HOMEWORK 6

Sriram Ashokkumar
908 216 3750
https://github.com/srirama02/CS760/tree/main/HW6

Instructions: Use this latex file as a template to develop your homework. Submit your homework on time as a single pdf file. Please wrap your code and upload to a public GitHub repo, then attach the link below the instructions so that we can access it. Answers to the questions that are not within the pdf are not accepted. This includes external links or answers attached to the code implementation. Late submissions may not be accepted. You can choose any programming language (i.e. python, R, or MATLAB). Please check Piazza for updates about the homework. It is ok to share the results of the experiments and compare them with each other.

1 Implementation: GAN (50 pts)

In this part, you are expected to implement GAN with MNIST dataset. We have provided a base jupyter notebook (gan-base.ipynb) for you to start with, which provides a model setup and training configurations to train GAN with MNIST dataset.

(a) Implement training loop and report learning curves and generated images in epoch 1, 50, 100. Note that drawing learning curves and visualization of images are already implemented in provided jupyter notebook. (20 pts)

Procedure 1 Training GAN, modified from Goodfellow et al. (2014)

Input: m: real data batch size, n_z: fake data batch size
Output: Discriminator D, Generator G
for number of training iterations **do**
# Training discriminator
Sample minibatch of m noise samples $\{z^{(1)}, z^{(2)}, \cdots, z^{(n_z)}\}$ from noise prior $p_g(z)$
Sample minibatch of m examples $\{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\}$
Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d}\big(\frac{1}{m}\sum_{i=1}^{m}\log D(x^{(i)}) + \frac{1}{n_z}\sum_{i=1}^{n_z}\log(1 - D(G(z^{(i)})))\big)$$

Training generator
Sample minibatch of n_z noise samples $\{z^{(1)}, z^{(2)}, \cdots, z^{(n_z)}\}$ from noise prior $p_g(z)$
Update the generator by ascending its stochastic gradient:

$$\nabla_{\theta_g}\frac{1}{n_z}\sum_{i=1}^{n_z}\log D(G(z^{(i)}))\big)$$

end for
# The gradient-based updates can use any standard gradient-based learning rule. In the base code, we are using Adam optimizer (Kingma and Ba, 2014)

Expected results are as follows.

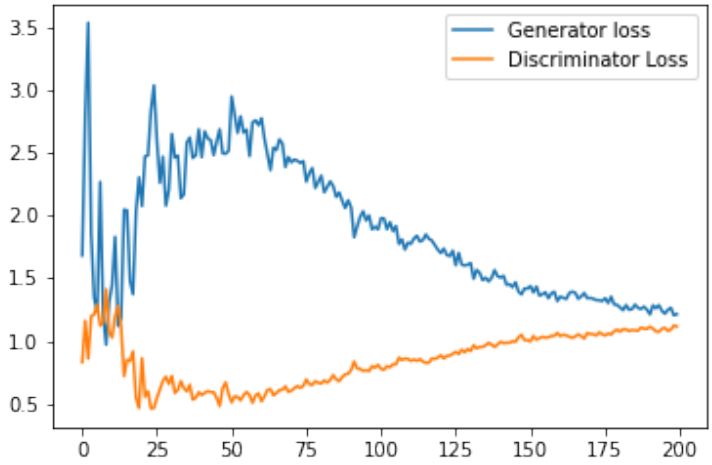
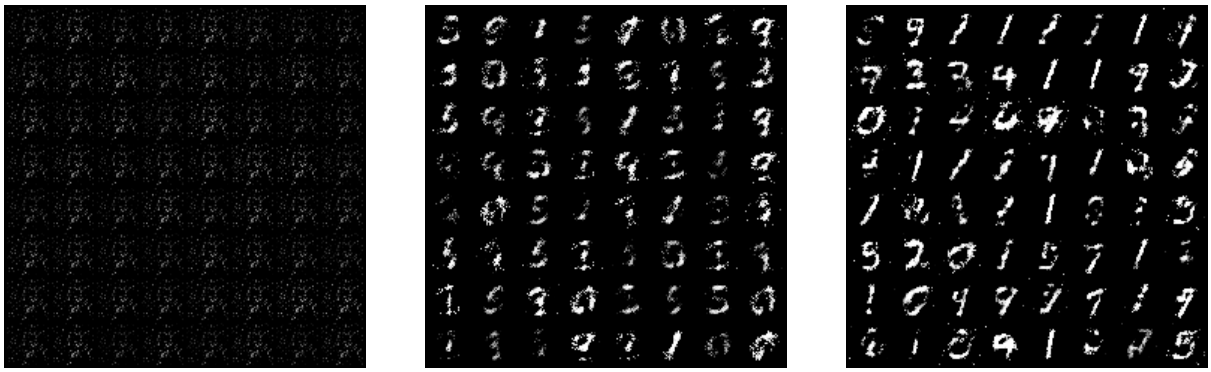


Figure 1: Learning curve



(a) epoch 1 (b) epoch 50 (c) epoch 100

Figure 2: Generated images by G

Solution:

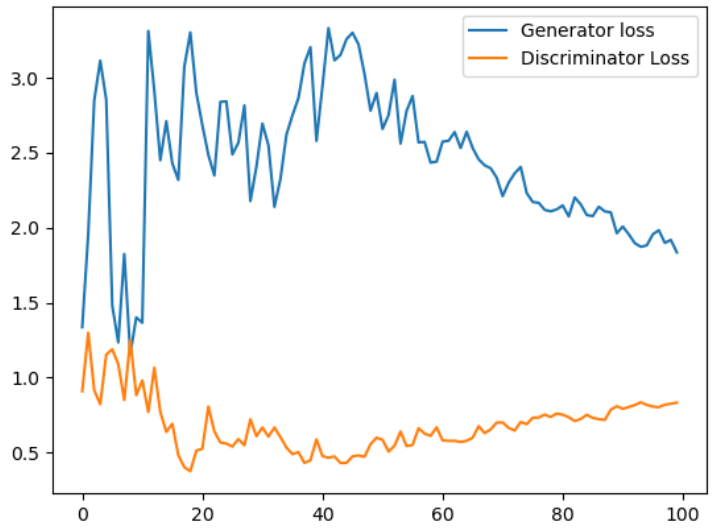


Figure 3: Learning curve for part a

(a) epoch 1 (b) epoch 50 (c) epoch 100

Figure 4: Generated images for part a

(b) Replace the generator update rule as the original one in the slide,
"Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{n_z} \sum_{i=1}^{n_z} \log(1 - D(G(z^{(i)})))$$

" , and report learning curves and generated images in epoch 1, 50, 100. Compare the result with (a). Note that it may not work. If training does not work, explain why it doesn't work.
You may find this helpful: https://jonathan-hui.medium.com/gan-what-is-wrong-with-the-gan-cost-function-6f594162ce01 (10 pts)

The training does not work as both the generator and descrimator loss goes to zero due to vanishing gradients problem as mentioned in the linked article.

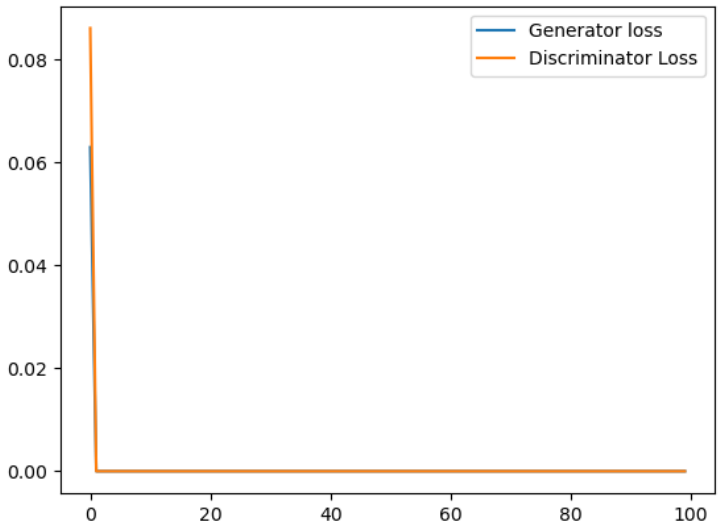
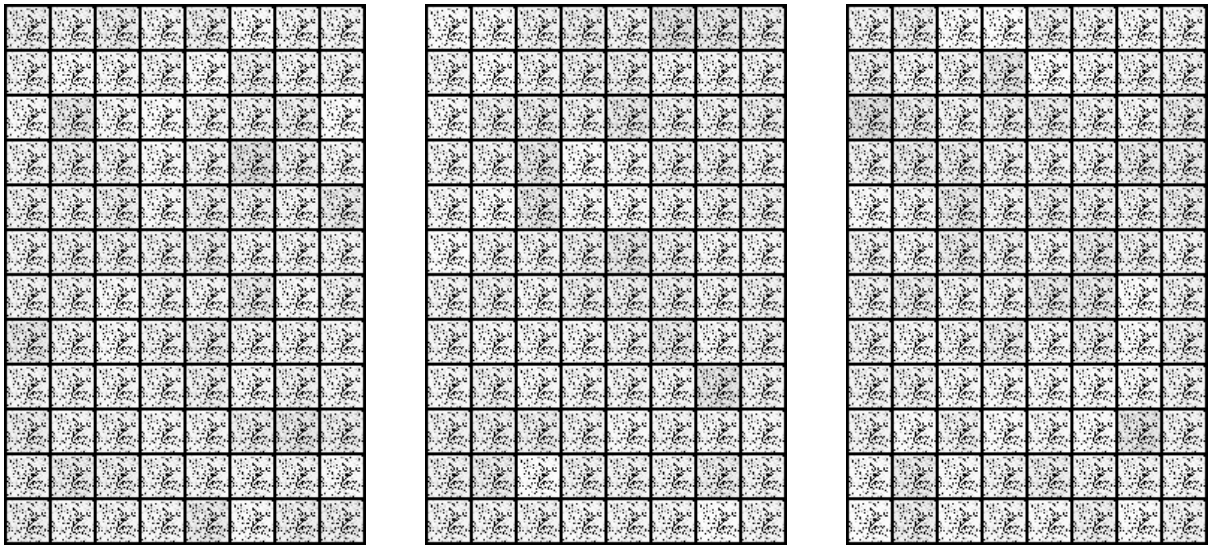


Figure 5: Learning curve for part b



(a) epoch 1 (b) epoch 50 (c) epoch 100

Figure 6: Generated images for part b

(c) Except the method that we used in (a), how can we improve training for GAN? Implement that and report your setup, learning curves, and generated images in epoch 1, 50, 100. This question is an open-ended question and you can choose whichever method you want. (20 pts)

I made a few changes. I added some label smoothing, where I used soft labels instead of hard labels. I also added some gaussian noise to the inputs of the discriminator which can help with overfitting. I also changed the generator loss function, so that instead of the standard BCE loss, I used negative of the discrimninator's predictions. I added learning rate scheduling so that the learning rates change dynamically during taining(decreases as the training progresses). I also use a smaller learning rate for the discriminator than the generator. The learning curve shown is over 200 epochs.

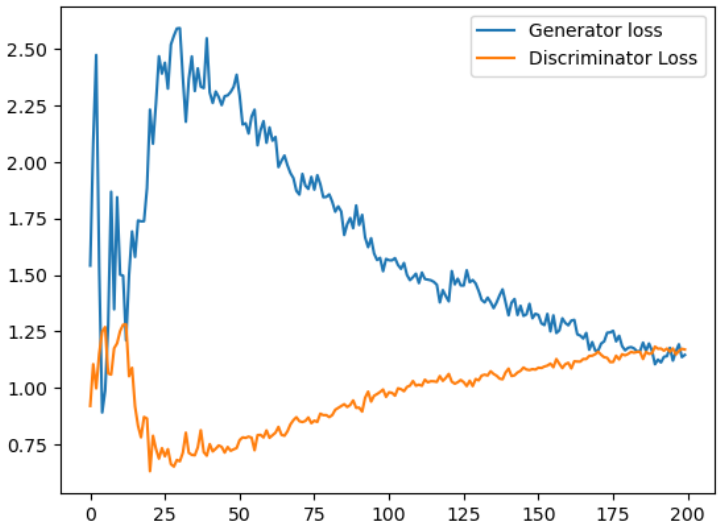


Figure 7: Learning curve for part c

(a) epoch 1 (b) epoch 50 (c) epoch 100

Figure 8: Generated images for part c

2 Directed Graphical Model [25 points]

Consider the directed graphical model (aka Bayesian network) in Figure 9.

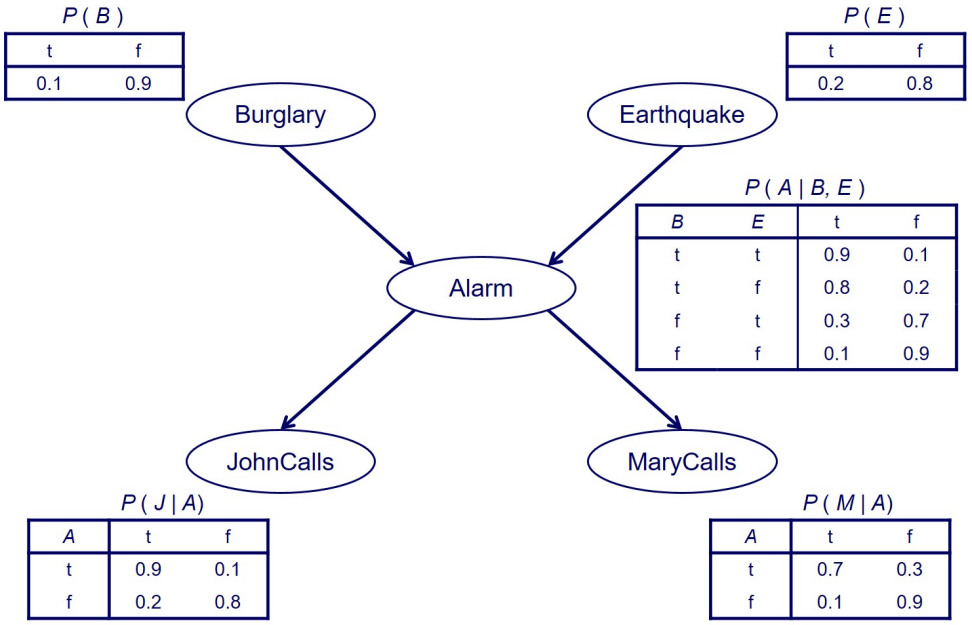


Figure 9: A Bayesian Network example.

Compute $P(B = t \mid E = f, J = t, M = t)$ and $P(B = t \mid E = t, J = t, M = t)$. (10 points for each) These are the conditional probabilities of a burglar in your house (yikes!) when both of your neighbors John and Mary call you and say they hear an alarm in your house, but without or with an earthquake also going on in that area (what a busy day), respectively.

1. $P(B = t \mid E = f, J = t, M = t) = \frac{P(B=t,E=f,J=t,M=t)}{P(B=t,E=f,J=t,M=t)+P(B=f,E=f,J=t,M=t)}$

$$P(B = t, E = f, J = t, M = t) = \sum_{a \in \{A, \neg A\}} P(B)P(\neg E)P(a|B, \neg E)P(J|a)P(M|a)$$
$$= 0.1 * 0.8 * 0.8 * 0.9 * 0.7 + 0.1 * 0.8 * 0.2 * 0.2 * 0.1 = 0.0406$$

$$P(B = f, E = f, J = t, M = t) = \sum_{a \in \{A, \neg A\}} P(\neg B)P(\neg E)P(a|\neg B, \neg E)P(J|a)P(M|a)$$
$$= 0.9 * 0.8 * 0.1 * 0.9 * 0.7 + 0.9 * 0.8 * 0.9 * 0.2 * 0.1 = 0.0583$$

$$P(B = t \mid E = f, J = t, M = t) = \frac{0.0406}{0.0406+0.0583} = 0.411$$

2. $P(B = t \mid E = t, J = t, M = t) = \frac{P(B=t,E=t,J=t,M=t)}{P(B=t,E=t,J=t,M=t)+P(B=f,E=t,J=t,M=t)}$

$$P(B = t, E = t, J = t, M = t) = \sum_{a \in \{A, \neg A\}} P(B)P(E)P(a|B, E)P(J|a)P(M|a)$$
$$= 0.1 * 0.2 * 0.9 * 0.9 * 0.7 + 0.1 * 0.2 * 0.1 * 0.2 * 0.1 = 0.01138$$

$$P(B = f, E = t, J = t, M = t) = \sum_{a \in \{A, \neg A\}} P(\neg B)P(E)P(a|\neg B, E)P(J|a)P(M|a)$$
$$= 0.9 * 0.2 * 0.3 * 0.9 * 0.7 + 0.9 * 0.2 * 0.7 * 0.2 * 0.1 = 0.03654$$

$$P(B = t \mid E = t, J = t, M = t) = \frac{0.01138}{0.01138+0.03654} = 0.237$$

3 Chow-Liu Algorithm [25 pts]

Suppose we wish to construct a directed graphical model for 3 features X, Y, and Z using the Chow-Liu algorithm. We are given data from 100 independent experiments where each feature is binary and takes value T or F. Below

is a table summarizing the observations of the experiment:

X	Y	Z	Count
T	T	T	36
T	T	F	4
T	F	T	2
T	F	F	8
F	T	T	9
F	T	F	1
F	F	T	8
F	F	F	32

1. Compute the mutual information $I(X, Y)$ based on the frequencies observed in the data. (5 pts) Given probabilities:

   - $P(X = T) = \frac{\text{Count of } (X=T)}{\text{Total Count}} = \frac{36+4+2+8}{100} = 0.50$
   - $P(X = F) = \frac{\text{Count of } (X=F)}{\text{Total Count}} = \frac{9+1+8+32}{100} = 0.50$
   - $P(Y = T) = \frac{\text{Count of } (Y=T)}{\text{Total Count}} = \frac{36+4+9+1}{100} = 0.50$
   - $P(Y = F) = \frac{\text{Count of } (Y=F)}{\text{Total Count}} = \frac{2+8+8+32}{100} = 0.50$
   - $P(X = T, Y = T) = \frac{\text{Count of } (X=T,Y=T)}{\text{Total Count}} = \frac{36+4}{100} = 0.40$
   - $P(X = T, Y = F) = \frac{\text{Count of } (X=T,Y=F)}{\text{Total Count}} = \frac{2+8}{100} = 0.10$
   - $P(X = F, Y = T) = \frac{\text{Count of } (X=F,Y=T)}{\text{Total Count}} = \frac{9+1}{100} = 0.10$
   - $P(X = F, Y = F) = \frac{\text{Count of } (X=F,Y=F)}{\text{Total Count}} = \frac{8+32}{100} = 0.40$

$$I(X, Y) = \sum_{x \in \{T,F\}} \sum_{y \in \{T,F\}} p(x, y) \log \left(\frac{p(x, y)}{p(x)p(y)} \right)$$

$$= P(X = T, Y = T) \log \left(\frac{P(X = T, Y = T)}{P(X = T)P(Y = T)} \right) + P(X = T, Y = F) \log \left(\frac{P(X = T, Y = F)}{P(X = T)P(Y = F)} \right)$$
$$+P(X = F, Y = T) \log \left(\frac{P(X = F, Y = T)}{P(X = F)P(Y = T)} \right) + P(X = F, Y = F) \log \left(\frac{P(X = F, Y = F)}{P(X = F)P(Y = F)} \right)$$

$$= 0.40 \cdot \log_2 \left(\frac{0.40}{0.50 \cdot 0.50} \right) + 0.10 \cdot \log_2 \left(\frac{0.10}{0.50 \cdot 0.50} \right)$$
$$+0.10 \cdot \log_2 \left(\frac{0.10}{0.50 \cdot 0.50} \right) + 0.40 \cdot \log_2 \left(\frac{0.40}{0.50 \cdot 0.50} \right)$$

$$= 0.278 \text{ bits}$$

$I(X, Y) \approx 0.278$ bits.

2. Compute the mutual information $I(X, Z)$ based on the frequencies observed in the data. (5 pts)

$$I(X, Z) = \sum_{x \in \{T,F\}} \sum_{z \in \{T,F\}} p(x, z) \log \left(\frac{p(x, z)}{p(x)p(z)} \right)$$

$$= 0.38 \cdot \log_2 \left(\frac{0.38}{0.50 \cdot 0.57} \right) + 0.12 \cdot \log_2 \left(\frac{0.12}{0.50 \cdot 0.43} \right)$$
$$+0.17 \cdot \log_2 \left(\frac{0.17}{0.50 \cdot 0.57} \right) + 0.33 \cdot \log_2 \left(\frac{0.33}{0.50 \cdot 0.43} \right)$$

$$= 0.134 \text{ bits}$$

$I(X, Z) \approx 0.134$ bits.

3. Compute the mutual information $I(Z, Y)$ based on the frequencies observed in the data. (5 pts)

$$I(Z,Y) = \sum_{z\in\{T,F\}} \sum_{y\in\{T,F\}} p(z,y) \log\left(\frac{p(z,y)}{p(z)p(y)}\right)$$

$$= 0.45 \cdot \log_2\left(\frac{0.45}{0.57 \cdot 0.50}\right) + 0.10 \cdot \log_2\left(\frac{0.10}{0.57 \cdot 0.50}\right)$$
$$+0.05 \cdot \log_2\left(\frac{0.05}{0.43 \cdot 0.50}\right) + 0.40 \cdot \log_2\left(\frac{0.40}{0.43 \cdot 0.50}\right)$$

$$= 0.398 \text{ bits}$$

$I(Z, Y) \approx 0.398$ bits.

4. Which undirected edges will be selected by the Chow-Liu algorithm as the maximum spanning tree? (5 pts)

   - $I(X, Y) \approx 0.278$ bits
   - $I(X, Z) \approx 0.134$ bits
   - $I(Z, Y) \approx 0.398$ bits

   Based on the calculated mutual info, the selected edges are the edge between Z and Y, since $I(Z, Y)$ has the highest value and the edge between X and Y, since $I(X, Y)$ is higher than $I(X, Z)$. The MST will include the edges (Z, Y) and (X, Y).

5. Root your tree at node X, assign directions to the selected edges. (5 pts)
   As the tree is rooted at X, the edge between X and Y is directed from X to Y. And as Y is connected to the root X, the edge between Y and Z is directed from Y to Z. Therefore, the directed edges in the tree, rooted at X, are $X \to Y$ and $Y \to Z$.

References

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.