HOMEWORK 5

Devansh Goenka
908 335 1354

Instructions:

- Please submit your answers in a single pdf file and your code in a zip file. pdf preferably made using latex. No need to submit latex code. See piazza post @114 for some recommendations on how to write the answers.
- Submit code for programming exercises. Though we provide a base code with python (jupyter notebook), you can use any programming language you like as long as you use the same model and dataset.
- Submit all the material on time.

1 Implementation: GAN (40 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:

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

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.

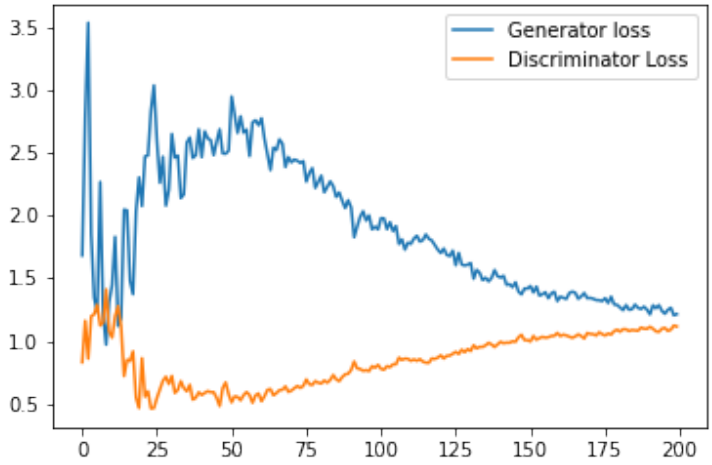
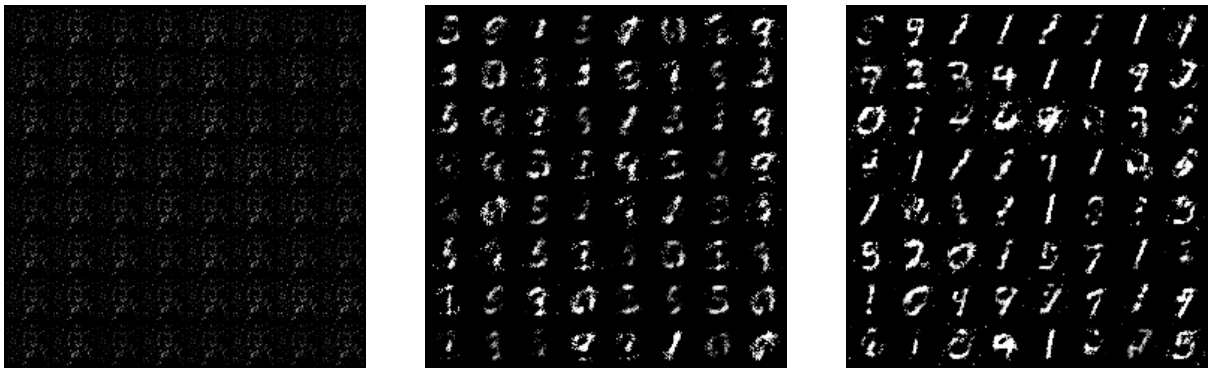


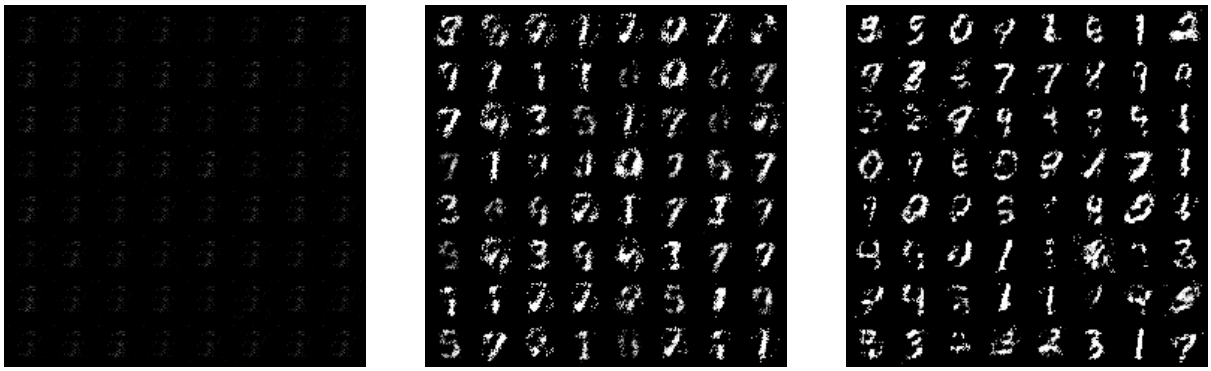
Figure 1: Learning curve



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

Figure 2: Generated images by G

After implementing and training the GAN using the above described steps, here are the results:



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

Figure 3: Generated images by G

And here is the learning curve for 100 epochs:

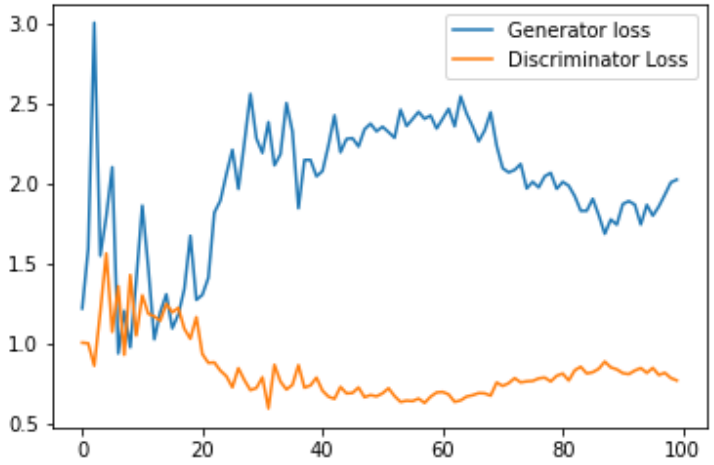


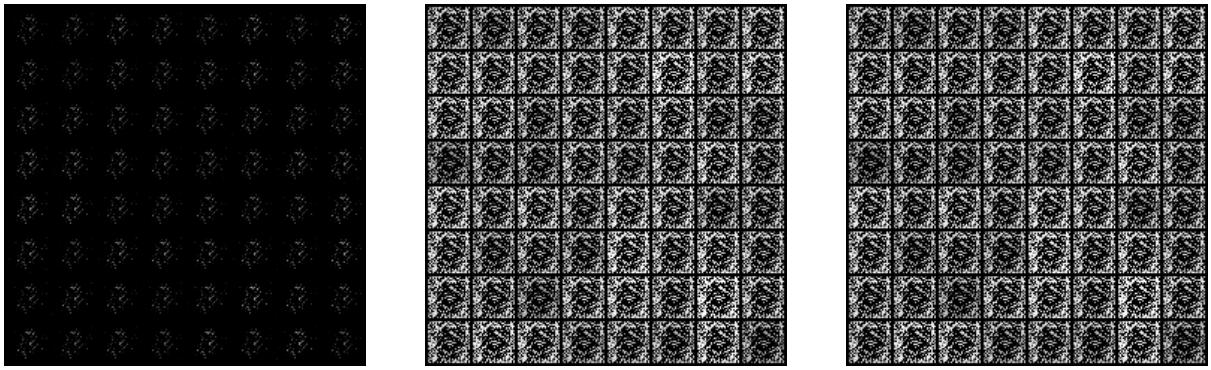
Figure 4: Learning curve for 1(a)

(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. (10 pts)

We replace the generator update rule with the one described above [essentially taking the negative of the BCE loss and performing gradient descent instead of ascent], and we see the following results:



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

Figure 5: Generated images by G with original generator update rule

And here is the learning curve for 100 epochs:

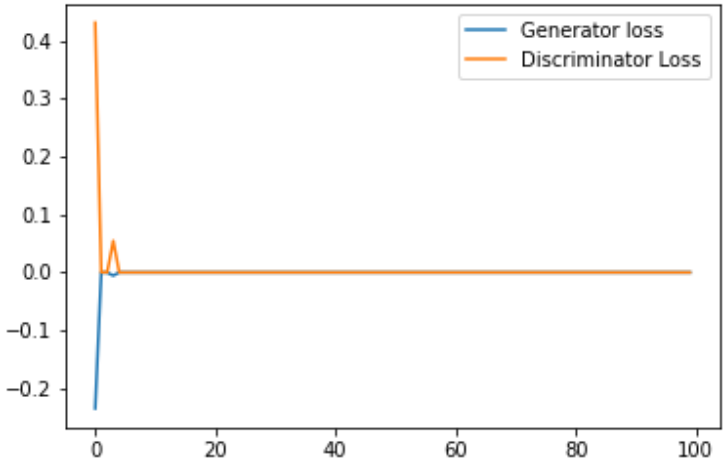


Figure 6: Learning curve for 1(b)

As we can see, this clearly fails to replicate the original distribution even though the loss seems to have converged. This is because, as suggested in the original paper by Goodfellow et. al., the original training objective for the Generator fails to provide strong gradients in the early training phases. This makes the discriminator reject samples with very high confidence, and eventually, training fails.

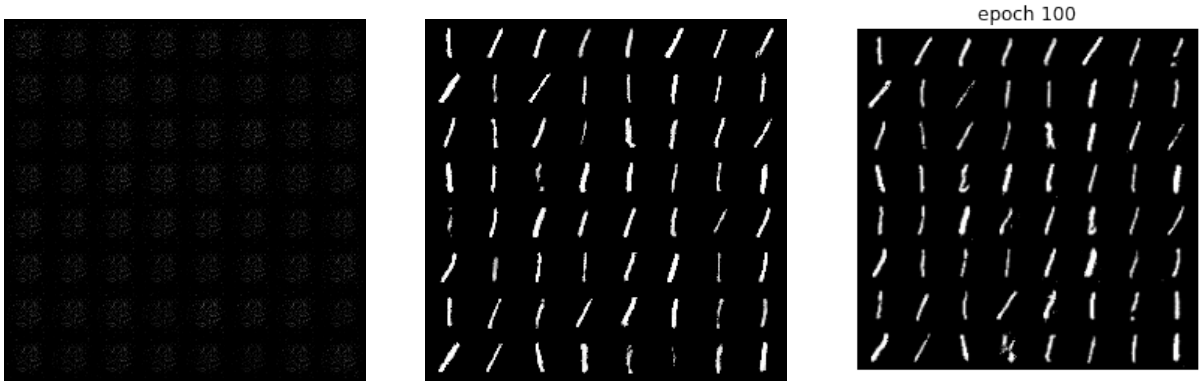
(c) Except the method that we used in (a), how can we improve training for GAN? Implement that and report learning curves and generated images in epoch 1, 50, 100. (10 pts)

We tried 2 different approaches to improve GAN training, and here are the results:

Approach 1: Based on Goodfellow et. al., we tried modifiying the number of steps 'k' in the original algorithm which signifies the number of steps the discriminator is trained for before updating the generator.

On trying for different values of 'k', the outcomes were similar, and although the training did converge faster, another common issue called "mode collapse" was encountered in this, where the generator just learnt how to byass the discriminator by producing only 1 class of images.

The generated images and training curve are as follows: (k=5 here)



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

Figure 7: Generated images by G with alternate 'k'

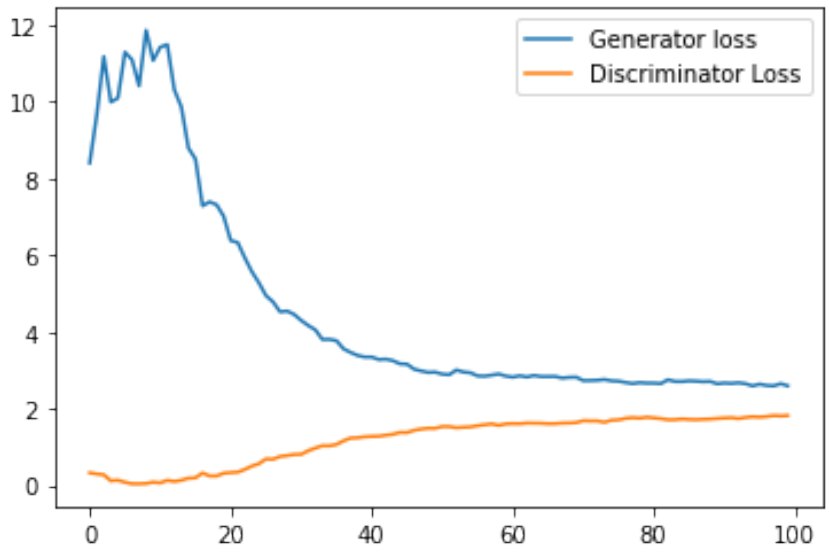


Figure 8: Learning curve for 1(c) with alternate 'k'

Approach 2: We also tried another approach called Least Squares GAN where we change the original loss functions a least squares loss as given by:

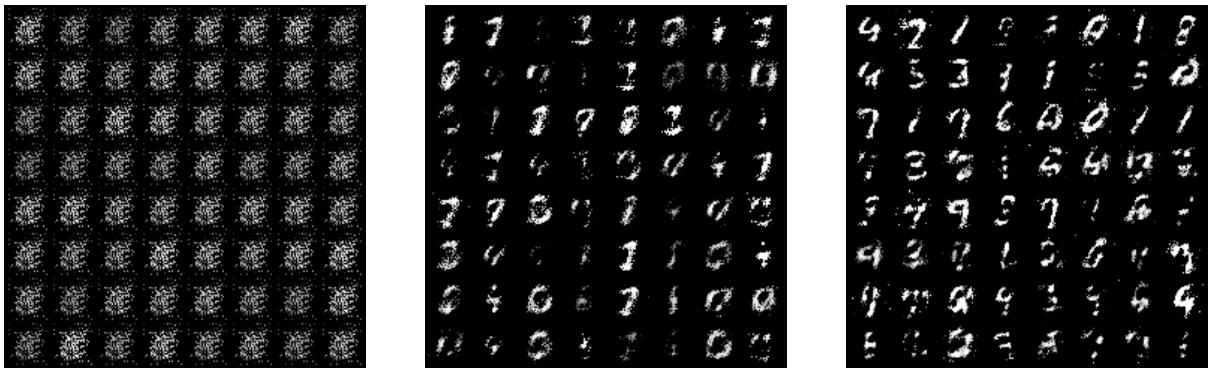
Ascending the stochastic gradient for Discriminator:

$$\nabla_{\theta_d}\Big(\frac{1}{m}\sum_{i=1}^{m}[D(x^{(i)})-1]^2+\frac{1}{n_z}\log\sum_{i=1}^{n_z}[D(G(z^{(i)}))]^2)\Big)$$

Ascending the stochastic gradient for Generator:

$$\nabla_{\theta_g}\frac{1}{n_z}\sum_{i=1}^{n_z}[D(G(z^{(i)}))-1]^2\Big)$$

We observed comparable performance to the original method used in 1(a), although convergence was slower. We hypothesize that if we train using this method for longer, we might get better results. Here are the results:



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

Figure 9: Generated images by G with LSGAN

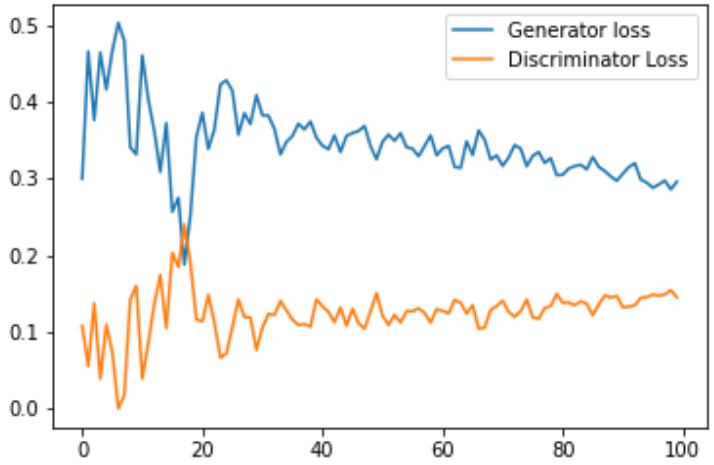


Figure 10: Learning curve for 1(c) with LSGAN

2 Ridge regression [15 pts]

Derive the closed-form solution in matrix form for the ridge regression problem:

$$\min_{\beta} \left(\frac{1}{n} \sum_{i=1}^{n} (z_i^\top \beta - y_i)^2 \right) + \lambda \|\beta\|_A^2$$

where

$$\|\beta\|_A^2 := \beta^\top A \beta$$

and

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

This A matrix has the effect of NOT regularizing the bias β_0, which is standard practice in ridge regression. Note: Derive the closed-form solution, do not blindly copy lecture notes.

Here, we can see that $\beta \in \mathbb{R}^3$.
Thus, we can re-write the above problem as:

$$\min_{\beta} (y - Z\beta)^2 + \lambda \|\beta\|_A^2$$

where $Z \in \mathbb{R}^{nx3}$ and $y \in \mathbb{R}^n$. After performing some linear algebra:

$$\min_{\beta} (y - Z\beta)^\top (y - Z\beta) + \lambda \beta^\top A \beta$$

$$\min_{\beta} y^\top y - \beta^\top Z^\top y - y^\top Z\beta + \beta^\top Z^\top Z\beta + \lambda \beta^\top A \beta$$

$$\min_{\beta} y^\top y - 2\beta^\top Z^\top y + \beta^\top (Z^\top Z + \lambda A)\beta$$

Now, to find the closed form solution, we differentiate this w.r.t to β and set it to 0.

$$\frac{\partial (y^\top y - 2\beta^\top Z^\top y + \beta^\top (Z^\top Z + \lambda A)\beta)}{\partial \beta} = 0$$

$$-2Z^\top y + 2(Z^\top Z + \lambda A)\beta = 0$$
$$(Z^\top Z + \lambda A)\beta = Z^\top y$$
$$\beta = (Z^\top Z + \lambda A)^{-1} Z^\top y$$

This is the closed form solution for the above ridge regression problem.

3 Review the change of variable in probability density function [25 pts]

In Flow based generative model, we have seen $p_\theta(x) = p(f_\theta(x))|\frac{\partial f_\theta(x)}{\partial x}|$. As a hands-on (fixed parameter) example, consider the following setting.

Let X and Y be independent, standard normal random variables. Consider the transformation $U = X + Y$ and $V = X - Y$. In the notation used above, $U = g_1(X, Y)$ where $g_1(X, Y)$ where $g_1(x, y) = x + y$ and $V = g_2(X, Y)$ where $g_2(x, y) = x - y$. The joint pdf of X and Y is $f_{X,Y} = (2\pi)^{-1}exp(-x^2/2)exp(-y^2/2), -\infty < x < \infty, -\infty < y < \infty$. Then, we can determine u, v values by x, y, i.e. $\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$.

(a) (5 pts) Compute Jacobian matrix

$$J = \begin{bmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \end{bmatrix}$$

(5 pts)

We see that $u = x + y$ and $v = x - y$. Conversely, we can say that:
$u = x + (x - v) = 2x - v$
or,
$x = \frac{u + v}{2}$
Similarly, $y = \frac{u - v}{2}$
Therefore:

$$J = \begin{bmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \end{bmatrix} = \begin{bmatrix} \frac{\partial (u+v)/2}{\partial u} & \frac{\partial (u+v)/2}{\partial v} \\ \frac{\partial (u-v)/2}{\partial u} & \frac{\partial (u-v)/2}{\partial v} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{bmatrix}$$

(b) (Forward) Show that the joint pdf of U, V is

$$f_{U,V}(u, v) = \big(\frac{1}{\sqrt{2\pi}\sqrt{2}}exp(-u^2/4)\big)\big(\frac{1}{\sqrt{2\pi}\sqrt{2}}exp(-v^2/4)\big)$$

(10 pts)

(Hint: $f_{U,V}(u, v) = f_{X,Y}(?, ?)|det(J)|$)

We know that:

$$f_{U,V}(u, v) = f_{X,Y}(x, y)|J|$$

And we have established that $x = \frac{u + v}{2}$ and $y = \frac{u - v}{2}$.
Therefore:

$$f_{U,V}(u, v) = f_{X,Y}(\frac{u + v}{2}, \frac{u - v}{2})|J|$$

We know that:

$$f_{X,Y}(x, y) = (2\pi)^{-1}exp(-x^2/2)exp(-y^2/2)$$

Substituting the values:

$$f_{U,V}(u, v) = (2\pi)^{-1}exp(-(\frac{u + v}{2})^2/2)exp(-(\frac{u - v}{2})^2/2)|J|$$

$$f_{U,V}(u, v) = (2\pi)^{-1}exp(-(\frac{u + v}{2})^2/2)exp(-(\frac{u - v}{2})^2/2)|J|$$

$$f_{U,V}(u, v) = (2\pi)^{-1}exp(\frac{-u^2 - v^2 - 2uv}{8})exp(\frac{-u^2 - v^2 - 2uv}{8})|J|$$

$$f_{U,V}(u,v) = (2\pi)^{-1} exp(\frac{-u^2}{4}) exp(\frac{-v^2}{4}) |J|$$

Now, $|det(J)| = |-\frac{1}{2}| = \frac{1}{2}$
Thus,

$$f_{U,V}(u,v) = \frac{1}{2\pi} exp(\frac{-u^2}{4}) exp(\frac{-v^2}{4}) \frac{1}{2} = \frac{1}{4\pi} exp(\frac{-u^2}{4}) exp(\frac{-v^2}{4})$$

Thus, we can re-write it as:

$$f_{U,V}(u,v) = \big(\frac{1}{\sqrt{2\pi}\sqrt{2}} exp(-u^2/4)\big)\big(\frac{1}{\sqrt{2\pi}\sqrt{2}} exp(-v^2/4)\big)$$

(c) (Inverse) Check whether the following equation holds or not.

$$f_{X,Y}(x,y) = f_{U,V}(x+y, x-y)|det(J)|^{-1}$$

(10 pts)

Using the derived joint PDF of U, V above we can plug in the values into the equation:

$$f_{U,V}(x+y, x-y) = \big(\frac{1}{\sqrt{2\pi}\sqrt{2}} exp(-(x+y)^2/4)\big)\big(\frac{1}{\sqrt{2\pi}\sqrt{2}} exp(-(x-y)^2/4)\big)$$

$$f_{U,V}(x+y, x-y) = \frac{1}{4\pi} exp(\frac{-(x^2+y^2+2xy)}{4}) exp(\frac{-(x^2+y^2-2xy)}{4})$$

$$f_{U,V}(x+y, x-y) = \frac{1}{4\pi} exp(\frac{-x^2-y^2-2xy}{4}) exp(\frac{-x^2-y^2+2xy}{4})$$

$$f_{U,V}(x+y, x-y) = \frac{1}{4\pi} exp(\frac{-x^2}{2}) exp(\frac{-y^2}{2})$$

And, $|det(J)|^{-1} = (1/2)^{-1} = 2$.
Therefore:

$$f_{U,V}(x+y, x-y)|J|^{-1} = \frac{1}{2\pi} exp(\frac{-x^2}{2}) exp(\frac{-y^2}{2}) = f_{X,Y}(x,y)$$

Therefore, the inverse relation holds true.

4 Directed Graphical Model [20 points]

Consider the directed graphical model (aka Bayesian network) in Figure 11.

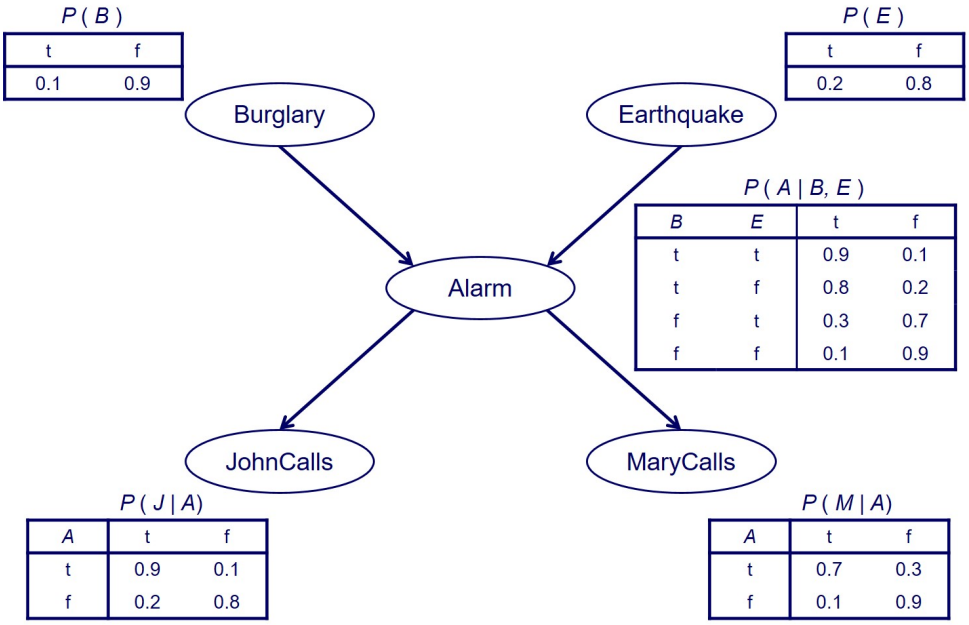


Figure 11: A Bayesian Network example.

Compute $P(B = t \mid E = f, J = t, M = t)$ and $P(B = t \mid E = t, J = t, M = t)$. 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.
We need to perform inference over this Bayes net via enumeration.

To find $P(B = t \mid E = f, J = t, M = t)$, we know that by the properties of conditional probability:

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

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

Therefore, first we find the joint probabilities $P(B = t, E = f, J = t, M = t)$ and $P(B = f, E = f, J = t, M = t)$ from the given Bayes net.

$$P(B = t, E = f, J = t, M = t) = \sum_{A=t,f} P(B = t)P(E = f)P(A \mid B = t, E = f)P(J = t \mid A)P(M = t \mid A)$$

$$= P(B = t)P(E = f) \sum_{A=t,f} P(A \mid B = t, E = f)P(J = t \mid A)P(M = t \mid A)$$

$$= (0.1) * (0.8) * [(0.8 * 0.9 * 0.7) + (0.2 * 0.2 * 0.1)] = 0.04064$$

Similarly, we find $P(B = f, E = f, J = t, M = t)$ as:

$$P(B = f, E = f, J = t, M = t) = \sum_{A=t,f} P(B = f)P(E = f)P(A \mid B = f, E = f)P(J = t \mid A)P(M = t \mid A)$$

$$= (0.9) * (0.8) * [(0.1 * 0.7 * 0.9) + (0.9 * 0.1 * 0.2)] = 0.05832$$

Therefore,

$$\frac{P(B = t, E = f, J = t, M = t)}{P(B = t, E = f, J = t, M = t) + P(B = f, E = f, J = t, M = t)} = \frac{0.04064}{0.04064 + 0.05832} = 0.4106$$

Hence, $P(B = t \mid E = f, J = t, M = t) = 0.4106$.
We perform the same steps to find $P(B = t \mid E = t, J = t, M = t)$ as well:

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

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

Now,

$$P(B = t, E = t, J = t, M = t) = \sum_{A=t,f} P(B = t)P(E = t)P(A \mid B = t, E = t)P(J = t \mid A)P(M = t \mid A)$$

$$= (0.1) * (0.2) * [(0.9 * 0.9 * 0.7) + (0.1 * 0.2 * 0.1)] = 0.01138$$

And,

$$P(B = f, E = t, J = t, M = t) = \sum_{A=t,f} P(B = f)P(E = t)P(A \mid B = f, E = t)P(J = t \mid A)P(M = t \mid A)$$

$$= (0.9 * 0.2) * [(0.3 * 0.9 * 0.7) + (0.7 * 0.2 * 0.1)] = 0.03654$$

Therefore,

$$P(B = t \mid E = t, J = t, M = t) = \frac{0.01138}{0.01138 + 0.03654} = 0.23747$$

Hence, $P(B = t \mid E = t, J = t, M = t) = 0.23747$

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*.