=1}^{n}\log\left(2\pi\sigma^2\right)-\frac{1}{2}\sigma^{-2}\sum_{i=1}^{n}(y_i-x_i^Tw)^2\right] \qquad \frac{d}{d\sigma}\sigma^{-2}=-2\sigma^{-3}$$

$$=\frac{\delta}{\delta\sigma}\left[\frac{-n}{2}\log\left(2\pi\sigma^2\right)-\frac{1}{2\sigma^2}\cdot\sum_{i=1}^{n}(y_i-x_i^Tw)^2\right]$$

$$=\frac{-n}{4\pi\sigma^2}\cdot 4\pi\sigma-\frac{1}{2}\cdot -2\sigma^{-3}\sum_{i=1}^{n}(y_i-x_i^Tw)^2$$

$$\boxed{\frac{\delta L}{\delta\sigma}=\frac{-n}{\sigma}+\frac{1}{\sigma^3}\sum_{i=1}^{n}(y_i-x_i^Tw)^2}$$

$$0=\frac{-n}{\sigma}+\frac{1}{\sigma^3}\sum_{i=1}^{n}(y_i-x_i^Tw)^2$$

$$\frac{n}{\sigma}=\frac{1}{\sigma^3}\sum_{i=1}^{n}(y_i-x_i^Tw)^2$$

$$\frac{\sigma^3}{\sigma}=\frac{1}{n}\sum_{i=1}^{n}(y_i-x_i^Tw)^2 \Longrightarrow \boxed{\sigma^2=\frac{1}{n}\sum_{i=1}^{n}(y_i-x_i^Tw)^2}$$