ECE 8803

PGM - ASSIGNMENT 3

JEET KIRAN PAWANI
GTid - 903397407

1 1.1 Stationary distribution $\pi(x)$ on chain (T) is given by

$$\pi(X = x') = \sum_{x \in Val(X)} \pi(X = x)\, T(x \to x') \rightarrow (1)$$

Gibbs sampling chain is given by

$$T_i\left((x_{-i}, x_i) \to (x_{-i}, x_i')\right) = P(x_i' / x_{-i}) \rightarrow (2)$$

To prove that $P(x/e)$ is a distribution of gibbs chain is also stationary distribution.

To Prove

$$P(x'/e) = \sum_{x_i \in Val(x_i)} P(x/e)\, T_c\left((x_{-i}, x_i) \to (x_{-i}, x_i')/e\right)$$

$$\Rightarrow P(x'/e) = \sum_{x_i \in Val(x_i)} P(x_1 \ldots x_k / e)\, \underbrace{T_i\left((x_i, x_i) \to (x_{-i}, x_i')/e\right)}_{P(x_i' / x_{-i}, e)}$$

Taking RHS :- Using Bayes Rule

$$RHS = \sum_{x \in Var(x_i)} \frac{P(x_1 \ldots x_k, e)}{P(e)} \underbrace{P(x_i' / x_{-i}, e)}_{\text{From (2)}}$$

$$= \sum_{x \in Var(x_i)} \frac{P(x_1 \ldots x_k, e)}{P(e)} \frac{P(x_i', x_{-i}, e)}{P(x_{-i}, e)}$$

$$= \sum_{x \in Val(x_i)} \frac{P(x_1 \ldots x_k, e)}{P(x_{-i}, e)} \frac{P(x_i', x_{-i}, e)}{P(e)}$$

Taking x_i terms outside summation

$$\Rightarrow RHS = \underbrace{\frac{P(x_i', x_{-i}/e)}{\text{Bayes rule}}}_{\text{Bayes rule}} \underbrace{\sum_{x \in Val(x_i)} P(x_i / e), x_{-i})}_{1}$$

$\Rightarrow$ RHS = $P(x'/e)$

This means LHS is a stationary distribution

1.2 $p(x) \propto \exp(\sin(x))$ $\quad -\pi \le x \le \pi$

$q(x) = N(x/0, \sigma^2)$

0 mean

variance $= \sigma^2$

↳ distribution $= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-x^2/2\sigma^2}$

$$\frac{p^d(s)}{q(x)} \le m$$

$$\frac{e^{\sin(x)}}{e^{-x^2/2\sigma^2}} \le m$$

$$\frac{1}{\sqrt{2\pi\sigma^2}}$$

$$m \ge e^{\sin(x)} e^{x^2/2\sigma^2} \sqrt{2\pi\sigma^2}$$

x ranges from $-\pi$ to π

$\sin(x)$ is max at $x = \pi/2$ $\sin(\pi/2) = 1$

$\Rightarrow$ x takes value π^2

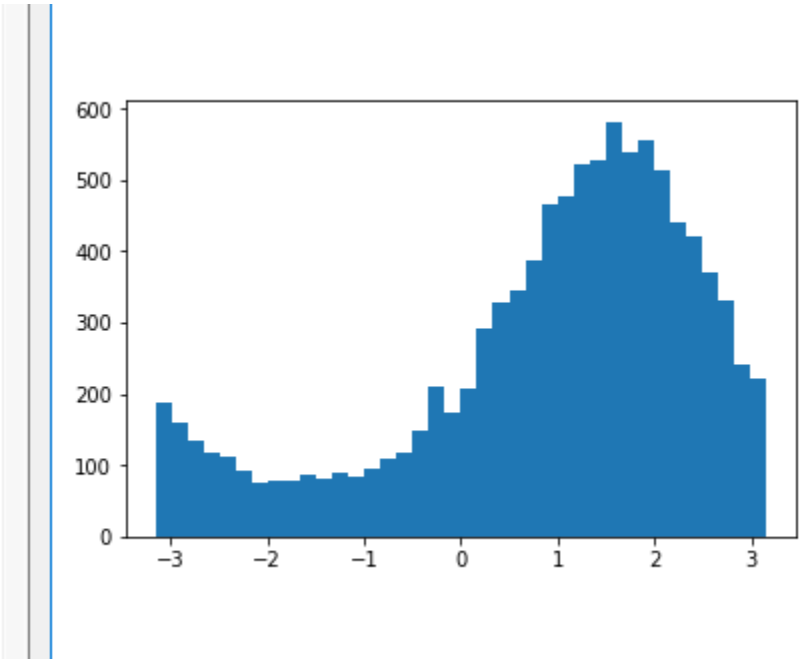
$\Rightarrow m = e^{\sin(x)} e^{\pi^2/2\sigma^2} \sqrt{2\pi\sigma^2} = \frac{p(x)}{q(x)}$

$$\Rightarrow \boxed{m = e^{1 + \pi^2/2\sigma^2} \sqrt{2\pi\sigma^2}}$$

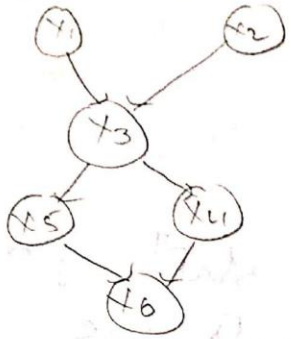
Question 1.2)

Spyder compiler was used.

After performing and implementing the rejection sampling the following plot was obtained.



1.3 $p(x_1, \ldots, x_6) = p(x_1)\, p(x_2)\, p(x_3/x_2, x_1)$ (2)
$p(x_4/x_3)\; p(x_5/x_3)\; p(x_6/x_4, x_5)$



Given x_5 is fixed in state $\underline{x_5}$

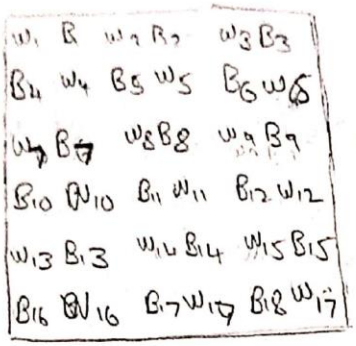
$$p(x_1, x_2, x_3, x_4, x_6 / x_5 = x_5) = \frac{p(x_1, x_2, x_3, x_4, x_6, x_5)}{p(x_5)}$$

$\Rightarrow$ Substituting the above eqn

$$p'(x_1, x_2, x_3, x_4, x_5) = \frac{p(x_1)\, p(x_2)\, p(x_3/x_2, x_1)\, p(x_4/x_3)\, p(x_5/x_3)\, p(x_6/x_4, x_5)}{\sum_{x_1, x_2, x_3, x_4, x_6} p(x_1)\, p(x_2)\, p(x_3/x_2, x_1)\, p(x_4/x_3)\, p(x_5/x_3)\, p(x_6/x_4, x_5)}$$

We can perform forward sampling on this easily but we need to add one more step in the process, i.e, rejection sampling as well. Since forward sampling generates all possible samples without seeing the evidence. So, lets say in the process when samples are generated, if the evidence given does not match with evidence in the sample generated, we simply reject that sample. Thus, we will have the final output samples with all evidence matching using updated forward sampling.

1.4



This is a 8×8 checkerboard diagram. W's is white box and B is black box. Hence, all these form a markov blanket.

Lets start from w_1, we can see that its neighbours are B_1 and B_4.
$\Rightarrow$ w_i's are independent of each other.

Also, if we notice black blocks also have white boxes as neighbours which form a markov blanket. $\Rightarrow$ B_i's are independent of each other.

Thus
$$p(b_1, b_2 \ldots | w_1, w_2 \ldots) = p(b_1 | w_1 \ldots) \; p(b_2 | w_1 w_2 \ldots) \ldots$$

Since all black boxes are independent of each other, we can perform gibbs sampling on black boxes by conditioning on its neighbour/markov blanketed white boxes. As, this is independent, sampling can be performed.

(3)

Similar procedure can be done to samples white bosses as well. We can generate samples by gibbs sampling, by condition white bosses on their markov blanket.

$\Rightarrow p(w_1 \dots / b_1, b_2 \dots) = p(w_1 / b_1 \dots b_n)$

$p(w_2 / b_1 \dots b_n)$

Independent gibbs sampling can be done to generate various samples of white bosses and black bosses

2] 2.1] In this question, we are given 2 teams of 20 players each and a fixed goalkeeper for both. In each game, the goalkeeper is fixed and 10 players are selected and played by each team. The dataset contains all players and 20 such combinations and also a predict of each team winning. (6)

Eqn gives

$$P(\text{Aces beats Bruisers} \mid t_a, t_b, a, b) = \sigma\left[\sum_{i=1}^{10} (a_{t_i}^{a} - b_{t_i}^{b})\right]$$

The question is to sample abilities of each of the player of both teams given the dataset. The abilities are in range $-2, -1, 0, 1, 2$ where -2 being worst and 2 being best.

Here we consider both teams to have uniform priors.

We need to estimate

$$P(a, b \mid D)$$

by expanding on bayes rule

$$P(a, b \mid D) = P(D \mid a, b)\, P(a, b)$$

a, b are uniform priors. So, we ignore that.

$$\Rightarrow P(a, b \mid D) \propto P(D \mid a, b)$$

$\Rightarrow P(a,b/D) \propto P(t_a, t_b, \text{Aces beat bruises}/a,b)$

which can be further written as

$P(a,b/D) \propto P(\text{Aces beat bruises}/t_a, t_b, a, b)$

So, we are considering ability of players to be independent, the following equation as been implemented on the data to get factors

$$\prod_{j=1}^{20} \left(\sigma\left(\sum_{i=1}^{10}(a_{ti}^{a} - b_{ti}^{b})\right)\right)^{A_j} \left[1 - \sigma\left(\sum_{i=1}^{10}(a_{ti}^{a} - b_{ti}^{b})\right)\right]^{1-A_j}$$

A_j is aces winning.

Gibbs sampling is being done and samples are generated on all possible ability combination.

For first part of que, we generate around 5000-8000 samples with 25000 burnin and 10 subsampling and average over all samples is found and 10 best players of each team is computed.

In part 2, using the above samples, we have fixed team Bruises, so we use brute force of selecting all combinations of $^{20}C_{10}$ for team Aces and predicting the best team to put for winning the game.

Question 2.1)

I ran this code on the Pycharm Compiler.

```
A best players
[11 12 13  2 10 20  7 16 14  1]
B best players
[13 16 12  7  6 20  2 10 14  5]
 BEst team A to play against 1 to 10 players of Team B is -
[ 1  2  7 10 11 12 13 14 16 20]
```

Number of samples generated was 8000.
Burn in period was 25000
Subsampling was – 10.

I have saved all the 8000 samples in the pickle file samples_q1.txt.

The best A team obtained after fining mean of all samples was (Screenshot of Output attached)
Team Aces – [11 , 12 ,13, 2 ,10,20,7,16,14,1]
Best 10 players of Team Bruisers are-
Team Bruisers – [13,16,12,7,6,20,2,10,14,5]

WE used the 8000 samples generated above and ran abilities of players of Team B 1 to 10 across all combinations of team A (ie 20C10 possibilities) and bes toutcome came out to be -
Best team Aces to play against Team Bruisers one to 10 players is –
[1,2,7,10,11,12,13,14,16,20]

2.2] This question tells us to predict if the person has a disease or not given the symptoms. (7)

The dataset contains D diseases data

di gives 1 or 0 for 1 ... D diseases.

then there is a symptom list. etc.

The disease symptom network is given by

$$p(s_1 \dots s_s, d_1 \dots d_D) = \prod_{j=1}^{S} p(s_j/d) \prod_{i=1}^{D} p(d_i)$$

$d = (d_1 \dots d_D)^T$ and

$$p(s_j = 1/d) = \sigma(w_j^T d + b_j)$$

To find

$[p(d_1 = 1/s) \dots p(d_D = 1/s)]$

Using bayes rule

$$p(d_i = 1/s) \propto \sum_{d} p(s_j/d_j)\, p(d_j)$$

$$p(s_j/d) = \prod_{j=1}^{S} [\sigma(w_j^T d + b_j)]^{s_j} \left[1 - \sigma(w_j^T d + b_j)\right]^{1-s_j}$$

also $p(d) = \prod_{i=1}^{D} (p(d_i))^{d_i} (1 - p(d_i))^{1-d_i}$

So, sampling has to be done from

$p(d_i = 1/s)$

The equation turns out to be -

$$p(d_i = 1 \mid s) = \prod_{j=1}^{S} \sigma(w_j^T d + b_j)^{s_i} \left(1 - \sigma(w_j^T d + b_j)\right)^{1-s_i} \times \prod_{i=1}^{D} p(d_i)^{d_i} \left(1 - p(d_i)\right)^{1-d_i}$$

Thus, I will be performing gibbs sampling on the above data for 1x40 raw vector of diseases and finally predicting the avg probability for every disease given the symptom list. I will generate over 8000 samples with burnin of 25000 samples and 15 subsampling, so the correlatedness in the data disappears. When all samples are generated, near over every sample is being found, which further corresponds to the prob of disease given the symptoms. The code and readings have been separately added.

Question 2.2)

Pycharm compiler was used.
The number of samples generated were 8000.
Burn in period was 25000
Sub sampling was 10.

I have explained the deriving part in the handwritten part.
The outputs obtained were –

The output of p(d1=1/s) are
Prob of disease 1 is 0.00225
Prob of disease 2 is 0.9975
Prob of disease 3 is 0.02
Prob of disease 4 is 1.0
Prob of disease 5 is 0.64725
Prob of disease 6 is 0.009375
Prob of disease 7 is 0.014625
Prob of disease 8 is 0.000375
Prob of disease 9 is 0.006375
Prob of disease 10 is 1.0
Prob of disease 11 is 0.003875
Prob of disease 12 is 1.0
Prob of disease 13 is 1.0
Prob of disease 14 is 1.0
Prob of disease 15 is 0.989125
Prob of disease 16 is 0.97075
Prob of disease 17 is 0.988875
Prob of disease 18 is 0.906375
Prob of disease 19 is 0.999875
Prob of disease 20 is 0.99425
Prob of disease 21 is 0.084625
Prob of disease 22 is 0.754
Prob of disease 23 is 1.0
Prob of disease 24 is 0.995
Prob of disease 25 is 0.004375
Prob of disease 26 is 0.999375
Prob of disease 27 is 0.01525
Prob of disease 28 is 0.0
Prob of disease 29 is 0.9985
Prob of disease 30 is 0.0
Prob of disease 31 is 0.0
Prob of disease 32 is 0.0925
Prob of disease 33 is 0.0115
Prob of disease 34 is 1.0
Prob of disease 35 is 0.0
Prob of disease 36 is 0.00275
Prob of disease 37 is 0.994875

Prob of disease 38 is 0.00125
Prob of disease 39 is 0.0
Prob of disease 40 is 0.000625
Prob of disease 41 is 0.993875
Prob of disease 42 is 0.99875
Prob of disease 43 is 1.0
Prob of disease 44 is 0.999875
Prob of disease 45 is 0.00025
Prob of disease 46 is 0.0175
Prob of disease 47 is 0.996
Prob of disease 48 is 0.9955
Prob of disease 49 is 0.0
Prob of disease 50 is 0.999875

2.3) $p(x/\mu_1, \mu_2) \propto \frac{1}{2} \exp\left(\frac{-1}{2}(x-\mu_1)^2\right) + \frac{1}{2}\exp\left(\frac{-1}{2}(x-\mu_2)^2\right)$

$p(\mu_1, \mu_2) \propto \exp\left(\frac{-\mu_1^2}{2\times 100}\right) \exp\left(\frac{-\mu_2^2}{2\times 100}\right)$

a) 100 samples need to be generated from 2 different gaussians. We use the multivariate - normal() in python to get these samples. We sample 50 each from both gaussians and store and use them for next part.

b) Now we need to do metropolis hastings to sample data from $p(\mu_1, \mu_2 / x_1 \ldots x_{100})$

using bayes rule.

$$p(\mu_1, \mu_2 / x_1 \ldots x_{100}) \propto p(x_1 \ldots x_{100} / \mu_1, \mu_2)\, p(\mu_1, \mu_2)$$

$$= \left(\frac{1}{2}\exp\left(\frac{-1}{2}(x-\mu_1)^2\right) + \frac{1}{2}\exp\left(\frac{-1}{2}(x-\mu_2)^2\right)\right) \left[\exp\left(\frac{-\mu_1^2}{2\times 100}\right) \exp\left(\frac{-\mu_2^2}{2\times 100}\right)\right]$$

The initial μ is set to 0 and the sampled μ is generated from $q(\mu_1', \mu_2' / \mu_1, \mu_2) = N(\mu_1, \sigma^2)\, N(\mu_2, \sigma^2)$ which is a combination of 2 distributions

The sample generated is accepted only when the following condition of metropolis hasting is satisfied

i.e $u < \min\left(1, \frac{p(\mu_1^*, \mu_2^*)\, Q(\mu_1', \mu_2' / \mu_1^*, \mu_2^*)}{p(\mu_1, \mu_2)\, Q(\mu_1^*, \mu_2^* / \mu_1, \mu_2)}\right)$

On satisfaction of above condition the sample is accepted.

c) Gibbs sampling

Since both means are from continuous distribution, direct sampling by gibbs is not possible so, a latent variable has to be introduced here which makes our approach to generate samples a little easier

As seen from question, we can say that the input is IID sequence. So, all z can be directly set we have to now find $P(z_i / x_i, \mu)$

$$\Rightarrow P(z_i / x_i, \mu) \propto N(x_i, \mu_2; \sigma^2) \times p(z)$$

$$P(z_i = 0 / x_i, \mu_1) \sim \exp\left(\frac{-1}{2}(x - \mu_1)^2\right)$$

$$P(z_i = 1 / x_i, \mu_2) \sim \exp\left(\frac{-1}{2}(x - \mu_2)^2\right)$$

Thus z are sampled directly.

Using these z's we sample new mean and covariance

$$P(\mu_k / \mu_{-k}, z, x) = P(\mu_k / z, x)$$

$$= P(x / \mu z) P(\mu_2)$$

$$= N(\mu_2, 1^2) \; N(\hat{\mu}_2, 10^2)$$

$$= N(\hat{\mu}_2, \hat{\lambda}_k)$$

$$\hat{x}_k \qquad \hat{\lambda}_k = (n_k/\sigma^2 + 1/\lambda^2)$$

$$\hat{\mu}_k = \frac{(n_k/\sigma^2)}{(n_k/\sigma^2 + 1/\lambda^2)}$$

$$n_k = \sum_{i=1}^{N} z_i^k$$

$$\hat{x}_k = \frac{\sum z_i^k x_i}{n_k}$$

Using the above equations, new u is sampled. Again we set burnin of 10,000 and save the next 1000 samples. The same experiment is repeated 6 times. (9)

Problem 2.3)

Both pycharm and Spyder environments were used.
In this problem we were told first generate samples randomly 50-50 from 2 distributions and the shuffle them .

Then we had to generate samples on this given data and estimate the mean by sampling. 2 methods were used they were metropolis hastings and Gibbs sampling.

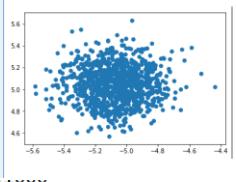
Metropolis hastings was done for variance of 0.5 and 5. Both estimated means corrected in the 6 runs of each only that the acceptance rate varied a lot. The 0.5 variance has a higher acceptance rate than the 5 variance data.

Average means table has been made for both cases and I have attached the plots as well.

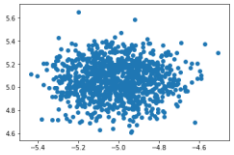
Metropolis Hastings Method
Var =0.5
Burnin samples – 10000

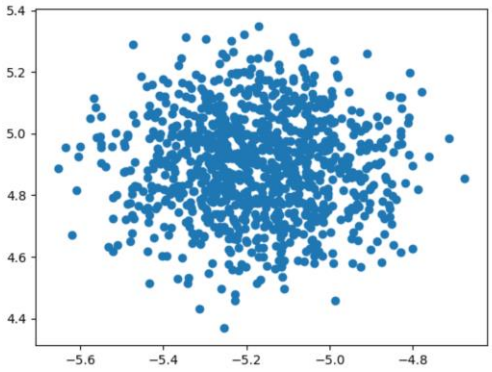
Variance= 0.5, burnin samples = 10000	Mean 1	Mean2
Run 1	-5.075	5.04
Run2	-5.011	5.064
Run3	-5.1766	4.899
Run 4	-5.032	5.0664
Run 5	-4.8787	4.7826
Run 6	-5.068	5.1394

```
 Acceptance Rate  0.01185227326601242
-5.075179421190729 5.048991378206445
```

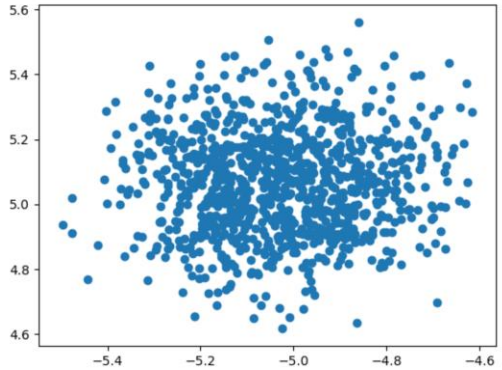


```
 Acceptance Rate  0.011792591893772332
-5.011894198438898 5.0647352756664965
```



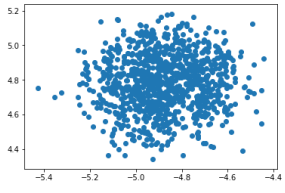



```
1000
 Acceptance Rate  0.01175143367490834
-5.176605392358712 4.899050900906937
```

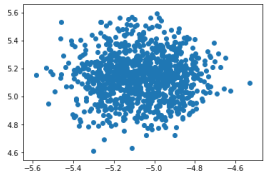



```
1000
 Acceptance Rate  0.011804702993672679
-5.032138737355659 5.066470290428847
```

```
1000
 Acceptance Rate  0.011868027533823879
-4.878793453657634 4.782635707285175
```



```
1000
 Acceptance Rate  0.12125621438098702
-5.068232962924761 5.139419403951286
```



```
In [59]:
```

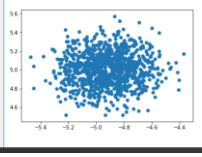
Variance = 5

Burnin 10000/100, Variance = 5	Mean 1	Mean 2
Run1	-4.9205	5.0033
Run2	-4.7044	4.9051
Run3	-4.9540	4.8925
Run4	-4.691	5.093
Run5	-5.023	4.898
Run6	-5.2217	5.0419

Sigma = 5

Burn in period – 10000

```
1000
 Acceptance Rate  0.0001478084951750133 8
-4.920595936375756 5.003362117735161
```

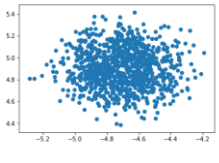


Spyder (Python 3.7)

Sigma = 5

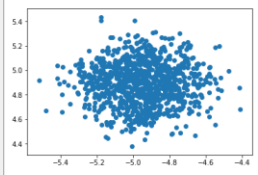
Burn in = 100

```
1000
 Acceptance Rate  0.0014565730041672554
-4.704482225814749 4.9051398269364705
```



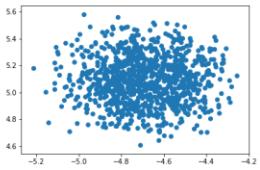
Sigma = 5, Burn in period =100

```
1000
 Acceptance Rate  0.00144109038663014
-4.954057708895428 4.892588977979972
```



Sigma = 5, Subsampling 100

```
 Acceptance Rate  0.001522183541847109
-4.691305930233809 5.0931031670561815
```

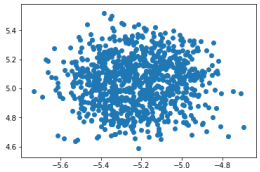


```
In [54]:
```

```
1000
 Acceptance Rate  0.0013961137776884265
-5.023981858073465 4.898907885900504
```

Plot was similar to above.

```
 Acceptance Rate  0.0014468639224480937
-5.221798455867988 5.041944754512047
```



Gibbs Sampling

Samples 1000 , Burnin = 25000, sub=10	Mean 1	Mean 2
Run1	5.0122	-4.9879
Run2	5.0133	-5.126
Run3	4.918	-4.912
Run4	5.1366	-5.1763
Run5	5.022	-5.187
Run6	4.9363	-5.027

Here WE have used number of samples as 1000, with burnin samples as 10000 and subsampling as 10. We get the average mean as almost that of distribution.

I have attached the graph as well as made a table of average means.

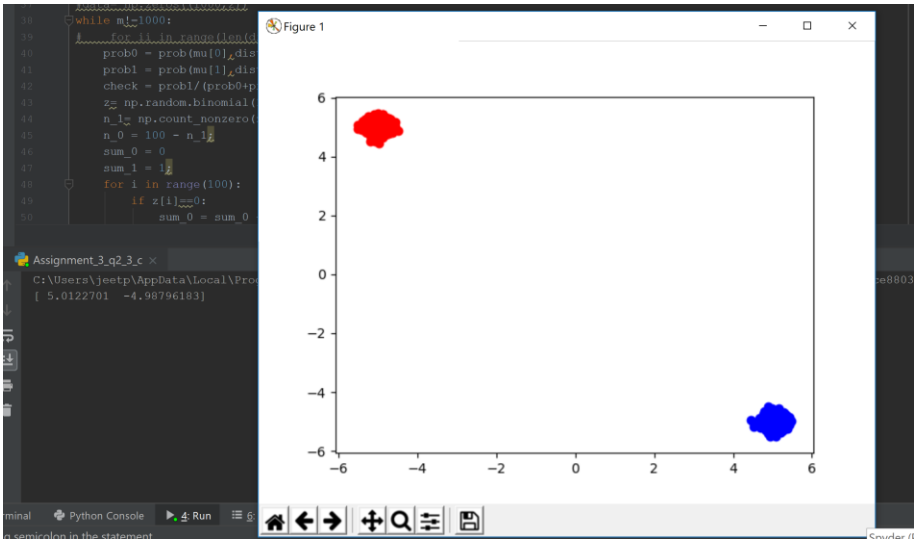
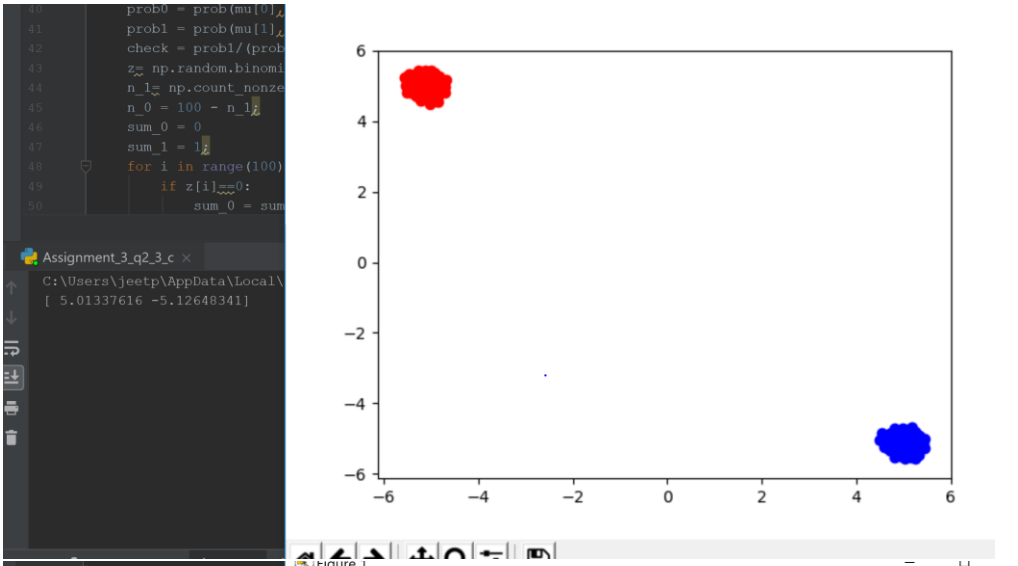


Figure 1
Assignment_3_q2_3_c
[5.0122701 -4.98796183]
Python Console
Run


Assignment_3_q2_3_c
[5.01337616 -5.12648341]

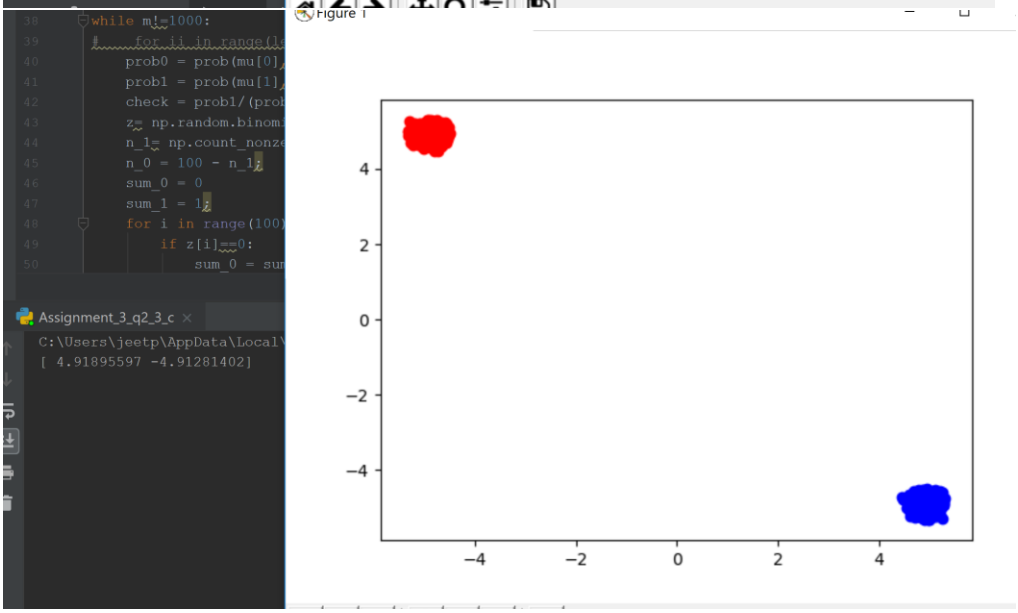
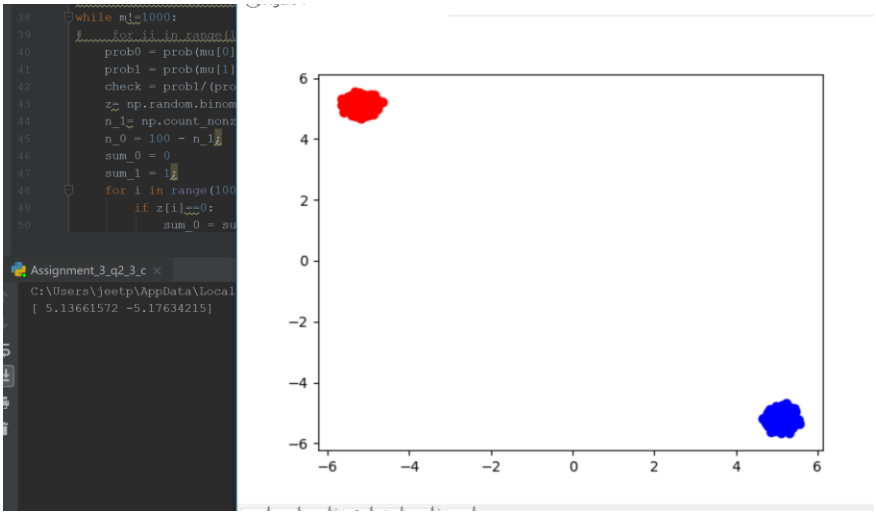
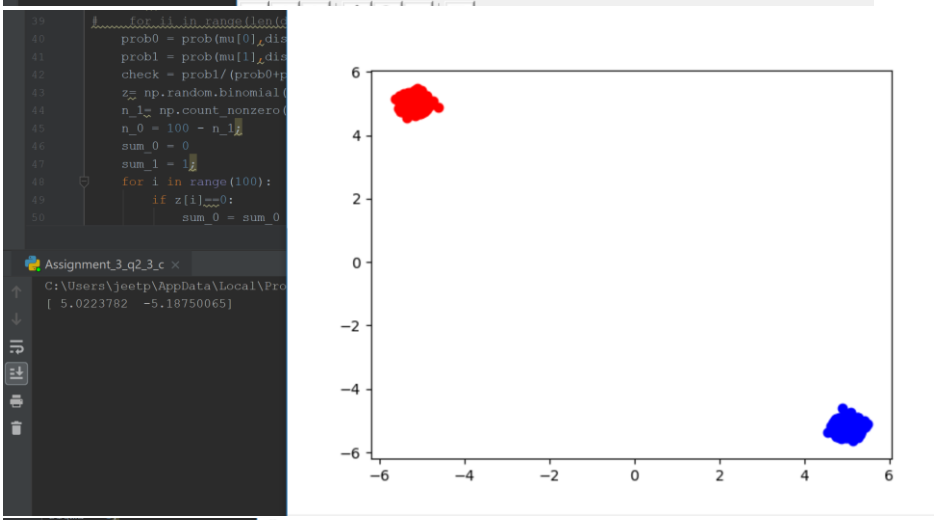


Figure 1
Assignment_3_q2_3_c
[4.91895597 -4.91281402]


Assignment_3_q2_3_c
C:\Users\jeetp\AppData\Local
[5.13661572 -5.17634215]


Assignment_3_q2_3_c
C:\Users\jeetp\AppData\Local\Pro
[5.0223782 -5.18750065]

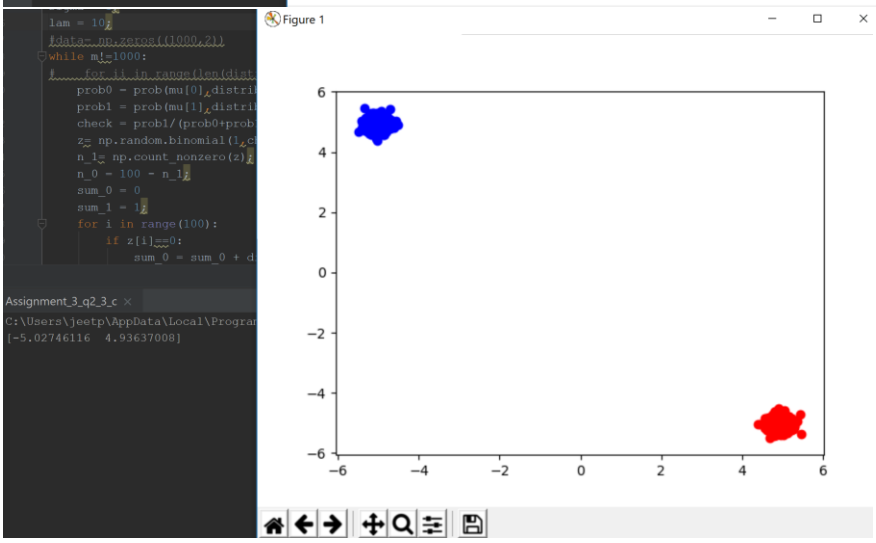
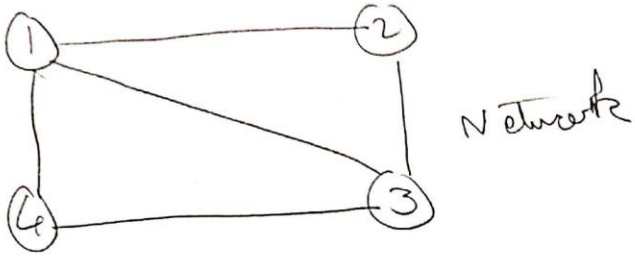


Figure 1
Assignment_3_q2_3_c
C:\Users\jeetp\AppData\Local\Progra
[-5.02746116 4.93637008]

3) 3.1

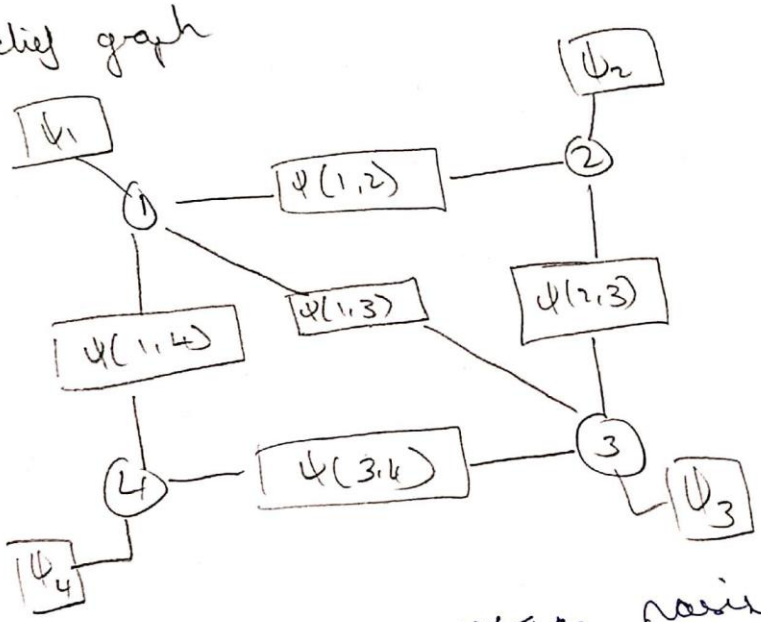
① Loopy Belief propagation.

Given 2×2 lattice



Network

Loopy belief graph



we implement the message passing algorithm here in the loopy graph as per the equations in the notes. Every nodes belief is updated according to the messages being passed.

The algo is implemented as follows -

$$b_i(x_i) \propto f_i(x_i) \prod_{a \in \text{neighbours}} \text{message}_{a \to i}(x_i) \quad -(1)$$

belief — variable — a's factors

$$\text{message}_{i \to a}(x_i) = \prod_{c \in \text{neighbors}(i)/a} \text{message}_{c \to i}(x_i) \to (1)$$ (10)

$$\text{message}_{a \to i}(x_i) = \sum_{x_a \setminus x_i} f_a(x_a) \prod_{j \in \text{neighbors}(a) \setminus i} \text{message}_{j \to a}(x_i)$$

The algorithm goes as follows

(1) initially from a node all its neighbours are scanned through and the message from factor to nodes is multiplied to beliefs

(2) Then Factor to node message is updated with the probability table given in ~~graph~~ question

(3) The node to factor message is further updated by multiplying messages from other neighbours factor to node messages.

These steps are repeated till convergence is reached.

~~A class fg based function is~~

A factor class maintained to store all factors and their messages

Convergence occurs in 6 iterations itself and mean error is almost 0.00185

Problem 3.1)

Pycharm compiler was used.

a)
In this question the Loopy belief message propagation was implemented and we have found out the mean error from the enumeration method.
This method is clearly better than the mean field method, as the error is almost 0.00185.
The algo followed has been written down

b) In this part mean field algo is implemented

A q value is calculated for all particular nodes for both states 0 and 1.

we set all states as unprocessed first.

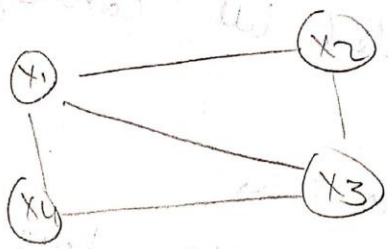
we pick an unprocessed variable

and update Q_i

$$Q_i(x_i) = \frac{1}{Z_i} \exp\left(\sum_{D_j : x_i \in D_j} Eq\ (\ln \psi(D_j)) \right)$$

→ node x_i is set as processed

If q_i is changed then we set its neighbours as unprocessed. This converges to local minima



message at x_i

$$Q(x_1) = \sum_{x_2} Q(x_2) \ln(\psi(x_1, x_2)) + \sum_{x_3} Q(x_3) \ln(\psi(x_1, x_3)) + \sum_{x_4} Q(x_4) \ln(\psi(x_1, x_4))$$

The average mean error comes to 0.101594 (11)

c) In this all marginals are calculated for all possibilities of all the states - Then marginal of every state being 0 or 1 is calculated and hence a final value is obtained.

d) Values obtained c is compared with part a and b and mean errors are calculated.

mean error loopy belief = 0.00185

mean error mean field = 0.10159

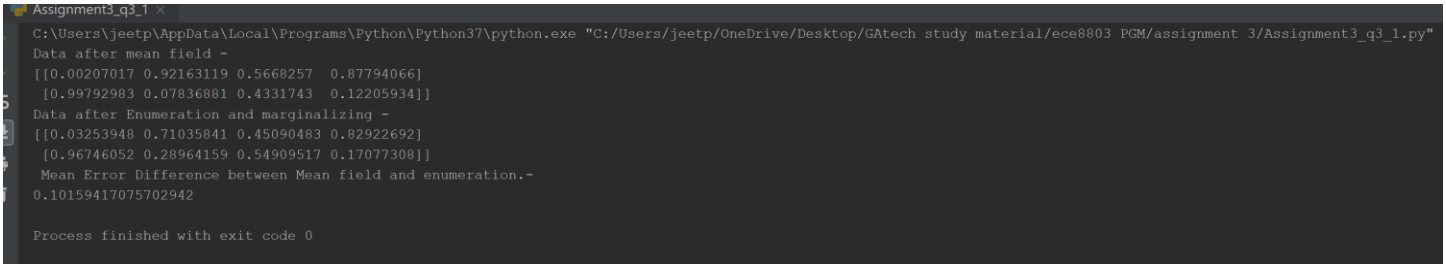
Loopy belief is a better way to implement the code

b)

In this question the mean field message propagation was implemented and we have found out the mean error from the enumeration method.

This method is clearly better than the mean field method, as the error is almost 0.10159

The algo followed has been written down.



3.2] ① $q(x) = \frac{1}{Z(\phi)} e^{\phi^T g(x)}$

To approximate $q(x)$ to a distribution $p(x)$ using KL divergence $KL(p/q)$

To show, $\langle g(x) \rangle_{p(x)} = \langle g(x) \rangle_{q(x)}$

$KL(p||q) = \int p(x) \ln\left(\frac{p(x)}{q(x)}\right)$ from notes

$= \int p(x) \ln(p(x)) - \int p(x) \ln(q(x))$

This is minimum (ie KL divergence between $p(x)$ and $q(x)$ when they are optimal

$\Rightarrow KL(p||q) = \langle \ln p(x) \rangle_{p(x)} - \langle \ln(q(x)) \rangle_{p(x)} \rightarrow ①$

Given $q(x) = \frac{1}{Z(\phi)} e^{\phi^T g(x)}$

$\Rightarrow \ln q(x) = \ln\left(\frac{1}{Z(\phi)} e^{\phi^T g(x)}\right) = \ln\left(\frac{1}{Z(\phi)}\right) + \ln\left(e^{\phi^T g(x)}\right)$

$= \phi^T g(x) - \ln(Z(\phi))$

$\rightarrow ②$

Sub ② in ①

$\Rightarrow KL(p||q) = \langle \ln p(x) \rangle_{p(x)} - \langle \phi^T g(x) - \ln(Z(\phi)) \rangle_{p(x)}$

$= \langle \ln p(x) \rangle_{p(x)} - \langle \phi^T g(x) \rangle_{p(x)}$

$+ \langle \ln(Z(\phi)) \rangle_{p(x)}$

(4)

we know, at minima $\frac{d\,KL\,divergence}{d\phi} = 0$

$$\Rightarrow \frac{d\,KL(p||q)}{d\phi} = 0$$

$$\frac{d}{d\phi}(KL(p||q)) = \frac{d}{d\phi}\left(\langle \ln(p(x))\rangle_{p(x)} - \langle \phi^T g(x)\rangle + \langle \ln(z(\phi)\rangle_{p(x)}\right]$$

$$\langle \ln z(\phi)\rangle_{p(x)} = \int p(x) \underbrace{\ln z(\phi)}_{\text{constant}} dx$$

$$= \ln z(\phi) \underbrace{\int p(x)\,dx}_{1} = \ln z(\phi)$$

$$\frac{d}{d\phi} KL(p||q) = \frac{d}{d\phi}\left(\langle \ln p(x)\rangle_{p(x)} - \langle \phi^T g(x)\rangle_{p(x)} + \ln(z(\phi))\right)$$

$$= 0 - \frac{d}{d\phi}\left[\int \phi^T g(x)\, p(x)\,dx\right] + \frac{1}{z(\phi)}\frac{dz(\phi)}{d\phi}$$

$$0 = 0 - \int g(x)\,p(x)\,dx + \frac{1}{z(\phi)}\frac{dz(\phi)}{d\phi}$$

$$\Rightarrow \langle g(x)\rangle_{p(x)} - \frac{1}{z(\phi)}\frac{dz(\phi)}{d\phi} = 0$$

we know $z(\phi) = \int e^{\phi^T g(x)}\,dx$

$$\frac{d\,z(\phi)}{d\phi} = \int g(x)\, e^{\phi^T g(x)}\,dx$$

$$\langle g(x)\rangle_{p(x)} - \frac{1}{z(\phi)}\frac{dz(\phi)}{d\phi} = 0$$

$$\langle g(x)\rangle_{p(x)} - \frac{1}{z(\phi)}\int g(x) e^{\phi^T g(x)} dx = 0$$

$$\int g(x) e^{\phi^T g(x)} dx \rightarrow q(x)$$

$$\langle g(x)\rangle_{p(x)} - \int g(x)\, q(x)\, dx = 0$$

$$\langle g(x)\rangle_{p(x)} - \langle g(x)\rangle_{q(x)} = 0$$

$$\Rightarrow \boxed{\langle g(x)\rangle_{p(x)} = \langle g(x)\rangle_{q(x)}}$$

3.2.2 To show $N(x/\mu, \sigma^2) = \frac{1}{z(\phi)} e^{\phi^T g(x)}$

where $g_1(x) = x$, $g_2(x) = x^2$ for some ϕ

$N(x/\mu, \sigma^2)$ expanding the gaussian

$$\Rightarrow N(x/\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

$$= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{x^2 - \mu^2 + 2x\mu}{2\sigma^2}\right]$$

from given constraints

$$= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{g_2(x)}{2\sigma^2} + \frac{\mu g_1(x)}{2\sigma^2} - \frac{\mu^2}{2\sigma^2}\right]$$

$g_1(x) = x \quad g_2(x) = x^2$

Let $\frac{1}{Z(\phi)} = \frac{1}{\sqrt{2\pi\sigma^2}}$ (5)

$\Rightarrow N(x/\mu, \sigma^2) = \frac{1}{Z(\phi)} \exp\left(-\frac{g_2(x)}{2\sigma^2} + \frac{\mu}{\sigma^2} g_1(x) - \frac{\mu^2}{2\sigma^2}\right)$

we can choose ϕ as in terms of $-\frac{\mu}{2\sigma^2}$, $\frac{\mu}{\sigma^2}$, $\frac{-1}{2\sigma^2}$)

and g in terms of $g_1(x)$, $g_2(x)$, 1

To match matrix order

$\Rightarrow \phi = \begin{bmatrix} -1/2\sigma^2 \\ \mu/\sigma^2 \\ -\mu^2/2\sigma^2 \end{bmatrix}$ $g = \begin{bmatrix} g_2(x) \\ g_1(x) \\ 1 \end{bmatrix}$

$\Rightarrow$ using the above $\frac{1}{Z(\phi)} e^{\phi^T g(x)} \rightarrow N(x/\mu, \sigma^2)$

3.2.3 To show at minimum $KL(p||q)$

$\mu = \langle x \rangle p(x)$ $\sigma^2 = \langle x^2 \rangle p(x) - \langle x \rangle^2 p(x)$

$\mu = \int x f(x) dx$

$\Rightarrow \mu = \int x \underbrace{N(x/\mu, \sigma^2)}_{\text{by this is } \frac{1}{Z(\phi)} e^{\phi^T g(x)} \text{ from 3.2.2}} dx$

$\mu = \int x \underbrace{\frac{1}{Z(\phi)} e^{\phi^T g(x)}}_{\rightarrow q(x) \text{ from 3.2.1}} dx$

$$\mu = \int x\, q(x)\, dx$$

$$\boxed{\mu = \langle x \rangle_{q(x)}}$$

We know

$$\sigma^2 = E(x^2) - (E(x))^2$$

$$= \int x^2 \mathcal{N}(x|\mu, \sigma^2)\, dx - \underbrace{\left[\int x\, \mathcal{N}(x|\mu, \sigma^2) dx\right]^2}_{\mu}$$

$$= \int x^2 \underbrace{\frac{1}{Z(\phi)} e^{\phi^T g(x)}}_{q(x)} dx - \left(\langle x \rangle_{q(x)}\right)^2 \quad \text{3.2.1}$$

$$= \int x^2 q(x)\, dx - \left(\langle x \rangle_{q(x)}\right)^2$$

$$= \langle x^2 \rangle_{q(x)} - \left(\langle x \rangle_{q(x)}\right)^2$$

From 3.2.1 at optimal KL is min. and $\langle g(x) \rangle_{p(x)} = \langle g(x) \rangle_{q(x)}$

$$\Rightarrow \langle x^2 \rangle_{q(x)} = \langle x^2 \rangle_{p(x)}$$

$$\langle x \rangle_{q(x)} = \langle x^2 \rangle_{p(x)}$$

$$\Rightarrow \boxed{\sigma^2 = \langle x^2 \rangle_{p(x)} - \left[\langle x \rangle_{p(x)}\right]^2}$$