# hw9

December 9, 2021

# 1 Mathematical Foundations on Deep Neural Network

## 1.1 Homework #9

### 1.1.1 2017-11362

## 1.2 Problem 1.

$$C = \{ (x_1, x_2) \in \mathbb{R}^2 : x_1 = a,\ 0 \le x_2 \le 1 \}$$

$$\Pi_C(y) = \underset{x \in C}{\operatorname{argmin}} \|x - y\|^2 = \underset{\substack{x_1 = a \\ 0 \le x_2 \le 1}}{\operatorname{argmin}} \{ (x_1 - y_1)^2 + (x_2 - y_2)^2 \}$$

$$\hat{x}_1 = \underset{x_1 = a}{\operatorname{argmin}} (x_1 - y_1)^2 = a$$

$$\hat{x}_2 = \underset{0 \le x_2 \le 1}{\operatorname{argmin}} (x_2 - y_2)^2 = \begin{cases} 1, & y_2 > 1 \\ y_2, & 0 \le y_2 \le 1 \\ 0, & y_2 < 0 \end{cases} = \min(\max(y_2, 0), 1)$$

$$\therefore \Pi_C(y) = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix} = \begin{bmatrix} a \\ \min(\max(y_2, 0), 1) \end{bmatrix}$$

## 1.3 Problem 2.

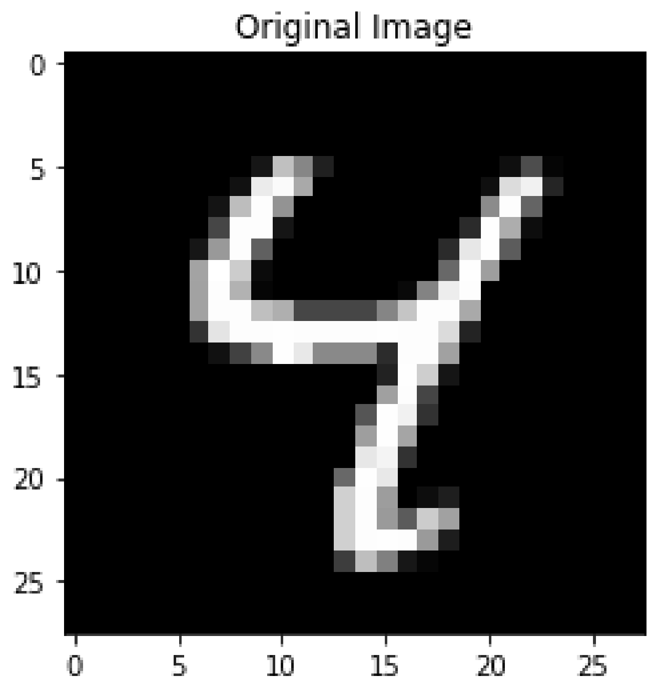
[2]:
```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torchvision
from torchvision import datasets, transforms
from torchvision.utils import save_image, make_grid

import numpy as np
import matplotlib.pyplot as plt

batch_size = 128
(full_dim, mid_dim, hidden) = (1 * 28 * 28, 1000, 5)
lr = 1e-3
```

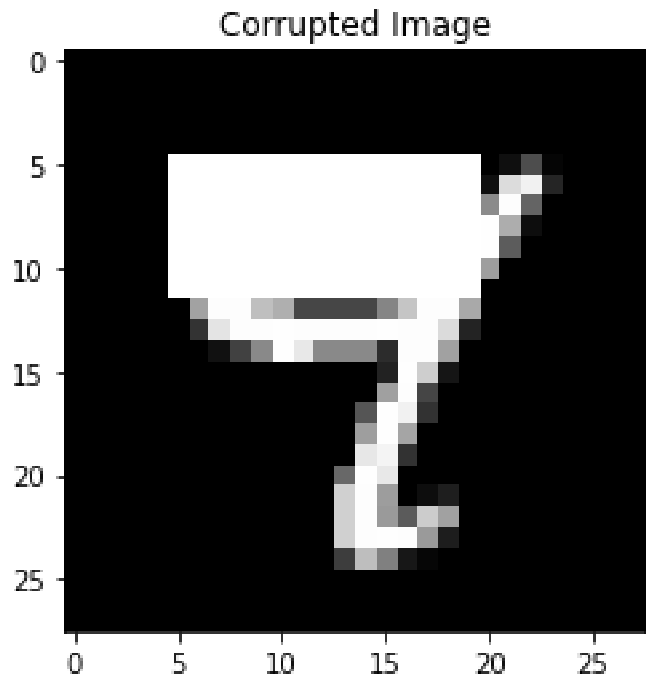
```
epochs = 100
device = torch.device("cpu")

##############################################
# STEP 1: Define dataset and preprocessing #
##############################################


class Logistic(torch.distributions.Distribution):
    def __init__(self):
        super(Logistic, self).__init__()

    def log_prob(self, x):
        return -(F.softplus(x) + F.softplus(-x))

    def sample(self, size):
        z = torch.distributions.Uniform(0., 1.).sample(size).to(device)
        return torch.log(z) - torch.log(1. - z)

####################################
# STEP 3: Implement Coupling Layer #
####################################

class Coupling(nn.Module):
    def __init__(self, in_out_dim, mid_dim, hidden, mask_config):
        super(Coupling, self).__init__()
        self.mask_config = mask_config

        self.in_block = nn.Sequential(nn.Linear(in_out_dim//2, mid_dim), nn.
 ↪ReLU())
        self.mid_block = nn.ModuleList([nn.Sequential(nn.Linear(mid_dim,␣
 ↪mid_dim), nn.ReLU())
                                                               for _ in␣
 ↪range(hidden - 1)])
        self.out_block = nn.Linear(mid_dim, in_out_dim//2)

    def forward(self, x, reverse=False):
        [B, W] = list(x.size())
        x = x.reshape((B, W//2, 2))
        if self.mask_config:
            on, off = x[:, :, 0], x[:, :, 1]
        else:
            off, on = x[:, :, 0], x[:, :, 1]

        off_ = self.in_block(off)
        for i in range(len(self.mid_block)):
            off_ = self.mid_block[i](off_)
```

```
        shift = self.out_block(off_)

        if reverse:
            on = on - shift
        else:
            on = on + shift

        if self.mask_config:
            x = torch.stack((on, off), dim=2)
        else:
            x = torch.stack((off, on), dim=2)
        return x.reshape((B, W))

class Scaling(nn.Module):
    def __init__(self, dim):
        super(Scaling, self).__init__()
        self.scale = nn.Parameter(torch.zeros((1, dim)), requires_grad=True)

    def forward(self, x, reverse=False):
        log_det_J = torch.sum(self.scale)
        if reverse:
            x = x * torch.exp(-self.scale)
        else:
            x = x * torch.exp(self.scale)
        return x, log_det_J

##########################
# STEP 4: Implement NICE #
##########################

class NICE(nn.Module):
    def __init__(self,in_out_dim, mid_dim, hidden, mask_config=1.0, coupling=4):
        super(NICE, self).__init__()
        self.prior = Logistic()
        self.in_out_dim = in_out_dim

        self.coupling = nn.ModuleList([
            Coupling(in_out_dim=in_out_dim,
                     mid_dim=mid_dim,
                     hidden=hidden,
                     mask_config=(mask_config+i)%2) \
            for i in range(coupling)])

        self.scaling = Scaling(in_out_dim)

    def g(self, z):
        x, _ = self.scaling(z, reverse=True)
```

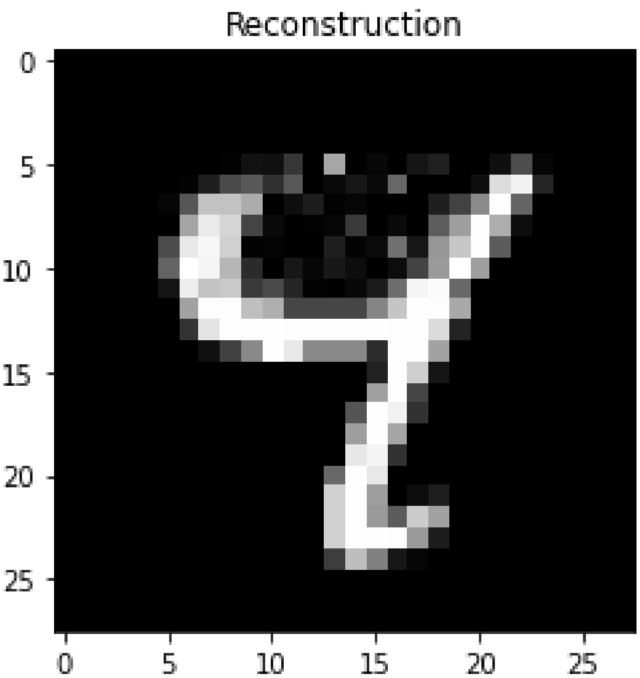
```
        for i in reversed(range(len(self.coupling))):
            x = self.coupling[i](x, reverse=True)
        return x

    def f(self, x):
        for i in range(len(self.coupling)):
            x = self.coupling[i](x)
        z, log_det_J = self.scaling(x)
        return z, log_det_J

    def log_prob(self, x):
        z, log_det_J = self.f(x)
        log_ll = torch.sum(self.prior.log_prob(z), dim=1)
        return log_ll + log_det_J

    def sample(self, size):
        z = self.prior.sample((size, self.in_out_dim)).to(device)
        return self.g(z)

    def forward(self, x):
        return self.log_prob(x)


# Load pre-trained NICE model onto CPU
model = NICE(in_out_dim=784, mid_dim=1000, hidden=5).to(device)
model.load_state_dict(torch.load('nice.pt',map_location=torch.device('cpu')))

# Since we do not update model, set requires_grad = False
model.requires_grad_(False)

# Get an MNIST image
testset = torchvision.datasets.MNIST(root='../data/mnist_data', train=False,␣
 ↪download=True, transform=torchvision.transforms.ToTensor())
test_loader = torch.utils.data.DataLoader(testset, batch_size=1, shuffle=False)
pass_count = 6
itr = iter(test_loader)
for _ in range(pass_count+1):
    image,_ = itr.next()

plt.figure(figsize = (4,4))
plt.title('Original Image')
plt.imshow(make_grid(image.squeeze().detach()).permute(1,2,0))
plt.show()
# plt.savefig('plt1.png')


# Create mask
```

```
mask = torch.ones_like(image,dtype=torch.bool)
mask[:,:,5:12,5:20] = 0

# Partially corrupt the image
image[mask.logical_not()] = torch.ones_like(image[mask.logical_not()])
plt.figure(figsize = (4,4))
plt.title('Corrupted Image')
plt.imshow(make_grid(image.squeeze()).permute(1,2,0))
plt.show()
# plt.savefig('plt2.png')
```






[3]:
```
lr = 1e-3
X = image.clone().requires_grad_(True)

optimizer = torch.optim.SGD([X], lr=lr)

for i in range(3000):
    loss = - model(X.view(1, -1))
    loss.backward()

    optimizer.step()
    optimizer.zero_grad()

    X.data = torch.clamp(image.data * mask + X.data * (~ mask), 0, 1)

# Plot reconstruction
plt.figure(figsize = (4,4))
plt.title('Reconstruction')
plt.imshow(make_grid(X.squeeze().detach()).permute(1,2,0))
plt.show()
# plt.savefig('plt3.png')
```




## 1.4 Problem 3.

(a) $\mathrm{VLB}^{(k)}_{\theta,\phi}(x) = \mathbb{E}_{z_1,\cdots,z_k\sim q_\phi(z|x)}\left[\log\left(\frac{1}{k}\sum_{i=1}^{k}\frac{p_\theta(x|z_i)\,p_z(z_i)}{q_\phi(z_i|x)}\right)\right]$

$\leq \log\left(\mathbb{E}_{z_1,\cdots,z_k\sim q_\phi(z|x)}\left[\frac{1}{k}\sum_{i=1}^{k}\frac{p_\theta(x|z_i)\,p_z(z_i)}{q_\phi(z_i|x)}\right]\right)$ ($\because \log x$ is concave $\Rightarrow \log(E(X)) \geq E(\log X)$)

$= \log\left(\frac{1}{k}\sum_{i=1}^{k}\mathbb{E}_{z_i\sim q_\phi(z|x)}\left[\frac{p_\theta(x|z_i)\,p_z(z_i)}{q_\phi(z_i|x)}\right]\right)$

$= \log\left(\mathbb{E}_{z\sim q_\phi(z|x)}\left[\frac{p_\theta(x|z)\,p_z(z)}{q_\phi(z|x)}\right]\right)$

$= \log\left(\int p_\theta(x|z)\,p_z(z)\,dz\right) = \log p_\theta(x)$

(b) Let $k \geq M$. Suppose $I \subset \{1,\cdots,k\}$ s.t $|I| = M$.

$\mathrm{VLB}^{(k)}_{\theta,\phi}(x) = \mathbb{E}_{z_1,\cdots,z_k\sim q_\phi(z|x)}\left[\log\left(\frac{1}{k}\sum_{i=1}^{k}\frac{p_\theta(x|z_i)\,p_z(z_i)}{q_\phi(z_i|x)}\right)\right]$

$= \mathbb{E}_{z_1,\cdots,z_k\sim q_\phi(z|x)}\left[\log\left(\mathbb{E}_{I=\{i_1,\cdots,i_M\}}\left[\frac{1}{M}\sum_{m=1}^{M}\frac{p_\theta(x|z_{i_m})\,p_z(z_{i_m})}{q_\phi(z_{i_m}|x)}\right]\right)\right]$

$\geq \mathbb{E}_{z_1,\cdots,z_k\sim q_\phi(z|x)}\left[\mathbb{E}_{I=\{i_1,\cdots,i_M\}}\left[\log\left(\frac{1}{M}\sum_{m=1}^{M}\frac{p_\theta(x|z_{i_m})\,p_z(z_{i_m})}{q_\phi(z_{i_m}|x)}\right)\right]\right]$

$= \mathbb{E}_{z_1,\cdots,z_M\sim q_\phi(z|x)}\left[\log\left(\frac{1}{M}\sum_{m=1}^{M}\frac{p_\theta(x|z_m)\,p_z(z_m)}{q_\phi(z_m|x)}\right)\right]$

$= \mathrm{VLB}^{(M)}_{\theta,\phi}(x)$.

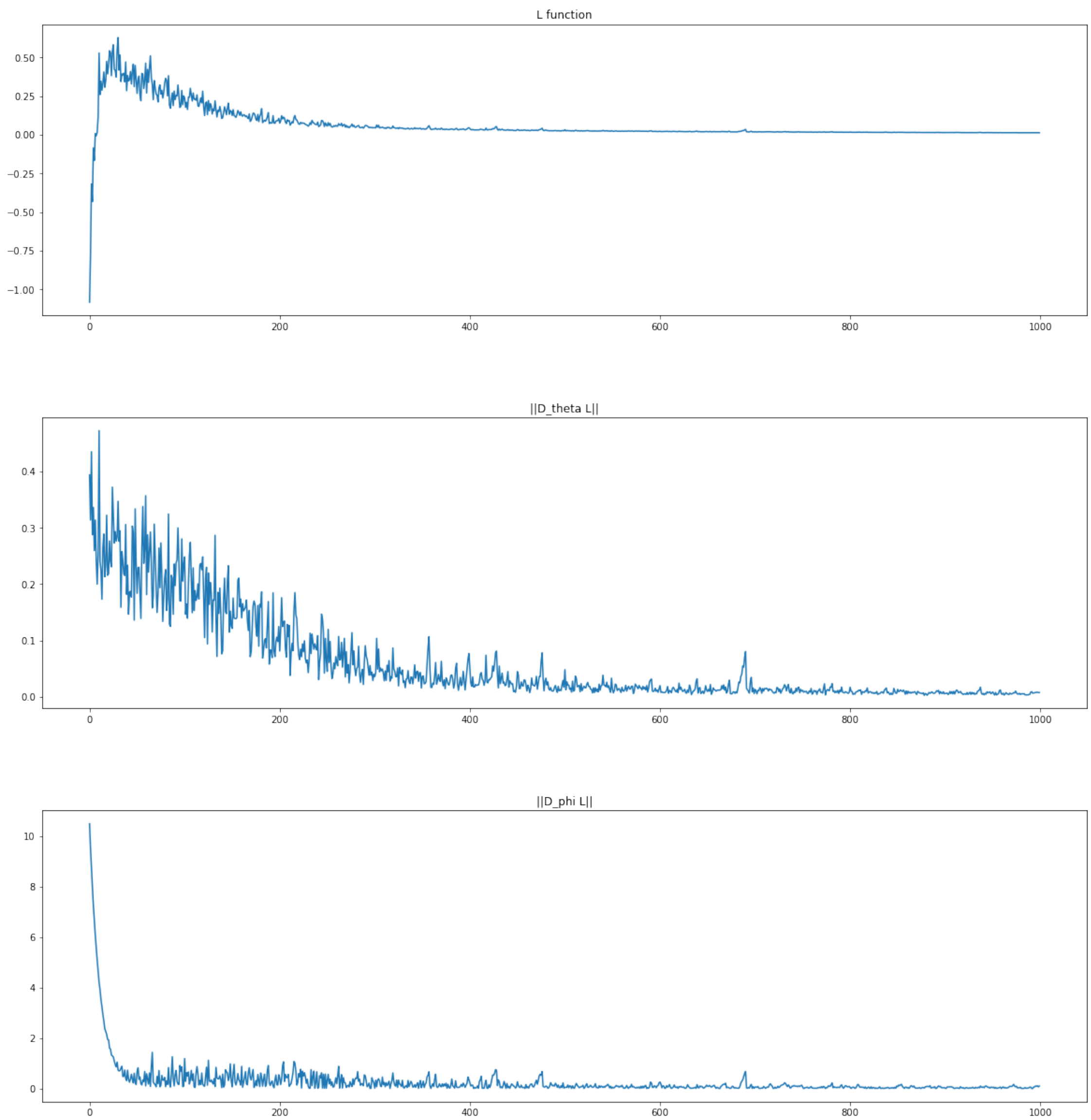
(c) $q_\phi(z|x)$ is "powerful enough" $\Rightarrow \exists \phi$ s.t $\log p_\theta(x) = \mathrm{VLB}^{(k)}_{\theta,\phi}(x)$.

By HW#8 problem 2, $\underset{\theta\in\Theta}{\text{maximize}} \log p_\theta(x) = \underset{\theta\in\Theta,\phi\in\Phi}{\text{maximize}} \mathrm{VLB}^{(k)}_{\theta,\phi}(x)$.

### 1.5 Problem 4.

[4]:
```
import numpy as np
from matplotlib import pyplot as plt
import math

N, p = 30, 20
np.random.seed(0)
X = np.random.randn(N,p)
Y = 2*np.random.randint(2, size = N) - 1
lamda = 30

theta = 0.1 * np.random.randn(p)
phi = 0.1 * np.random.randn(p)
alpha = 3e-1
beta = 1e-4

epoch = 1000
L_val = []
d_phi_val = []
d_theta_val = []

for _ in range(epoch):
    for __ in range(N):
        ind = np.random.randint(N)
        Xi, Yi = X[ind, :], Y[ind]
        expo = np.exp(Yi * ((Xi - phi) @ theta))
        grad_theta = - Yi / (1 + expo) * (Xi - phi)
        grad_phi = Yi / (1 + expo) * theta - lamda * phi
        theta -= alpha * grad_theta
        phi += beta * grad_phi

    L_i = np.average(np.log(1 + np.exp(-Y * ((X - phi.reshape(1,-1)) @␣
 ↪theta)))) - lamda/2 * np.linalg.norm(phi, axis=0, ord=2) **2
    d_phi = np.average(Y / (1 + np.exp(Y * ((X-phi.reshape(1,-1)) @ theta))) *␣
 ↪theta - lamda * phi
    d_theta = np.average((-Y / (1 + np.exp(Y * ((X-phi.reshape(1,-1)) @␣
 ↪theta)))).reshape(-1,1)*(X-phi.reshape(1,-1)), axis=0)
    L_val.append(L_i)
    d_phi_val.append(d_phi)
    d_theta_val.append(d_theta)

fig, ax = plt.subplots(figsize=(20, 20))
plt.subplots_adjust(left=0.125,
                    bottom=0.1,
                    right=0.9,
                    top=0.9,
```

```
                        wspace=0.2,
                        hspace=0.35)
plt.subplot(3, 1, 1)
plt.title("L function")
plt.plot(L_val)
plt.subplot(3, 1, 2)
plt.title("||D_theta L||")
plt.plot(np.linalg.norm(d_theta_val, axis=1, ord=2))
plt.subplot(3, 1, 3)
plt.title("||D_phi L||")
plt.plot(np.linalg.norm(d_phi_val, axis=1, ord=2))
plt.show()
```



## 1.6 Problem 5.

(a) $\min_{P_A} \max_{P_B} E_{P_A, P_B}[\text{points of } B]$

For $P_A = (x, y, 1-x-y)^t$, $0 \le x, y,\ x+y \le 1$, $P_B = (a, b, 1-a-b)^t$, $0 \le a, b,\ a+b \le 1$.

$$E[\text{pts. of } B] = \frac{1}{9}\left[xb + y(1-a-b) + (1-x-y)a - ya - (1-x-y)b - x(1-a-b)\right]$$

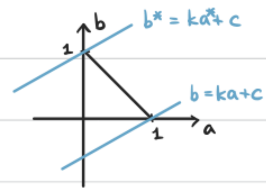
$$= \frac{xb}{3} - \frac{ya}{3} + \frac{1}{9}(y - x + a - b)$$

$$= \frac{1}{3}\left(x - \frac{1}{3}\right)\left(b - \frac{1}{3}\right) - \frac{1}{3}\left(y - \frac{1}{3}\right)\left(a - \frac{1}{3}\right)$$

$$E_{P_A^*, P_B} \overset{①}{\le} E_{P_A^*, P_B^*} \overset{②}{\le} E_{P_A, P_B^*}$$

① $\Leftrightarrow \left(x^* - \frac{1}{3}\right)\left(b - \frac{1}{3}\right) - \left(y^* - \frac{1}{3}\right)\left(a - \frac{1}{3}\right) \le \left(x^* - \frac{1}{3}\right)\left(b^* - \frac{1}{3}\right) - \left(y^* - \frac{1}{3}\right)\left(a^* - \frac{1}{3}\right)$

$\Leftrightarrow \left(x^* - \frac{1}{3}\right)(b - b^*) \le \left(y^* - \frac{1}{3}\right)(a - a^*)$, $\forall a, b$ s.t $0 \le a, b,\ a+b \le 1$




② $\Leftrightarrow (x^* - x)\left(b^* - \frac{1}{3}\right) \le (y^* - y)\left(a^* - \frac{1}{3}\right)$, $\forall x, y$ s.t $0 \le x, y,\ x+y \le 1$

i) $x^* > \frac{1}{3}$, $y^* > \frac{1}{3}$ : ① $\Rightarrow b - b^* \le k(a - a^*)$, $k > 0$ $\Rightarrow a^* = 0$, $b^* = 1$

② $\Rightarrow 2x^* + y^* \le 2x + y$ (contradiction)

ii) $x^* > \frac{1}{3}$, $y^* < \frac{1}{3}$ : ① $\Rightarrow b - b^* \le -k(a - a^*)$, $k > 0$ $\Rightarrow$ $a^* = 0, b^* = 1$ $(k < 1)$, $a^* = 1$, $b^* = 0$ $(k > 1)$

② $\Rightarrow \begin{cases} k < 1 : 2x^* + y^* \le 2x + y \text{ (contradiction)} \\ k > 1 : x^* + 2y^* \le x + 2y \text{ (contradiction)} \end{cases}$

We can similarly derive contradiction at other case except $x^* = y^* = \frac{1}{3}$.

$\therefore x^* = y^* = \frac{1}{3}$.

Similarly $a^* = b^* = \frac{1}{3}$.

$\therefore P_A^* = \left[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right]^t$, $P_B = \left[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right]^t$ is the unique solution of minimax problem.

(b) Arbitrary $P_A^*$ might satisfy $E_{P_A^*, P_B^*} \le E_{P_A, P_B^*}$.

However, minimax problem should satisfy both $E_{P_A^*, P_B} \le E_{P_A^*, P_B^*}$ and $E_{P_A^*, P_B^*} \le E_{P_A, P_B^*}$.

$E_{P_A^*, P_B} \le E_{P_A^*, P_B^*}$ cannot be satisfied when $P_A^* \ne \left[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right]^t$.

$\therefore$ any stretagy is not optimal for A.