ECE 8803-GGDL Homework 1

Arati Ganesh

September 15, 2023

1. Data generation in low and high dimensions. In class, we identified high dimensionality as a critical challenge in learning generative models. In this problem, we like to train a generative model in small dimensions and then examine why it does not scale well

   (a) Consider black and white images of size 3 × 3. Develop a probabilistic model that generates images with a probability proportional to the number of black pixels in that image. Draw five samples and visualize them. (HINT: use the Categorical distribution)

   i. Generate 2^9 (512) Samples: Start by generating a total of 512 binary images. Each pixel can be either black (0) or white (1).

   ii. Define Category Probabilities: Set up a categorical distribution. Category 0 represents image with no black pixels, while Category 9 represents images with 9 black pixels. These probabilities are distributed such that images with more black pixels have a higher likelihood of being generated.

Category	0	1	2	3	4	5	6	7	8	9
Probability	0.0	0.0222	0.0444	0.0667	0.0889	0.1111	0.1333	0.1556	0.1778	0.2

Table 1: Categorical Distribution

   iii. Count Black Pixels: For each generated image, we count the number of black pixels and determine its probability based on the black pixel count and create a dataset.

   iv. Visualize Sampled Images: Visualize the 5 sampled images to observe the results.

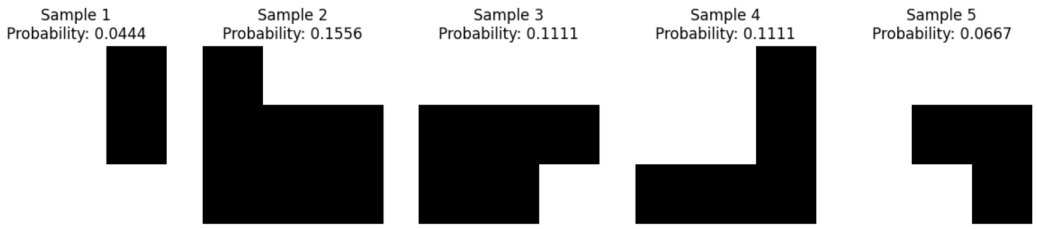


Figure 1: Samples from Distribution

   (b) Assume we no longer know the data generation rule and want to learn it from a handful of data points. Now, sample 200 unique images. Consider the probability of generating each training image as the label and train a Multilayer Perceptron (MLP). Use the MLP with proper normalization to develop a new probabilistic model to generate images. Draw five samples and visualize them

   i. Sampling Images : From the 512 generated samples, randomly sample 200 images and flatten it.

   ii. Building Network : Build a simple MLP with 3 layers as below.

iii. Training the MLP - Train the MLP with 3x3 input images for 500 epochs. The final layer consists of a sigmoid activation which outputs a probability.

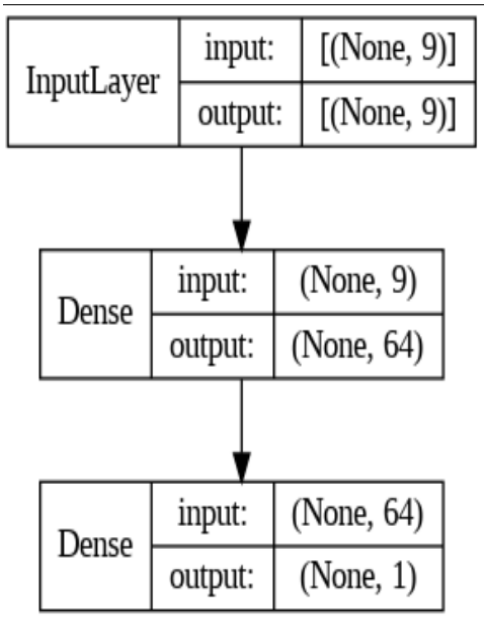


Figure 2: MLP 3x3

iv. Plot the distribution : Plot the distribution of the trained model alongside the original test data (50 data points).

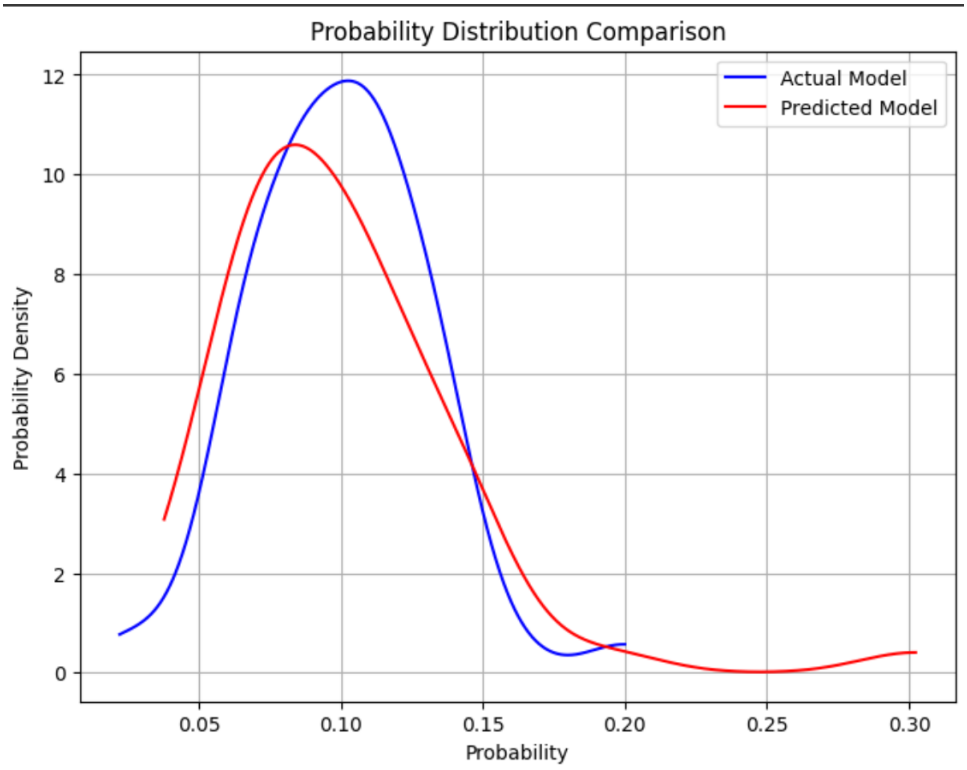


Figure 3: Probability Distribution Comparison

v. Generate 5 samples : Feed the model with the test data and reject the images with a whose predicted probability is not within 10 percent threshold of the true probability. The images not rejected are the images generated by the model.

Figure 4: Samples from Distribution

(c) Now consider the same problem in 28 × 28 dimensions. Draw 200 random samples and train an MLP. Explain why following the same strategy to learn a generative model does not scale. Bring at least two reasons

1. Generate 200 samples of 28x28 dimension : Can't generated all combinations because memory intensive.
2. Building Network : Build a simple MLP with 5 layers as below.

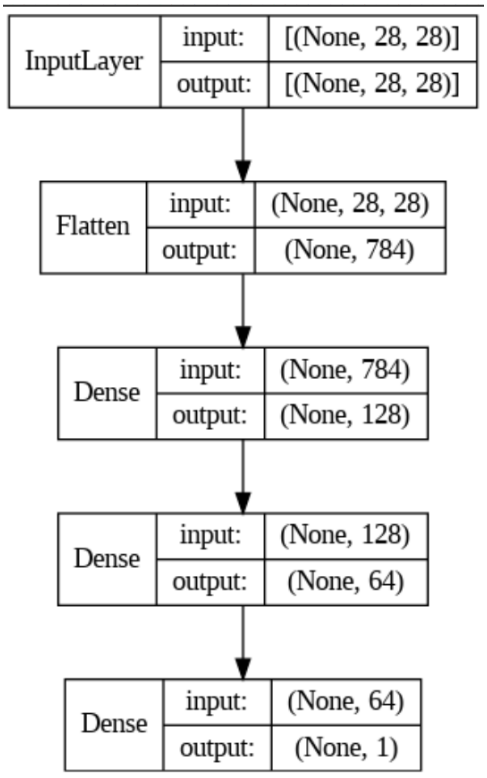


Figure 5: MLP 28X28 input

3. Training the MLP : Train the MLP with 28x28 input images for 500 epochs. The final layer consists of a sigmoid activation which outputs a probability.

4. Plot the distribution : Plot the distribution of the trained model alongside the original test data (50 data points).

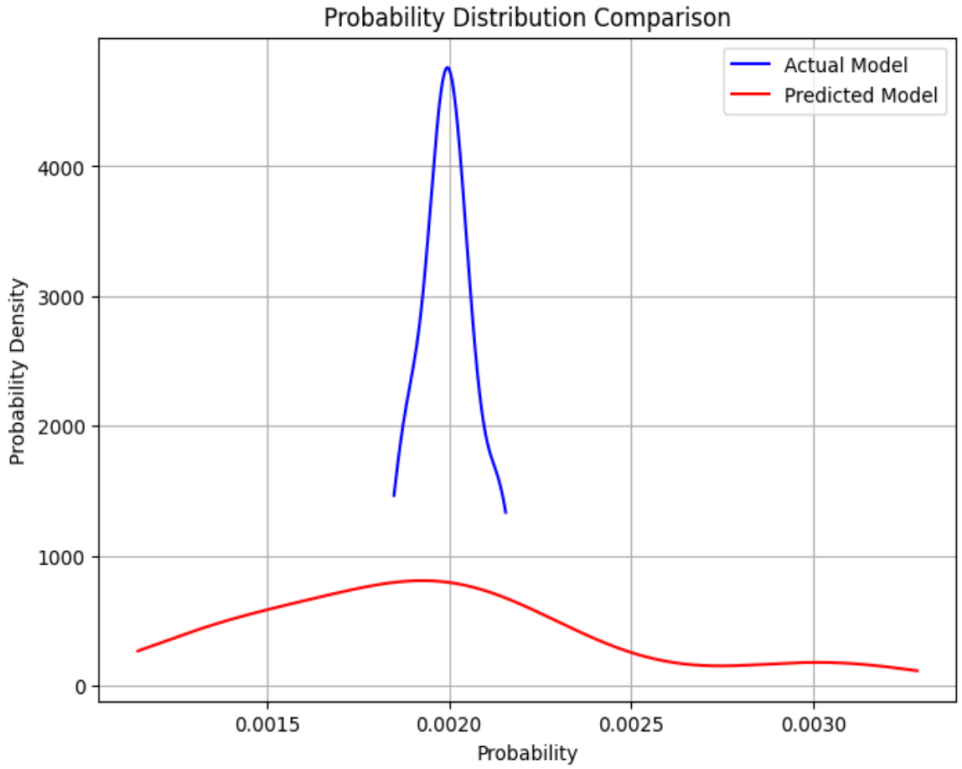


Figure 6: MLP 28X28 input

Reasons why it does not scale -

i. Exponential Growth in Image possibilities: For a 28x28 image, there are (28*28) possibilities, making it computationally infeasible to model and store all probabilities.

ii. Curse of High Dimensionality: In high-dimensional spaces, data becomes sparse, and the training data needed to cover the space grows exponentially. Training models on such data requires an impractical amount of training data to be effective.

---

2. Generative modeling with independent assumption. Consider the binarized MNIST digit dataset https://www.tensorflow.org/datasets/catalog/binarized_mnist. Train a pixel-independent generative model by properly forming the likelihood and minimizing the maximum likelihood. Detail the mathematical steps to arrive at the model. Generate ten samples. What do you learn?

Given that the dataset consists of binary values (0s and 1s), we can naturally formulate the problem as a **Bernoulli distribution**. The primary objective is to maximize the likelihood of observing the dataset under this distribution. In other words, we aim to find the set of model parameters that best explains the observed binary data.

By maximizing the likelihood, we strive to find the most plausible configuration of parameters that generate data similar to the observed binary images.

Likelihood for a Bernoulli distribution is given by -

$$L(p) = \prod_{i=1}^{n} p^{x_i}(1-p)^{(1-x_i)}$$

$$\text{LogL}(\theta) = \sum_{i=1}^{n} \log\left(p^{x_i}(1-p)^{(1-x_i)}\right)$$

$$\text{LogL}(\theta) = \sum_{i=1}^{n} \left(x_i \log(p) + (1-x_i)\log(1-p)\right)$$

$$\text{LogL}(\theta) = \log p \sum_{i=1}^{n} x_i + \log(1-p) \sum_{i=1}^{n} (1-x_i)$$

Differentiate and set to zero:

$$\sum_{i=1}^{n} x_i - p\sum_{i=1}^{n} x_i = p\sum_{i=1}^{n}(1-x_i)$$

$$p = \frac{\sum_{i=1}^{n} x_i}{n}$$

Therefore, the Maximum Likelihood Estimator (MLE) for a Bernoulli distribution is the **sample mean** value.

Steps to model a pixel-independent generative model:

- Load the Binary MNIST Dataset.
- Iterate through all the pixels and keep a count of black and white pixels.
- Calculate the probability of black and white based on the count.
- Sample and generate 10 images considering each pixel's probability. p = [black, white]

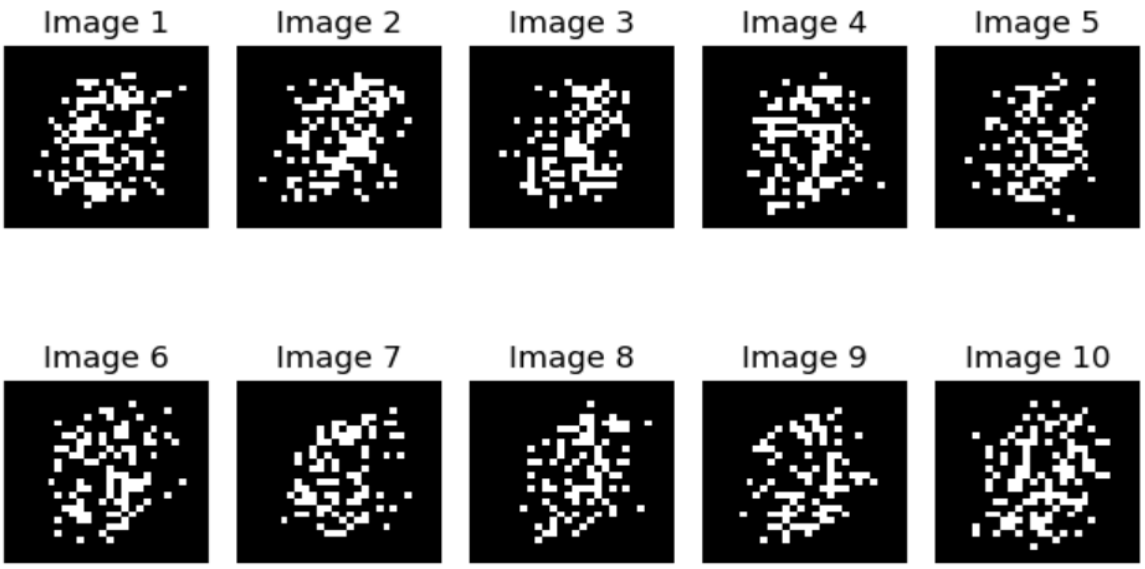
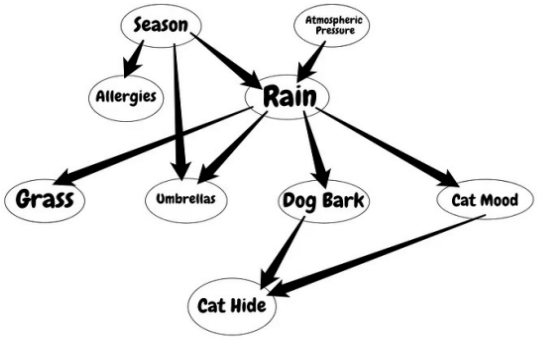


Figure 7: Samples from Pixel Independent Model

The pixel-independent model could only learn noise and no meaningful information. Therefore it is understood that the assumption of pixel independence is **too strong** and it **cannot accurately model real-world problems.**

---

3. Cats, dogs, and DAGs. What conditional independencies can we infer from this DAG? Write your steps to find all the independencies. Which independencies do you agree with, and which do you not agree?



From the DAG, write the probability equation

$$\begin{aligned} P(S, AP, A, R, G, U, DB, CM, CH) = {} & P(S) \cdot P(AP) \cdot P(A|S) \\ & \cdot P(R|AP, S) \cdot P(G|R) \cdot P(U|S, R) \cdot P(DB|R) \cdot P(CM|R) \\ & \cdot P(CH|DB, CM) \end{aligned}$$

From the chain rule of probability

$$\begin{aligned} P(S, AP, A, R, G, U, DB, CM, CH) = {} & P(S) \cdot P(AP|S) \cdot P(A|S, AP) \\ & \cdot P(R|S, AP, A) \cdot P(G|S, AP, A, R) \cdot P(U|S, AP, A, R, G) \\ & \cdot P(DB|S, AP, A, R, G, U) \cdot P(CM|S, AP, A, R, G, U, DB) \\ & \cdot P(CH|S, AP, A, R, G, U, DB, CM) \end{aligned}$$

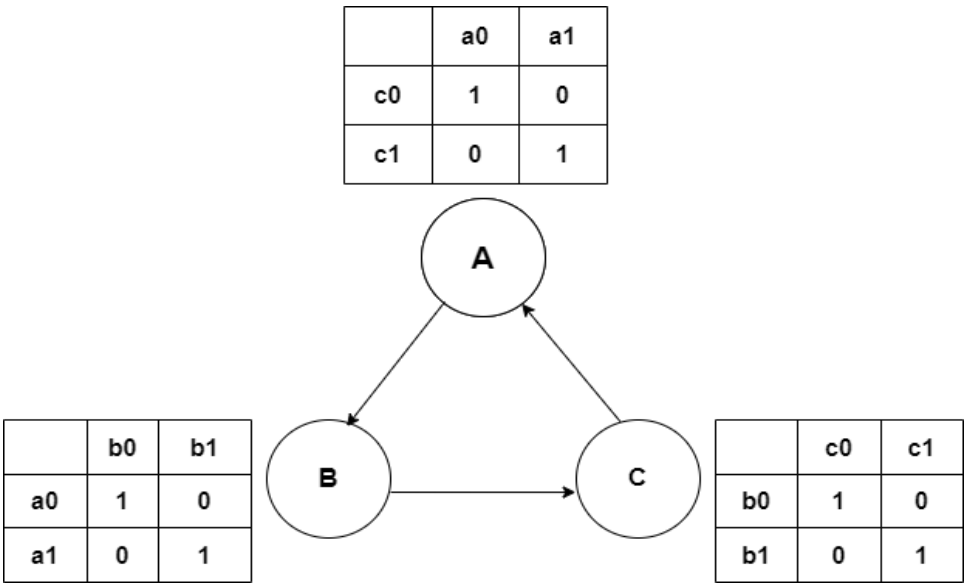
$$\begin{aligned} &P(AP) = P(AP|S) \implies AP|S \\ &P(A|S) = P(A|S, AP) \implies A \perp\!\!\!\perp AP \mid S \\ &P(R|S, AP) = P(R|S, AP, A) \implies R \perp\!\!\!\perp A \mid S, AP \\ &P(G|R) = P(G|S, AP, A, R) \implies G \perp\!\!\!\perp S, AP, A \mid R \\ &P(U|S, R) = P(U|S, AP, A, R, G) \implies U \perp\!\!\!\perp AP, A, G \mid S, R \\ &P(DB|R) = P(DB|S, AP, A, R, G, U) \implies DB \perp\!\!\!\perp S, AP, A, G \mid R \\ &P(CM|R) = P(CM|S, AP, A, R, G, U, DB) \implies CM \perp\!\!\!\perp S, AP, A, G, U \mid R \\ &P(CH|DB, CM) = P(CH|S, AP, A, R, G, U, DB, CM) \implies CH \perp\!\!\!\perp S, AP, A, R, G, U \mid DB, CM \end{aligned}$$

I do not agree with -
1. Cat Mood to be independent of Dog Bark given Rain.

I agree with -
1. Allergies to be independent of Rain given Season

---

4. Joint probability of CGs. Consider the following directed cyclic graph. Prove why would the joint probability not be legal for this graph. Hint: design a counter-example assuming a simple distribution for each variable (e.g., binary) and show that the attained joint distribution is invalid.



	a0	a1
c0	1	0
c1	0	1

	b0	b1
a0	1	0
a1	0	1

	c0	c1
b0	1	0
b1	0	1

Figure 8: Directed Cyclic Graph

For any valid joint distribution, two restrictions should be satisfied:

(a) All probabilities in the distribution should be non-negative:

$$P(X_i \geq 0 \quad \text{for all} \quad i)$$

(b) All the probabilities should sum to one:

$$\sum_{\text{all possible outcomes}} P(\text{outcome}) = 1$$

And we can obtain the joint distribution easily: in all 2^3 possibilities, at least two are 1:

$$P(a_0, b_0, c_1) = P(a_0, b_1, c_0) = P(a_0, b_1, c_1) = P(a_1, b_0, c_0) = P(a_1, b_0, c_1) = P(a_1, b_1, c_0) = 0$$

$$P(a_0, b_0, c_0) = P(a_0|c_0)P(b_0|a_0)P(c_0|b_0) = 1$$

$$P(a_1, b_1, c_1) = P(a_1|c_1)P(b_1|a_1)P(c_1|b_1) = 1$$

Then,

$$\sum_{A,B,C} P(A,B,C) = 2 > 1$$

So, the joint distribution is invalid. Therefore the joint probability of a Directed Cyclic Graph is not legal.

---

5. NADE and MADE on MNIST. We will now compare the performance of a NADE and MADE implementation on the binarized MNIST dataset. We will use the implementation in https://github.com/EugenHotaj/pytorch-generative, which you will need to clone into your working directory (make sure you have the necessary Pytorch environment or follow the installation steps in the repository page).

   (a) Train the provided NADE model (check the reproduce() function provided with the model) for 30 epochs and a hidden dimension of size 300. Visualize a few samples from the trained model (you can do this either by using the sample() method provided with the model, or visualizing the tensorboard log after training).

   The figure below shows a few samples generated by the NADE models after training for 30 epochs.

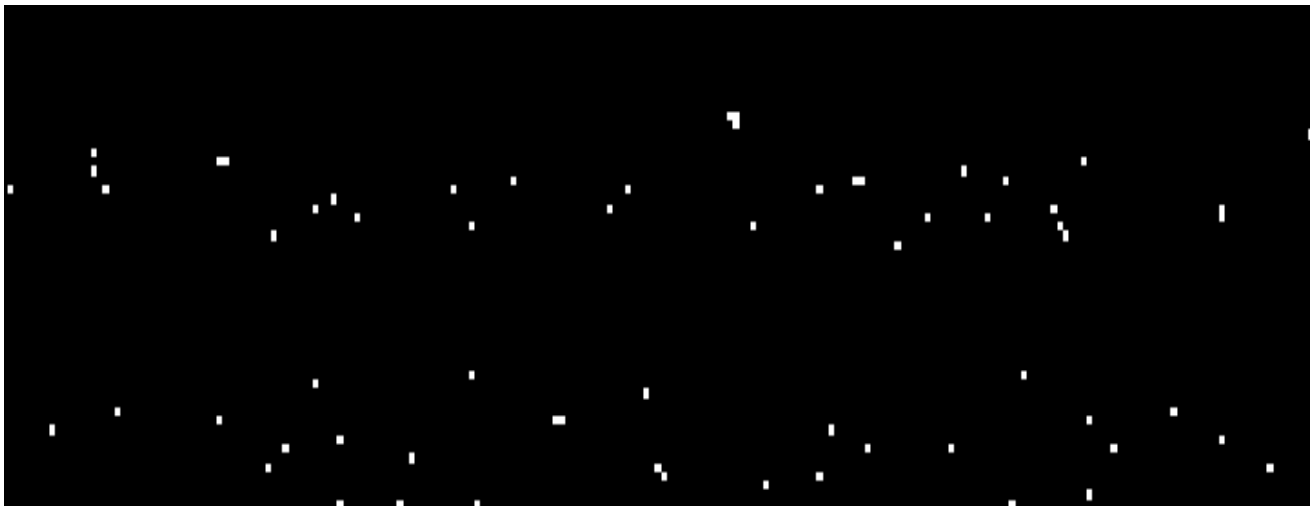


Figure 9: NADE sample images

   (b) Repeat the process with the provided MADE model (using the same parameters described above). Does one of the models perform better (generates more plausible samples) than the other? Explain why this is the case

   The figure below shows a few samples generated by the MADE models after training for 30 epochs.

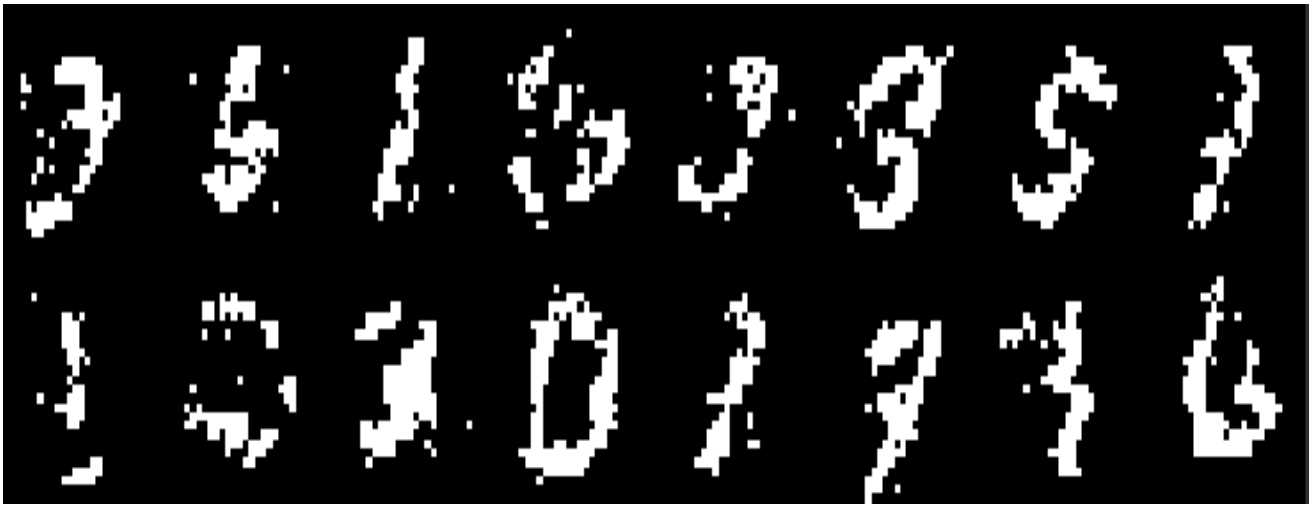


Figure 10: MADE sample images

Generation is slow in NADE because the model generates one pixel (or one random variable) at a time. Since MADE uses a parallel network, image generation is faster.

(c) Which model would be more suitable for this particular type of data? Justify your answer using generative performance, runtime, complexity, or other aspects you consider relevant.

1. Generative Performance - Generative performance of MADE is better because the ordering of pixels is not sequential but random.
2. Runtime Performance - Runtime performance is better in MADE because it is parallel and has fewer number of parameters because of masking.

---

6. Autoregressive autoencoders. Take the autoregressive autoencoder example from the slides of lecture 3. Now assume we increase the problem dimension from 3 to 4 by adding a new variable x4 and consider the following new ordering for the variables: x2, x4, x1, x3. Follow the steps we reviewed in class and find the new masking matrices that turn the autoencoder into an autoregressive model. Detail all your steps. Note: there is no unique solution.

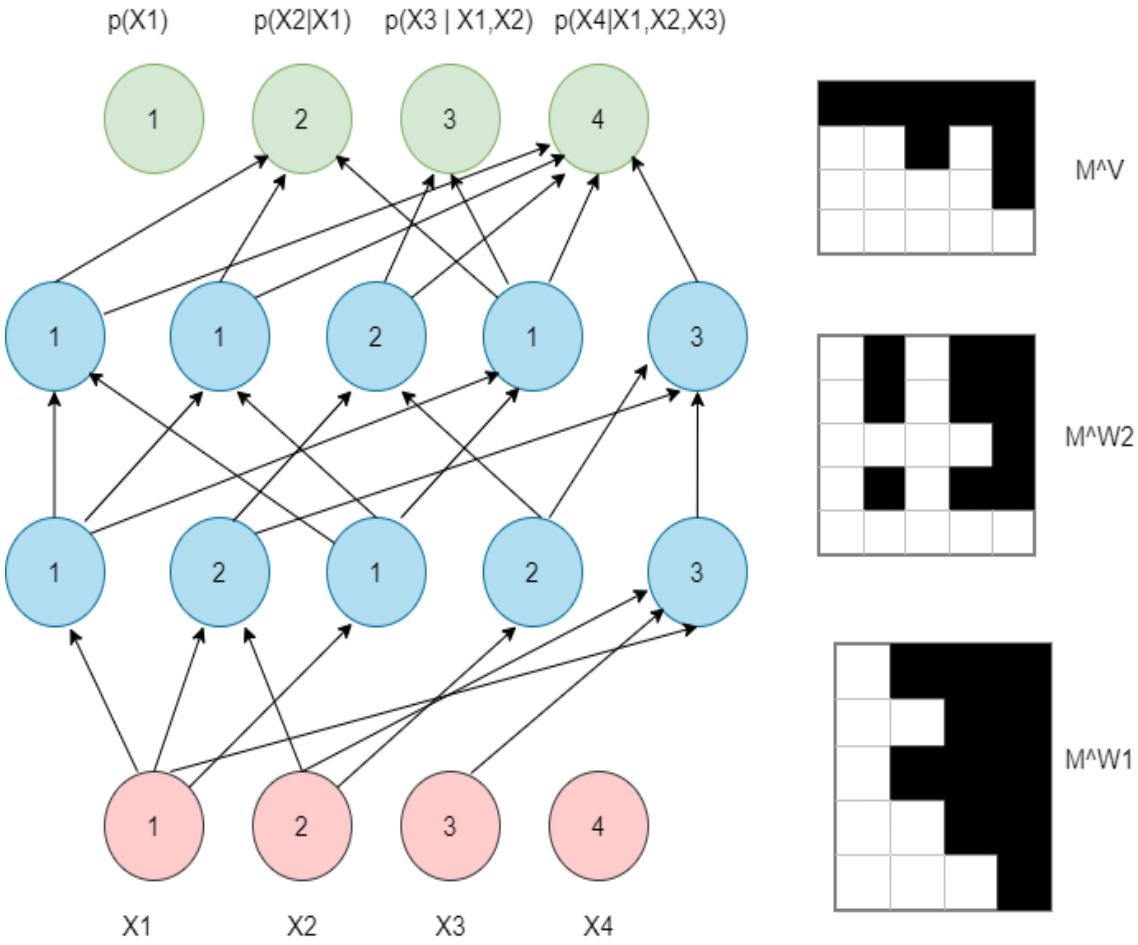


Figure 11: Autoregressive autoencoder

Masks are use to disallow certain paths.

- Begin with a random ordering of input and output layers. For the hidden node randomly number till one less than total input nodes.
- For each hidden unit, connect to all the nodes less than or equal to it's numbering
- For output unit, connect to all nodes less than it's numbering

---

7. MLE on the exponential family. Suppose $D = \{x(1), x(2), \ldots, x(n)\}$ is drawn from an exponential distribution $\exp(\lambda)$. Form the maximum likelihood and estimate λ.

   Consider a set of independent and identically distributed (i.i.d) observations $\{x_1, x_2, \ldots, x_n\}$ sampled from the exponential distribution with parameter λ. The probability density function (PDF) of the exponential distribution is given by:

   $$f(x; \lambda) = \lambda e^{-\lambda x} \text{ for } x \geq 0$$

   Consider a set of independent and identically distributed (i.i.d) observations $\{x_1, x_2, \ldots, x_n\}$ sampled from the exponential distribution with parameter λ. The probability density function (PDF) of the exponential distribution is given by:

$$f(x;\lambda) = \lambda e^{-\lambda x} \text{ for } x \geq 0$$

The likelihood function for these observations is the product of the PDF for each observation:

$$L(\lambda) = \prod_{i=1}^{n} f(x_i;\lambda)$$

Substituting the PDF of the exponential distribution:

$$L(\lambda) = \prod_{i=1}^{n} \lambda e^{-\lambda x_i}$$

Since the observations are independent, the likelihood simplifies to:

$$L(\lambda) = \lambda^n e^{-\lambda \sum_{i=1}^{n} x_i}$$

To find the maximum likelihood estimate (MLE) for λ, we take the natural logarithm (ln) of the likelihood function to simplify calculations:

$$\ln(L(\lambda)) = \ln(\lambda^n e^{-\lambda \sum_{i=1}^{n} x_i})$$

Using properties of logarithms:

$$\ln(L(\lambda)) = n\ln(\lambda) - \lambda \sum_{i=1}^{n} x_i$$

Next, we compute the derivative of $\ln(L(\lambda))$ with respect to λ and set it equal to zero to find the maximum of the likelihood function:

$$\frac{d}{d\lambda}[\ln(L(\lambda))] = \frac{d}{d\lambda}\left[n\ln(\lambda) - \lambda \sum_{i=1}^{n} x_i\right] = 0$$

$$\ln(L(\lambda)) = n\ln(\lambda) - \lambda \sum_{i=1}^{n} x_i$$

Now, let's find the derivative of $\ln(L(\lambda))$ with respect to λ:

$$\frac{d}{d\lambda}[\ln(L(\lambda))] = n - \sum_{i=1}^{n} x_i\lambda$$

To find the maximum likelihood estimate (MLE), we set this derivative equal to zero:

$$n - \sum_{i=1}^{n} x_i\lambda = 0$$

Solving for λ:

$$\lambda = \frac{n}{\sum_{i=1}^{n} x_i}$$

So, the maximum likelihood estimate (MLE) for the parameter λ in the exponential distribution is:

$$\lambda_{\text{MLE}} = \frac{n}{\sum_{i=1}^{n} x_i}$$

This is the maximum likelihood estimate for λ based on the observed data.
