HOMEWORK 6

AKSHAY KUMAR
9082076713

GitHub: https://github.com/AkshayK325/CS-760-ML.git

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 n_z noise samples $\{z^{(1)}, z^{(2)}, \cdots, z^{(n_z)}\}$ from noise prior $p_g(z)$
Sample minibatch of $\{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\}$
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)

The result figures are as shown below. (I have removed the old ones)

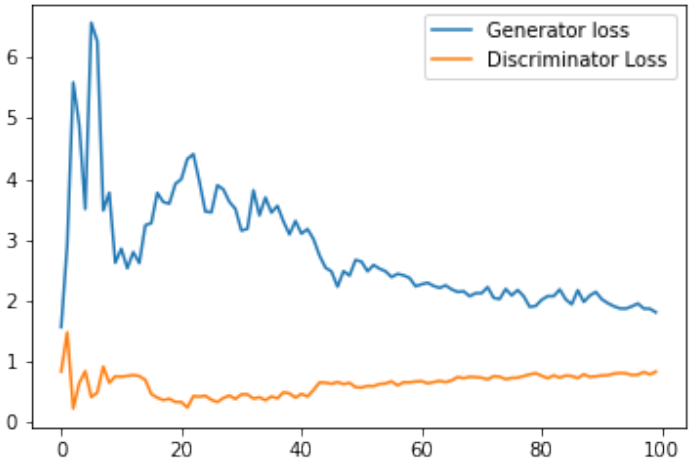
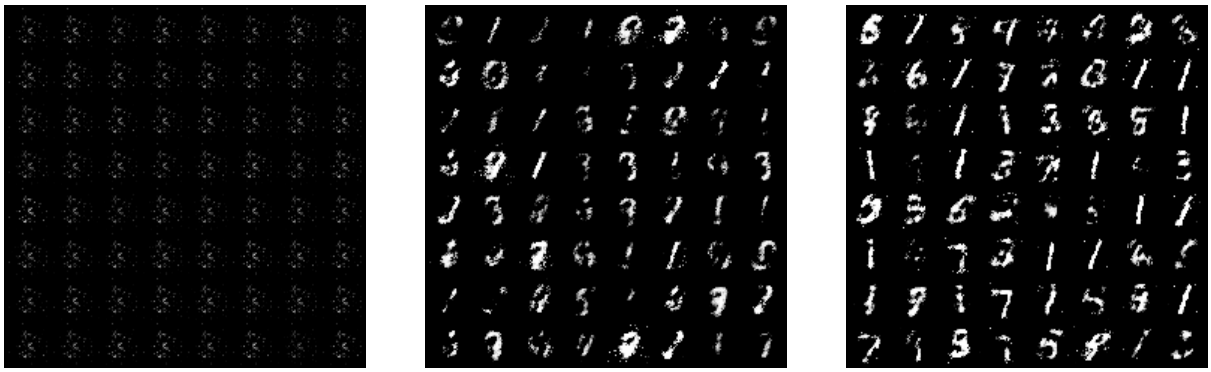


Figure 1: Learning curve



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

Figure 2: Generated images by G

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

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

" , 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)

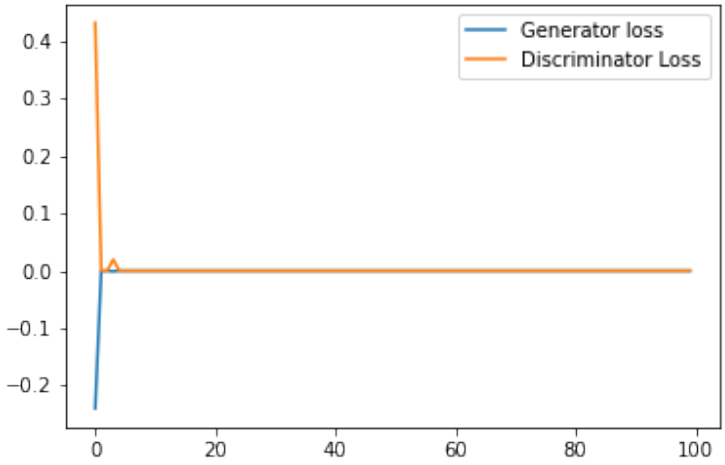
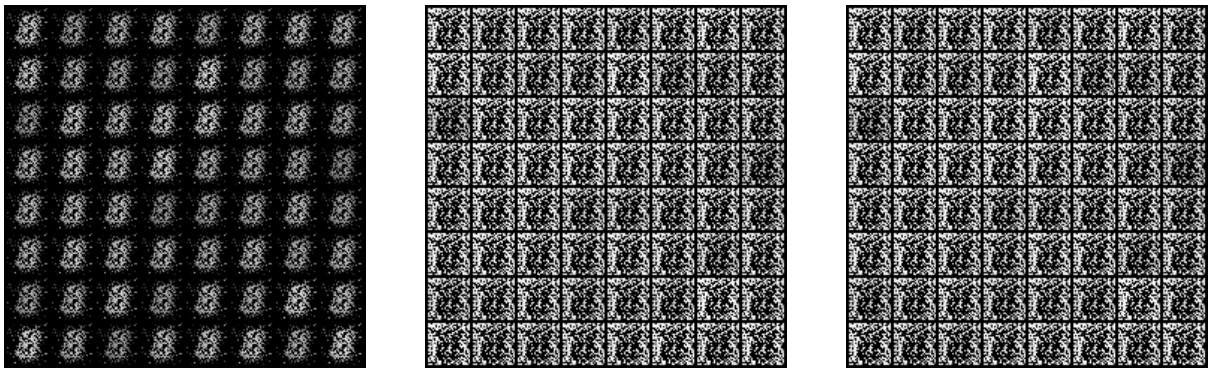


Figure 3: Learning curve for descending stochastic gradient G



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

Figure 4: Generated images by descending stochastic gradient G

- Early Saturation: When the generator produces poor-quality samples in the initial stages of training, the discriminator easily recognizes them as fake, causing $D(G(z))$ to be close to 0. This leads to $\log(1 - D(G(z)))$ saturating near zero.
- Vanishing Gradients: In the logarithmic function, as the input approaches 1, the gradient (slope) approaches zero. So, if the discriminator is too confident (i.e., $D(G(z))$ is very close to 0), the term $\log(1 - D(G(z)))$ will have a very small gradient, leading to vanishing gradients for the generator. This makes it difficult for the generator to learn and improve.

(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)

Least Squares Loss Function: The idea behind LSGAN is to use the least squares loss function for both the generator and the discriminator, which can help mitigate the vanishing gradients problem and produce higher quality results.

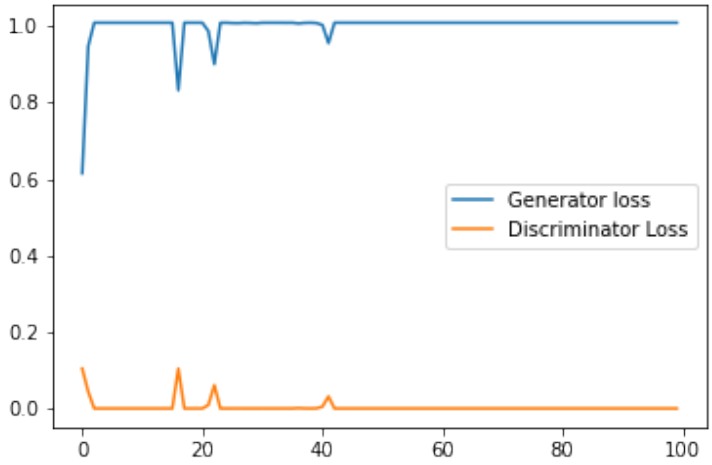
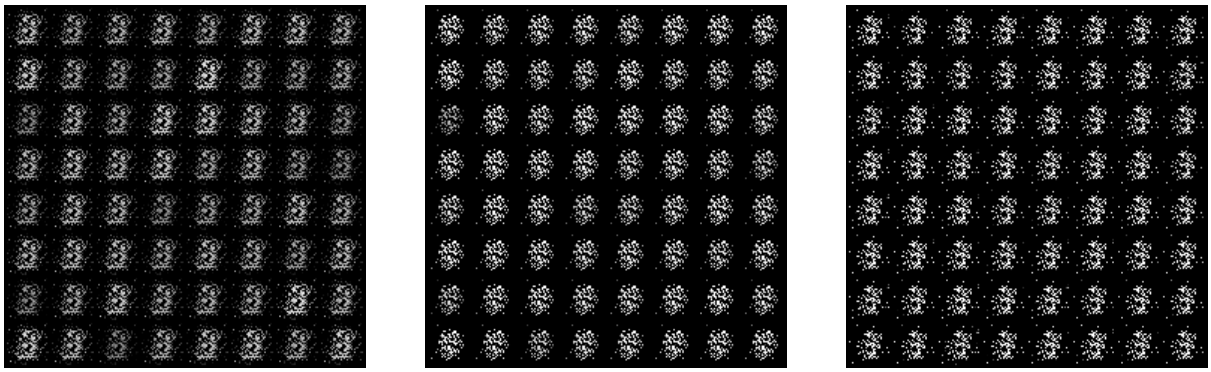


Figure 5: Learning curve for MSE loss



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

Figure 6: Generated images by MSE loss as objective

2 Directed Graphical Model [25 points]

Consider the directed graphical model (aka Bayesian network) in Figure 7.

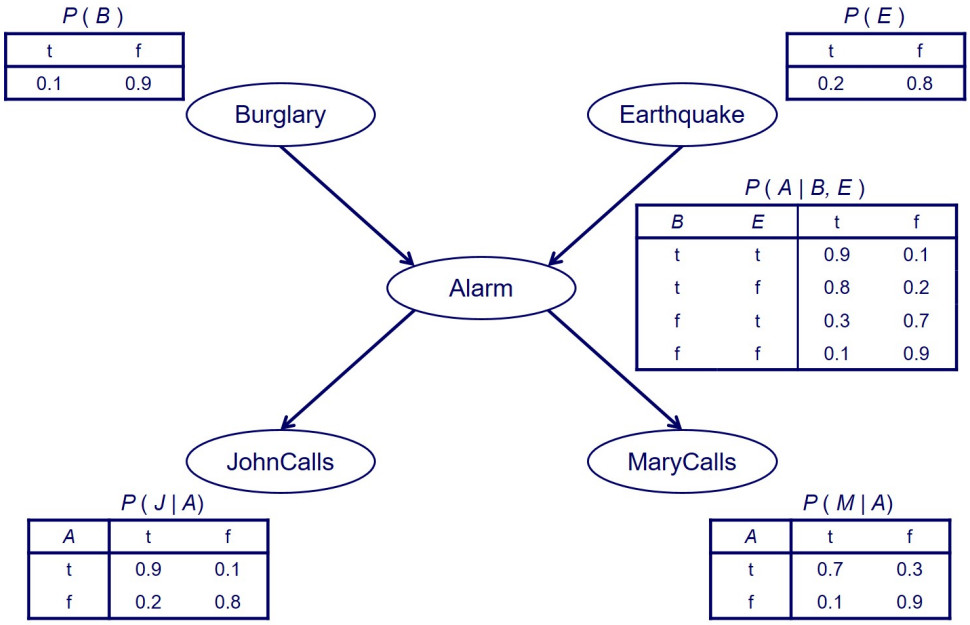


Figure 7: A Bayesian Network example.

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.

Given the probabilities:

- $P(B) = [t : 0.1, f : 0.9]$
- $P(E) = [t : 0.2, f : 0.8]$
- $P(A \mid B, E)$ is given by the table.
- $P(J \mid A) = [t : 0.9, f : 0.2]$
- $P(M \mid A) = [t : 0.7, f : 0.1]$

For $P(B = t \mid E = f, J = t, M = t)$:

Let's calculate joint probabilities:

$$P(B = t, E = f, J = t, M = t) = P(B = t) \times P(E = f) \times P(A = t \mid B = t, E = f) \times P(J = t \mid A = t) \times P(M = t \mid A = t$$
$$= 0.1 \times 0.8 \times 0.8 \times 0.9 \times 0.7$$

$$P(B = f, E = f, J = t, M = t) = P(B = f) \times P(E = f) \times P(A = t \mid B = f, E = f) \times P(J = t \mid A = t) \times P(M = t \mid A =$$
$$= 0.9 \times 0.8 \times 0.1 \times 0.9 \times 0.7$$

For $P(B = t \mid E = t, J = t, M = t)$:

Let's calculate joint probabilities:

$$P(B = t, E = t, J = t, M = t) = P(B = t) \times P(E = t) \times P(A = t \mid B = t, E = t) \times P(J = t \mid A = t) \times P(M = t \mid A = t)$$
$$= 0.1 \times 0.2 \times 0.9 \times 0.9 \times 0.7$$

$$P(B=f,E=t,J=t,M=t) = P(B=f) \times P(E=t) \times P(A=t \mid B=f,E=t) \times P(J=t \mid A=t) \times P(M=t \mid A=t$$
$$= 0.9 \times 0.2 \times 0.3 \times 0.9 \times 0.7$$

Now, for $E=f, J=t, M=t$:

$$P(B=t,E=f,J=t,M=t) = 0.1 \times 0.8 \times 0.8 \times 0.9 \times 0.7 = 0.04032$$
$$P(B=f,E=f,J=t,M=t) = 0.9 \times 0.8 \times 0.1 \times 0.9 \times 0.7 = 0.04032$$
$$P(E=f,J=t,M=t) = 0.04032 + 0.04032 = 0.08064$$
$$P(B=t \mid E=f,J=t,M=t) = \frac{0.04032}{0.08064} = 0.5$$

For $E=t, J=t, M=t$:

$$P(B=t,E=t,J=t,M=t) = 0.1 \times 0.2 \times 0.9 \times 0.9 \times 0.7 = 0.01008$$
$$P(B=f,E=t,J=t,M=t) = 0.9 \times 0.2 \times 0.3 \times 0.9 \times 0.7 = 0.03744$$
$$P(E=t,J=t,M=t) = 0.01008 + 0.03744 = 0.04752$$
$$P(B=t \mid E=t,J=t,M=t) = \frac{0.01008}{0.04752} \approx 0.212$$

Therefore, the complete solution for the given Bayesian network is:

- $P(B=t \mid E=f,J=t,M=t) = 0.5$
- $P(B=t \mid E=t,J=t,M=t) \approx 0.212$

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)
2. Compute the mutual information $I(X,Z)$ based on the frequencies observed in the data. (5 pts)
3. Compute the mutual information $I(Z,Y)$ based on the frequencies observed in the data. (5 pts)
4. Which undirected edges will be selected by the Chow-Liu algorithm as the maximum spanning tree? (5 pts)
5. Root your tree at node X, assign directions to the selected edges. (5 pts)

To compute mutual information, we can use the formula:

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

where $P(X = x, Y = y)$ is the joint probability of X and Y, and $P(X = x)$ and $P(Y = y)$ are the marginal probabilities.
Let's calculate the mutual information for $I(X, Y)$, $I(X, Z)$, and $I(Z, Y)$ using the given data.

1. Calculate Joint Probabilities for X and Y:
   - $P(X = T, Y = T) = \frac{\text{Count of } (T,T)}{\text{Total Count}} = \frac{36+4}{100} = \frac{40}{100} = 0.4$
   - $P(X = T, Y = F) = \frac{\text{Count of } (T,F)}{\text{Total Count}} = \frac{2+8}{100} = \frac{10}{100} = 0.1$
   - $P(X = F, Y = T) = \frac{\text{Count of } (F,T)}{\text{Total Count}} = \frac{9+1}{100} = \frac{10}{100} = 0.1$
   - $P(X = F, Y = F) = \frac{\text{Count of } (F,F)}{\text{Total Count}} = \frac{8+32}{100} = \frac{40}{100} = 0.4$
2. Calculate Marginal Probabilities for X and Y:
   - $P(X = T) = \frac{\text{Count of } X=T}{\text{Total Count}} = \frac{36+4+2+8}{100} = \frac{50}{100} = 0.5$
   - $P(X = F) = \frac{\text{Count of } X=F}{\text{Total Count}} = \frac{9+1+8+32}{100} = \frac{50}{100} = 0.5$
   - $P(Y = T) = \frac{\text{Count of } Y=T}{\text{Total Count}} = \frac{36+4+9+1}{100} = \frac{50}{100} = 0.5$
   - $P(Y = F) = \frac{\text{Count of } Y=F}{\text{Total Count}} = \frac{2+8+8+32}{100} = \frac{50}{100} = 0.5$
3. Compute Mutual Information $I(X, Y)$:

$$I(X,Y) = 0.4 \log_2 \left(\frac{0.4}{0.5 \times 0.5} \right) + 0.1 \log_2 \left(\frac{0.1}{0.5 \times 0.5} \right) + 0.1 \log_2 \left(\frac{0.1}{0.5 \times 0.5} \right) + 0.4 \log_2 \left(\frac{0.4}{0.5 \times 0.5} \right)$$

$$I(X,Y) = 0.4 \times 0.6781 + 0.1 \times (-1.3219) + 0.1 \times (-1.3219) + 0.4 \times 0.6781$$

$$I(X,Y) = 0.278$$

4. Compute $I(X, Z)$:

$$I(X,Z) = 0.14$$

5. Compute $I(Z, Y)$:

$$I(Z,Y) = 0.405$$

.

6. The Chow-Liu algorithm constructs a maximum spanning tree for a graphical model by selecting edges based on the mutual information between variables. In this context, it will select edges that connect the pairs of variables with the highest mutual information, under the constraint that no cycles are formed.
   - $I(X, Y) \approx 0.278$
   - $I(X, Z) \approx 0.140$
   - $I(Z, Y) \approx 0.405$
7. Rooting the tree at node X and assigning directions to the edges involves converting the undirected edges of the maximum spanning tree into a directed acyclic graph (DAG). The directed edges in the tree are:
   (a) $X \to Y$
   (b) $Y \to Z$

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*.