HOMEWORK 6

Rishideep Reddy Rallabandi
9084949099

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.

Github link: https://github.com/Rishideep08/CS760-HW6

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)

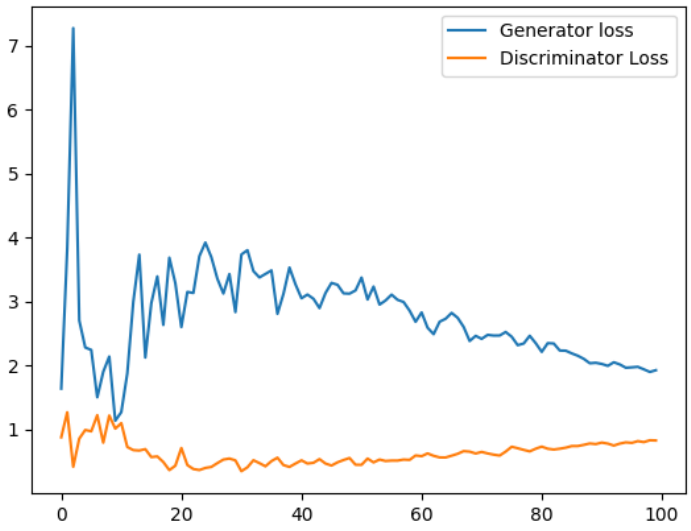


Figure 1: Learning curve

(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)

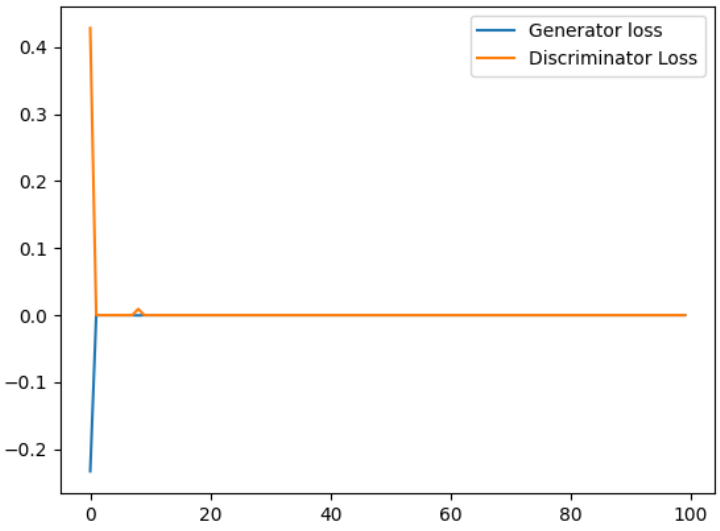
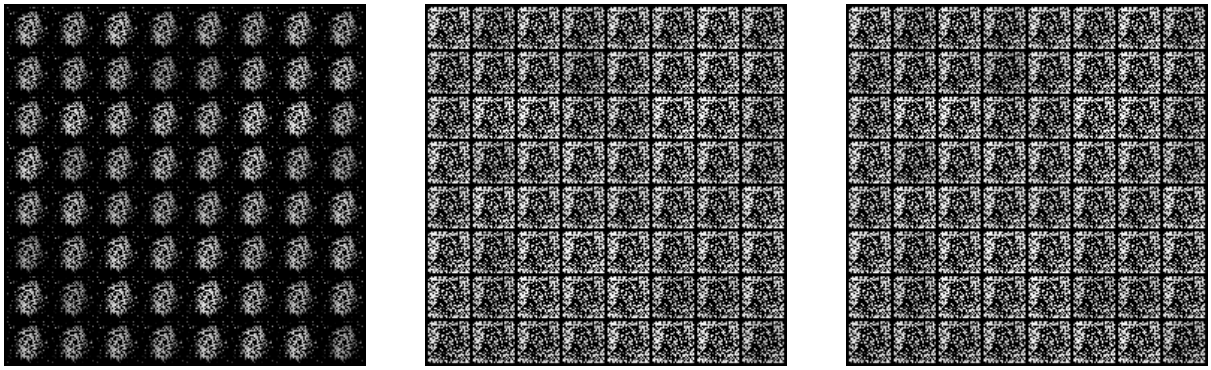


Figure 3: Learning curve



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

Figure 4: Generated images by G

The model does not learn anything and collapses. The change made to update the generator doesn't lead to realistic images. This happens because the new method for adjusting the generator's parameters becomes less effective as the discriminator gets better at distinguishing real from fake images.

Although in theory, the settings that make $\frac{1}{n_z}\sum_{i=1}^{n_z}\log(1-D(G(z^{(i)})))$ as small as possible are the same as those that make $\frac{1}{n_z}\sum_{i=1}^{n_z}\log D(G(z^{(i)}))$ as large as possible, when we look at how the gradients change during the learning process, the first values decrease so much that they practically disappear. On the other hand, the latter values don't disappear entirely but might behave in an unstable way.

(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)

Label Smoothing: It involves using values such as 0.95 for true labels and 0.05 for false labels instead of exclusively using 1s and 0s. This approach prevents the model from being overly confident in its predictions and aids in better generalization.

Adding Noise: Incorporating noise serves as a regularization technique that enhances the generator's performance.

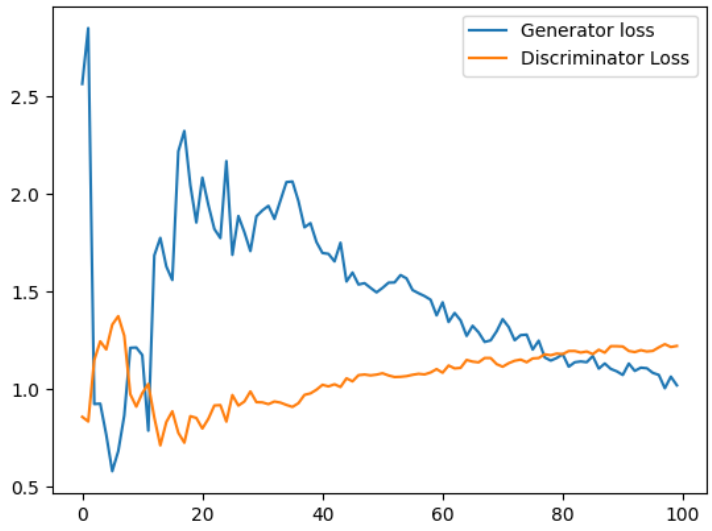
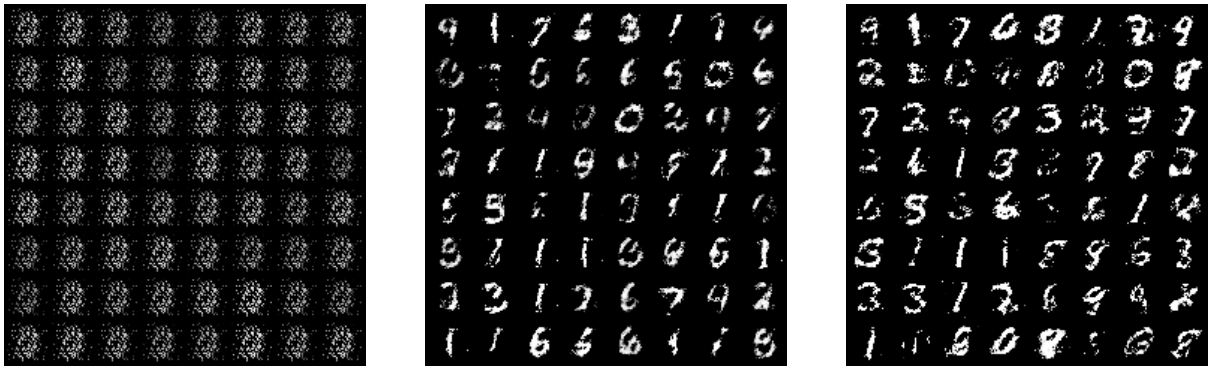


Figure 5: Learning curve



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

Figure 6: Generated images by G

2 Directed Graphical Model [25 points]

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

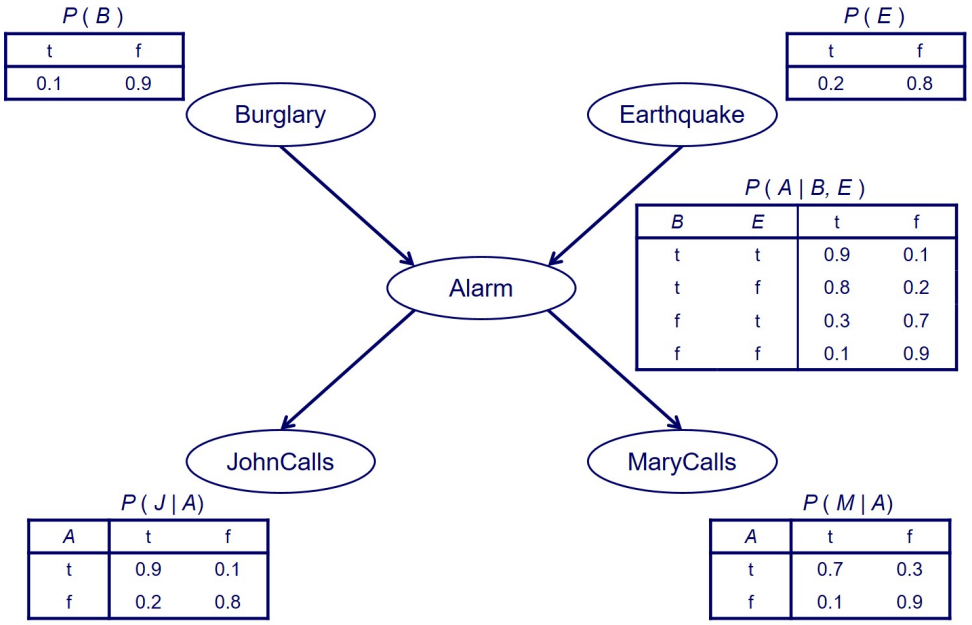


Figure 7: A Bayesian Network example.

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.

$$
\begin{aligned}
a) P(B = t \mid E = f, J = t, M = t) &= \frac{P(B = t, E = f, J = t, M = t)}{P(E = f, J = t, M = t)} \\
P(E = f, J = t, M = t) &= P(B = t, E = f, J = t, M = t) + P(B = f, E = f, J = t, M = t) \\
P(B = t, E = f, J = t, M = t) &= \sum_A P(B = t, E = f, A, J = t, M = t) \\
&= \sum_A P(B = t) \times P(E = f) \times P(A \mid B = t, E = f) \\
&\quad \times P(J = t \mid A) \times P(M = t \mid A) \\
&= 0.1 \times 0.8 \times 0.8 \times 0.9 \times 0.7 \\
&\quad + 0.1 \times 0.8 \times 0.2 \times 0.2 \times 0.1 \\
&= 0.04064 \\
P(B = f, E = f, J = t, M = t) &= \sum_A P(B = f, E = f, A, J = t, M = t) \\
&= \sum_A P(B = f) \times P(E = f) \times P(A \mid B = f, E = f) \\
&\quad \times P(J = t \mid A) \times P(M = t \mid A) \\
&= 0.9 \times 0.8 \times 0.1 \times 0.9 \times 0.7 \\
&\quad + 0.9 \times 0.8 \times 0.9 \times 0.2 \times 0.1 \\
&= 0.05832 \\
P(B = t \mid E = f, J = t, M = t) &= \frac{0.04064}{0.04064 + 0.05832} = 0.41067098
\end{aligned}
$$

$$
\begin{aligned}
b) P(B=t \mid E=t, J=t, M=t) &= \frac{P(B=t, E=t, J=t, M=t)}{P(E=t, J=t, M=t)} \\
P(E=t, J=t, M=t) &= P(B=t, E=t, J=t, M=t) + P(B=f, E=t, J=t, M=t) \\
P(B=t, E=t, J=t, M=t) &= \sum_A P(B=t, E=t, A, J=t, M=t) \\
&= \sum_A P(B=t) \times P(E=t) \times P(A \mid B=t, E=t) \\
&\quad \times P(J=t \mid A) \times P(M=t \mid A) \\
&= 0.1 \times 0.2 \times 0.9 \times 0.9 \times 0.7 \\
&\quad + 0.1 \times 0.2 \times 0.1 \times 0.2 \times 0.1 \\
&= 0.01138 \\
P(B=f, E=t, J=t, M=t) &= \sum_A P(B=f, E=t, A, J=t, M=t) \\
&= \sum_A P(B=f) \times P(E=t) \times P(A \mid B=f, E=t) \\
&\quad \times P(J=t \mid A) \times P(M=t \mid A) \\
&= 0.9 \times 0.2 \times 0.3 \times 0.9 \times 0.7 \\
&\quad + 0.9 \times 0.2 \times 0.7 \times 0.2 \times 0.1 \\
&= 0.03654 \\
P(B=t \mid E=t, J=t, M=t) &= \frac{0.01138}{0.01138 + 0.03654} = 0.23747913
\end{aligned}
$$

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)

$$
\begin{aligned}
I(X, Y) &= \sum_{x \in \{t,f\}} \sum_{y \in \{t,f\}} P(x, y) \log_2 \frac{P(x, y)}{P(x)P(y)} \\
&= 0.4 \log_2 \frac{0.4}{0.5 \times 0.5} + 0.1 \log_2 \frac{0.1}{0.5 \times 0.5} \\
&\quad + 0.1 \log_2 \frac{0.1}{0.5 \times 0.5} + 0.4 \log_2 \frac{0.4}{0.5 \times 0.5} \\
&\approx 0.2781
\end{aligned}
$$

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

$$
\begin{aligned}
I(X,Z) &= \sum_{x\in\{t,f\}}\sum_{z\in\{t,f\}} P(x,z)\log_2\frac{P(x,z)}{P(x)P(z)} \\
&= 0.38\log_2\frac{0.38}{0.5\times 0.55} + 0.12\log_2\frac{0.12}{0.5\times 0.45} \\
&\quad + 0.17\log_2\frac{0.17}{0.5\times 0.55} + 0.33\log_2\frac{0.33}{0.5\times 0.45} \\
&\approx 0.1328
\end{aligned}
$$

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

$$
\begin{aligned}
I(Z,Y) &= \sum_{z\in\{t,f\}}\sum_{y\in\{t,f\}} P(z,y)\log_2\frac{P(z,y)}{P(z)P(y)} \\
&= 0.45\log_2\frac{0.45}{0.55\times 0.5} + 0.1\log_2\frac{0.1}{0.55\times 0.5} \\
&\quad + 0.05\log_2\frac{0.05}{0.45\times 0.5} + 0.4\log_2\frac{0.4}{0.45\times 0.5} \\
&\approx 0.3973
\end{aligned}
$$

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

$$
\begin{aligned}
I(X,Y) &= 0.2781 \\
I(X,Z) &= 0.1328 \\
I(Z,Y) &= 0.3973
\end{aligned}
$$

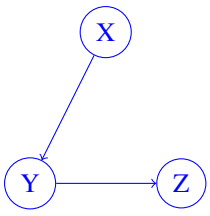
The Chow-Liu algorithm first selects the edge with the largest weight (in this case, the edge weights are the mutual information values), which will be (Z,Y).

The second largest edge will be (X,Y).

We can't add (X,Z) because it will create a cycle in the graph.

So, the maximum spanning tree generated is $G = \big(V := \{X,Y,Z\}, E := \{(Z,Y),(X,Y)\}\big)$.

5. Root your tree at node X, assign directions to the selected edges. (5 pts)



Probability distributions:

$P(X) =$

T	F
0.5	0.5

$P(Y \mid X) =$

X	T	F
T	0.8	0.2
F	0.2	0.8

$P(Z \mid Y) =$

Y	T	F
T	0.9	0.1
F	0.2	0.8

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*.