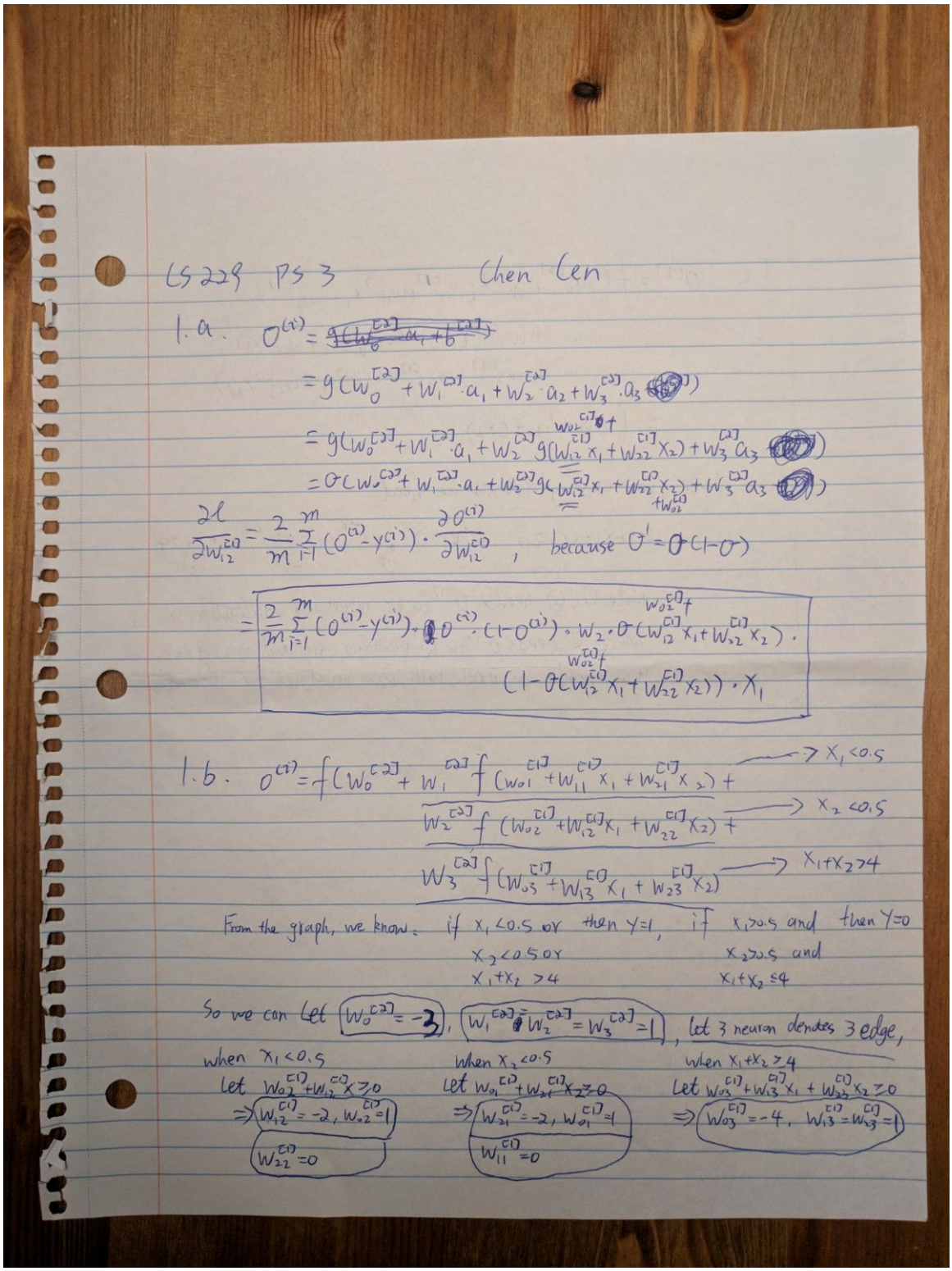
1.

(a)(b)



CS229 PS3 Chen Cen

1.a. $O^{(i)} = g(w_0^{[2]} + w_1^{[2]} \cdot a_1 + w_2^{[2]} \cdot a_2 + w_3^{[2]} \cdot a_3)$

$= g(w_0^{[2]} + w_1^{[2]} \cdot a_1 + w_2^{[2]} g(w_{02}^{[1]} + w_{12}^{[1]} x_1 + w_{22}^{[1]} x_2) + w_3^{[2]} a_3)$

$= \sigma(w_0^{[2]} + w_1^{[2]} \cdot a_1 + w_2^{[2]} g(w_{12}^{[1]} x_1 + w_{22}^{[1]} x_2 + w_{02}^{[1]}) + w_3^{[2]} a_3)$

$\frac{\partial l}{\partial w_{12}^{[1]}} = \frac{2}{m} \sum_{i=1}^{m} (O^{(i)} - y^{(i)}) \cdot \frac{\partial O^{(i)}}{\partial w_{12}^{[1]}}$, because $\sigma' = \sigma(1-\sigma)$

$= \frac{2}{m} \sum_{i=1}^{m} (O^{(i)} - y^{(i)}) \cdot O^{(i)} \cdot (1 - O^{(i)}) \cdot w_2 \cdot \sigma(w_{12}^{[1]} x_1 + w_{22}^{[1]} x_2 + w_{02}^{[1]}) \cdot (1 - \sigma(w_{12}^{[1]} x_1 + w_{22}^{[1]} x_2 + w_{02}^{[1]})) \cdot x_1$

1.b. $O^{(i)} = f(w_0^{[2]} + w_1^{[2]} f(w_{01}^{[1]} + w_{11}^{[1]} x_1 + w_{21}^{[1]} x_2) + w_2^{[2]} f(w_{02}^{[1]} + w_{12}^{[1]} x_1 + w_{22}^{[1]} x_2) + w_3^{[2]} f(w_{03}^{[1]} + w_{13}^{[1]} x_1 + w_{23}^{[1]} x_2))$

$\rightarrow x_1 < 0.5$

$\rightarrow x_2 < 0.5$

$\rightarrow x_1 + x_2 > 4$

From the graph, we know: if $x_1 < 0.5$ or $x_2 < 0.5$ or $x_1 + x_2 > 4$ then $Y = 1$, if $x_1 > 0.5$ and $x_2 > 0.5$ and $x_1 + x_2 \leq 4$ then $Y = 0$

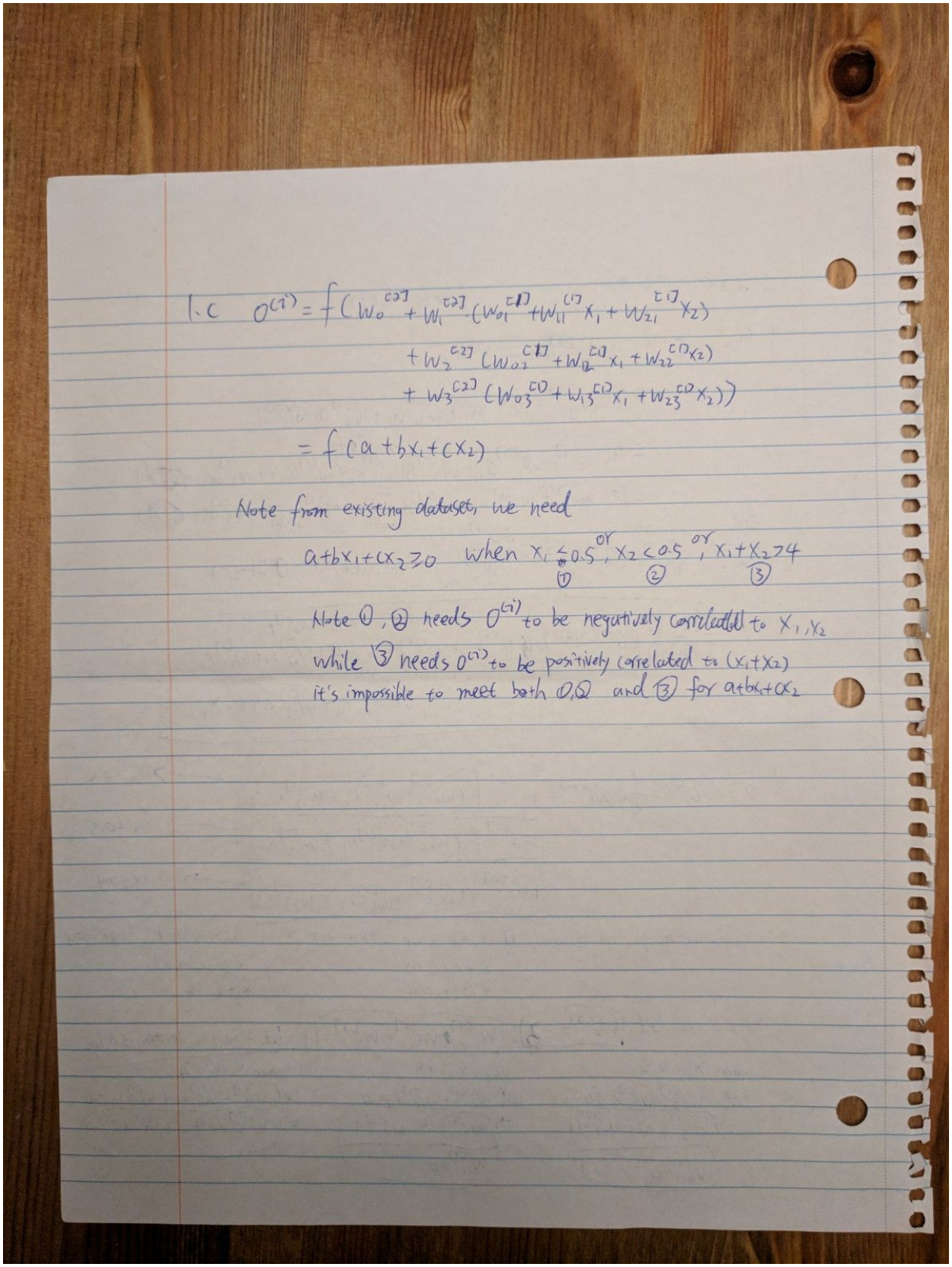
So we can let $w_0^{[2]} = -3$, $w_1^{[2]} = w_2^{[2]} = w_3^{[2]} = 1$, let 3 neuron denotes 3 edge,

when $x_1 < 0.5$ let $w_{02}^{[1]} + w_{12}^{[1]} x \geq 0$ $\Rightarrow$ $w_{12}^{[1]} = -2$, $w_{02}^{[1]} = 1$; $w_{22}^{[1]} = 0$

when $x_2 < 0.5$ let $w_{01}^{[1]} + w_{21}^{[1]} x_2 \geq 0$ $\Rightarrow$ $w_{21}^{[1]} = -2$, $w_{01}^{[1]} = 1$; $w_{11}^{[1]} = 0$

when $x_1 + x_2 \geq 4$ let $w_{03}^{[1]} + w_{13}^{[1]} x_1 + w_{23}^{[1]} x_2 \geq 0$ $\Rightarrow$ $w_{03}^{[1]} = -4$, $w_{13}^{[1]} = w_{23}^{[1]} = 1$

(c)



1.c $O^{(i)} = f(w_0^{[2]} + w_1^{[2]}(w_{01}^{[1]} + w_{11}^{[1]}x_1 + w_{21}^{[1]}x_2) + w_2^{[2]}(w_{02}^{[1]} + w_{12}^{[1]}x_1 + w_{22}^{[1]}x_2) + w_3^{[2]}(w_{03}^{[1]} + w_{13}^{[1]}x_1 + w_{23}^{[1]}x_2))$

$= f(a + bx_1 + cx_2)$

Note from existing datasets, we need

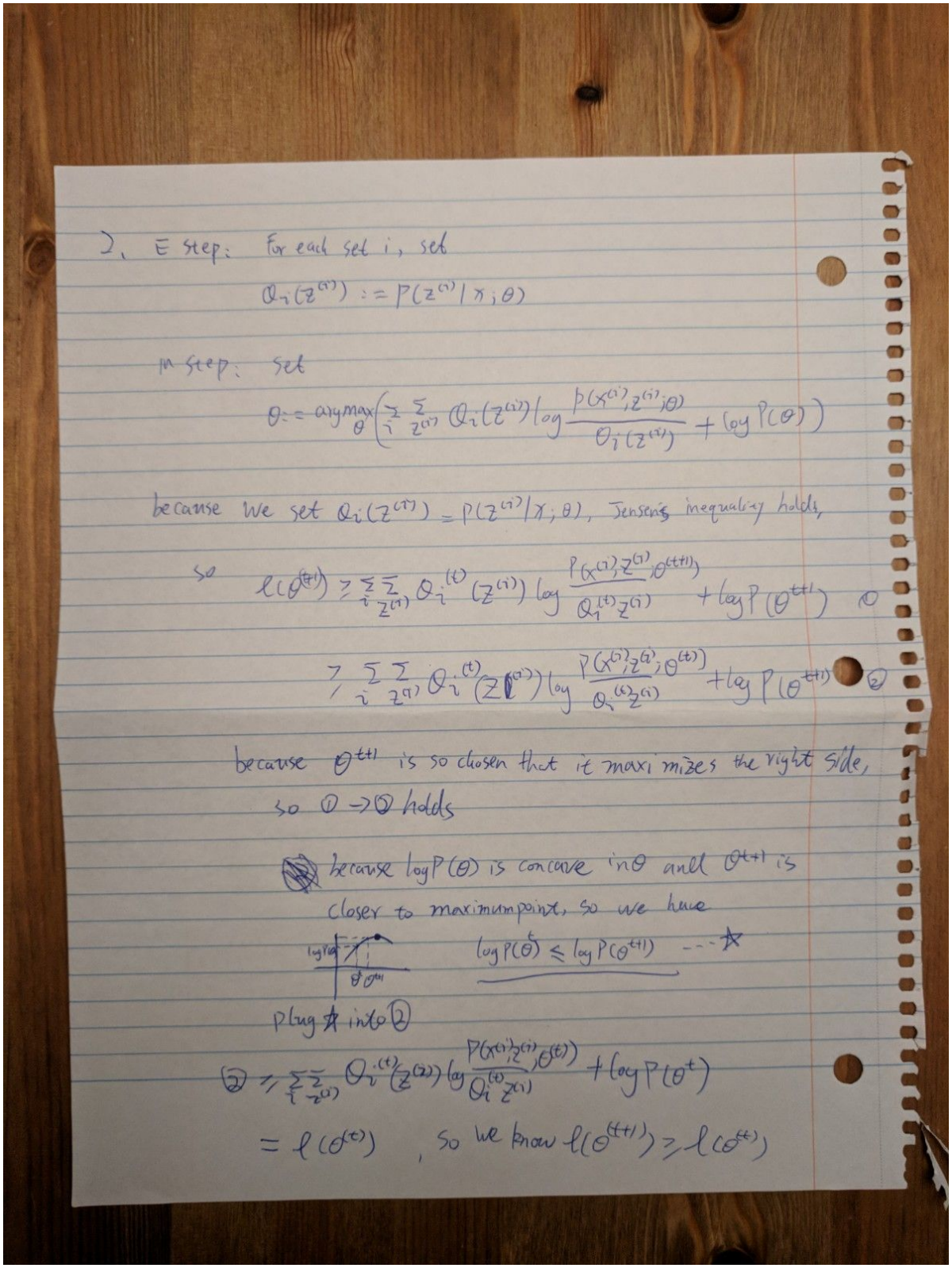
$a + bx_1 + cx_2 \geq 0$ when $x_1 \leq 0.5$ ① or $x_2 < 0.5$ ② or $x_1 + x_2 > 4$ ③

Note ①, ② needs $O^{(i)}$ to be negatively correlated to x_1, x_2

while ③ needs $O^{(i)}$ to be positively correlated to $(x_1 + x_2)$

it's impossible to meet both ①,② and ③ for $a + bx_1 + cx_2$

2.



1. E step: For each set i, set

$$Q_i(z^{(i)}) := p(z^{(i)} | x; \theta)$$

M step: Set

$$\theta := \arg\max_\theta \left(\sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})} + \log p(\theta) \right)$$

because we set $Q_i(z^{(i)}) = p(z^{(i)} | x; \theta)$, Jensen's inequality holds,

so $$l(\theta^{(t+1)}) \geq \sum_i \sum_{z^{(i)}} Q_i^{(t)}(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \theta^{(t+1)})}{Q_i^{(t)}(z^{(i)})} + \log p(\theta^{(t+1)}) \quad ①$$

$$\geq \sum_i \sum_{z^{(i)}} Q_i^{(t)}(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \theta^{(t)})}{Q_i^{(t)}(z^{(i)})} + \log p(\theta^{(t+1)}) \quad ②$$

because $\theta^{(t+1)}$ is so chosen that it maximizes the right side,
so ① → ② holds

because $\log p(\theta)$ is concave in θ and $\theta^{(t+1)}$ is closer to maximum point, so we have

$$\log p(\theta^{(t)}) \leq \log p(\theta^{(t+1)}) \quad \text{---} \star$$

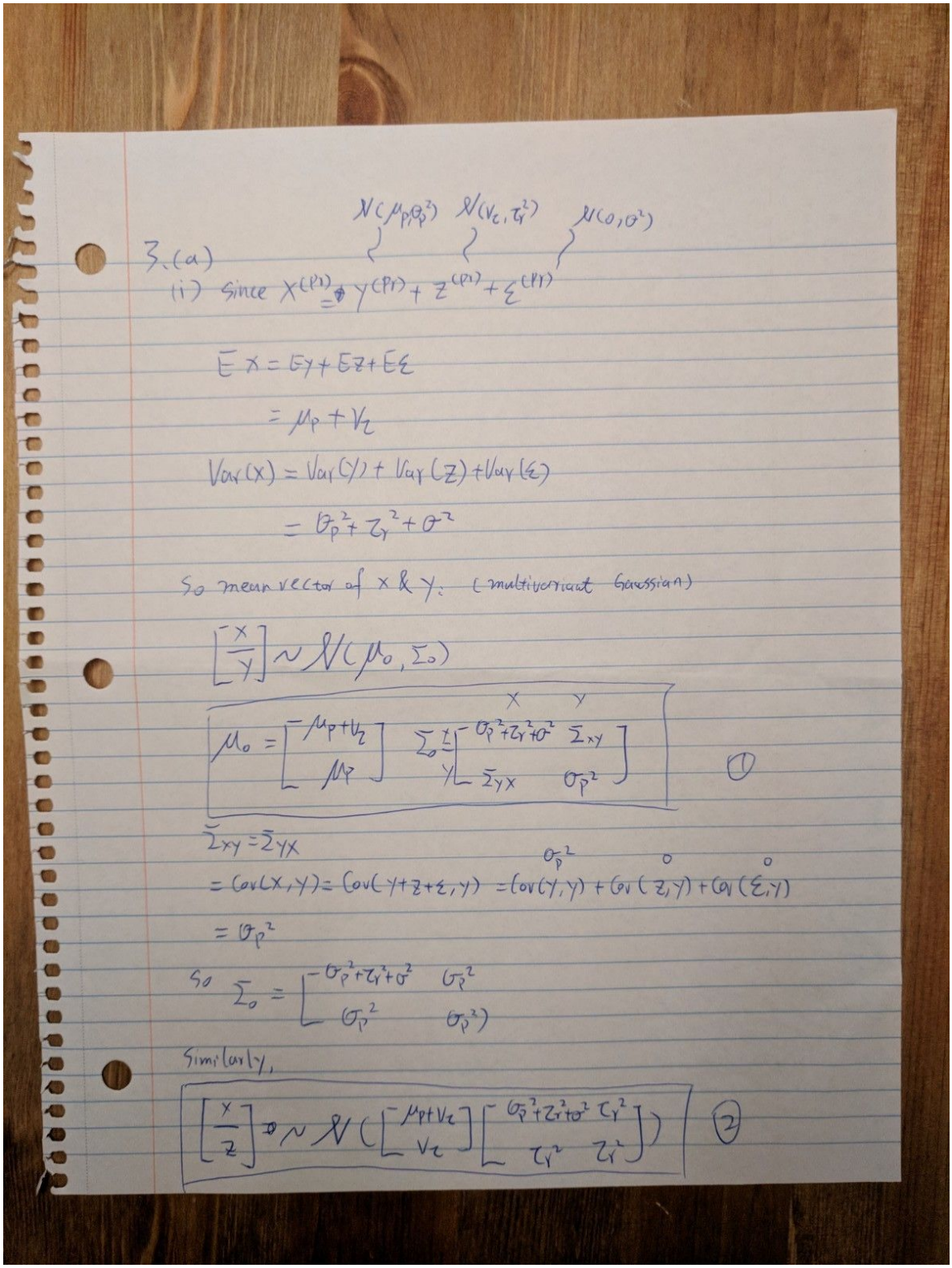
Plug ★ into ②

$$② \geq \sum_i \sum_{z^{(i)}} Q_i^{(t)}(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \theta^{(t)})}{Q_i^{(t)}(z^{(i)})} + \log p(\theta^{(t)})$$

$$= l(\theta^{(t)}), \text{ so we know } l(\theta^{(t+1)}) \geq l(\theta^{(t)})$$

3.

(a)



3. (a)

(i) Since $X^{(pi)} = Y^{(pi)} + Z^{(pi)} + \xi^{(pi)}$, with $Y^{(pi)} \sim N(\mu_p, \sigma_p^2)$, $Z^{(pi)} \sim N(\nu_z, \tau_r^2)$, $\xi^{(pi)} \sim N(0, \sigma^2)$

$$E X = EY + EZ + E\xi$$

$$= \mu_p + \nu_z$$

$$Var(X) = Var(Y) + Var(Z) + Var(\xi)$$

$$= \sigma_p^2 + \tau_r^2 + \sigma^2$$

So mean vector of X & Y: (multivariate Gaussian)

$$\begin{bmatrix} X \\ Y \end{bmatrix} \sim N(\mu_0, \Sigma_0)$$

$$\mu_0 = \begin{bmatrix} \mu_p + \nu_z \\ \mu_p \end{bmatrix} \quad \Sigma_0 = \begin{bmatrix} \sigma_p^2 + \tau_r^2 + \sigma^2 & \Sigma_{XY} \\ \Sigma_{YX} & \sigma_p^2 \end{bmatrix} \quad (1)$$

$\Sigma_{XY} = \Sigma_{YX}$

$$= Cov(X, Y) = Cov(Y + Z + \xi, Y) = Cov(Y, Y) + Cov(Z, Y) + Cov(\xi, Y)$$

$$= \sigma_p^2 \quad (0 \text{ and } 0)$$

So $\Sigma_0 = \begin{bmatrix} \sigma_p^2 + \tau_r^2 + \sigma^2 & \sigma_p^2 \\ \sigma_p^2 & \sigma_p^2 \end{bmatrix}$

Similarly,

$$\begin{bmatrix} X \\ Z \end{bmatrix} \sim N\left(\begin{bmatrix} \mu_p + \nu_z \\ \nu_z \end{bmatrix}, \begin{bmatrix} \sigma_p^2 + \tau_r^2 + \sigma^2 & \tau_r^2 \\ \tau_r^2 & \tau_r^2 \end{bmatrix} \right) \quad (2)$$

Combine ① & ②

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_p + V_z \\ \mu_p \\ V_z \end{bmatrix}, \begin{array}{c} \\ X \\ Y \\ Z \end{array} \begin{bmatrix} \sigma_p^2 + \tau_1^2 + \theta^2 & \sigma_p^2 & \tau_1^2 \\ \sigma_p^2 & \sigma_p^2 & 0 \\ \tau_1^2 & 0 & \tau_2^2 \end{bmatrix} \right)$$

X Y Z

mean vector ↑ cov matrix ↑

(ii) From ② above,

we know $Z|X \sim \mathcal{N}(\mu_{Z|X}, \Sigma_{Z|X})$

per conditional distribution from notes,

$$\mu_{Z|X} = V_z + \frac{\tau_1^2}{\sigma_p^2 + \tau_1^2 + \theta^2} \cdot \frac{1}{\sigma_p^2} \cdot (X - (\mu_p + V_z))$$

$$\Sigma_{Z|X} = \tau_2^2 - \frac{1}{\sigma_p^2 + \tau_1^2 + \theta^2} (\tau_1^2)^2 = \frac{\tau_1^2 \sigma_p^2 + \tau_2^2 \theta^2}{\sigma_p^2 + \tau_1^2 + \theta^2}$$

Similarly, from ① above

$$\mu_{Y|X} = \mu_p + \frac{\sigma_p^2}{\sigma_p^2 + \tau_1^2 + \theta^2} \cdot \frac{1}{\sigma_p^2} \cdot (X - (\mu_p + V_z))$$

$$\Sigma_{Y|X} = \sigma_p^2 - \frac{1}{\sigma_p^2 + \tau_1^2 + \theta^2} (\sigma_p^2)^2 = \frac{\tau_1^2 \sigma_p^2 + \theta^2 \sigma_p^2}{\sigma_p^2 + \tau_1^2 + \theta^2}$$

So $Q_i(z^{(i)}, y^{(i)}) = P(z^{(i)}, y^{(i)} | x^{(i)}, \theta) \sim$

$$\mathcal{N}\left(\begin{bmatrix} -V_z + \frac{\tau_1^2}{\sigma_p^2 + \tau_1^2 + \theta^2} (X - \mu_p - V_z) \\ \mu_p + \frac{\sigma_p^2}{\sigma_p^2 + \tau_1^2 + \theta^2} (X - \mu_p - V_z) \end{bmatrix}, \begin{bmatrix} \frac{\tau_1^2 \sigma_p^2 + \tau_2^2 \theta^2}{\sigma_p^2 + \tau_1^2 + \theta^2} & 0 \\ 0 & \frac{\tau_1^2 \sigma_p^2 + \theta^2 \sigma_p^2}{\sigma_p^2 + \tau_1^2 + \theta^2} \end{bmatrix} \right)$$

③

(b)

3. (b) We want to maximize,

$$\sum \log P(z^{(i)}, y^{(i)} \mid x^{(i)}; \mu_p, V_z, \theta_p^2, \tau_r^2)$$

$$= \sum \log \left(P(z^{(i)} \mid x^{(i)}) \cdot P(y^{(i)} \mid x^{(i)})\right)$$

$$= \sum \log \frac{1}{\sqrt{2\pi\theta_p^2}} \exp\left(-\frac{1}{2\theta_p^2}(y-\mu_p)^2\right) \cdot \frac{1}{\sqrt{2\pi\tau_r^2}} \exp\left(-\frac{1}{2\tau_r^2}(z-V_z)^2\right)$$

$$= \sum \left(\log \frac{1}{2\pi\theta_p\tau_r} - \frac{1}{2\theta_p^2}(y^2 - 2y\mu_p + \mu_p^2) - \frac{1}{2\tau_r^2}(z^2 - 2zV_z + V_z^2)\right)$$

$$= \sum \left(\log \frac{1}{2\pi\theta_p\tau_r} - \frac{1}{2\theta_p^2}(E(y^2) - 2E(y)\cdot\mu_p + \mu_p^2) - \frac{1}{2\tau_r^2}(E(z^2) - 2E(z)V_z + V_z^2)\right)$$

From (3) we know

$$E(y^2) = \frac{\theta_p^2(\tau_r^2+\theta^2)}{\theta_p^2+\tau_r^2+\theta^2} \qquad E(y) = \mu_p + \frac{\theta_p^2}{\theta_p^2+\tau_r^2+\theta^2}(x-\mu_p-V_z)$$

$$E(z^2) = \frac{\tau_r^2(\theta_p^2+\theta^2)}{\theta_p^2+\tau_r^2+\theta^2} \qquad E(z) = V_z + \frac{\tau_r^2}{\theta_p^2+\tau_r^2+\theta^2}(x-\mu_p-V_z)$$

Let $(A) = \sum_{i=1}^{P} -\frac{1}{2\theta_p^2}(E(y^2) - 2E(y)\cdot\mu_p + \mu_p^2)$

$$\frac{\partial (A)}{\partial \mu_p} = -\frac{1}{2\theta_p^2}\sum_{i=1}^{P}(-2E(y) + 2\mu_p) = 0 \Rightarrow \frac{1}{\theta_p^2}\sum_{i=1}^{P}E(y) - 2P\mu_p = 0$$

$$\Rightarrow \boxed{\mu_p = \frac{1}{P}\sum_{i=1}^{P}E(y)}$$

Let $\frac{\partial A}{\partial \sigma_p^2} = 0$ $\sigma_p^2 = ?$

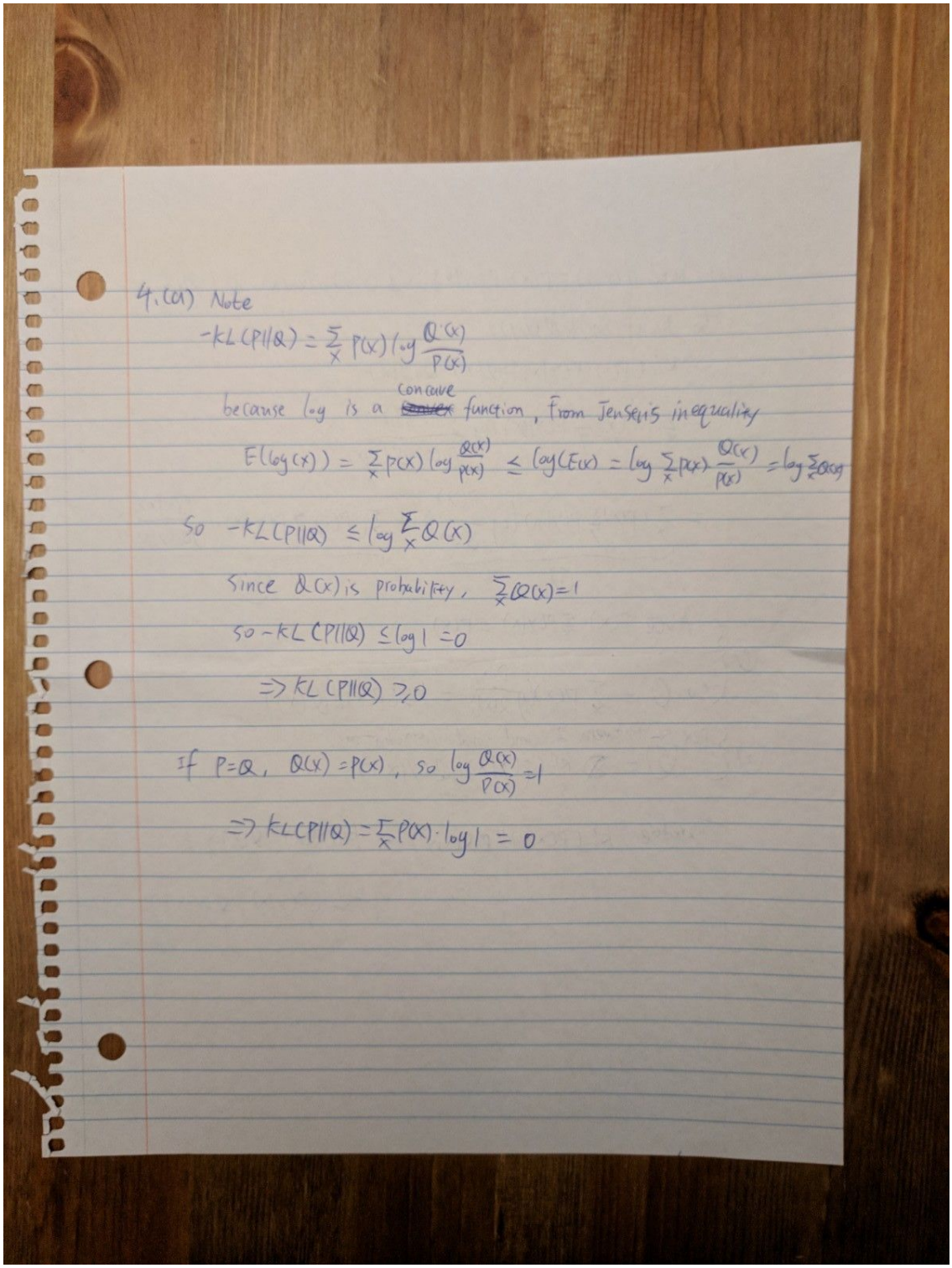
Let $B = \sum_{i=1}^{R} - \frac{1}{2\sigma_p^2}(\quad - \quad 2E(z)V_2 + V_2^2)$

Let $\frac{\partial B}{\partial V_2} = 0 \Rightarrow V_2 = \frac{1}{R}\sum_{i=1}^{R} E(z)$

Let $\frac{\partial B}{\partial \quad} = 0 \Rightarrow \quad \sigma_p^2 = ?$

4.

(a)



4.(a) Note

$$-KL(P\|Q) = \sum_x P(x) \log \frac{Q(x)}{P(x)}$$

because log is a concave function, from Jensen's inequality

$$E(\log(x)) = \sum_x P(x) \log \frac{Q(x)}{P(x)} \le \log(E(x)) = \log \sum_x P(x) \frac{Q(x)}{P(x)} = \log \sum_x Q(x)$$

So $-KL(P\|Q) \le \log \sum_x Q(x)$

Since $Q(x)$ is probability, $\sum_x Q(x) = 1$

So $-KL(P\|Q) \le \log 1 = 0$

$\Rightarrow KL(P\|Q) \ge 0$

If $P=Q$, $Q(x) = P(x)$, so $\log \frac{Q(x)}{P(x)} = 1$

$\Rightarrow KL(P\|Q) = \sum_x P(x) \cdot \log 1 = 0$

(b)

(b). Note $P(x,y) = P(x) \cdot P(y|x)$, $Q(x,y) = Q(x) \cdot Q(y|x)$

So $KL(P(x,y) \| Q(x,y))$

$= KL(P(x) \cdot P(y|x) \| Q(x) \cdot Q(y|x))$

$= \sum_x \left(P(x) \sum_y P(y|x) \log \frac{P(x) \cdot P(y|x)}{Q(x) \cdot Q(y|x)} \right)$

$= \underbrace{\sum_x \left(P(x) \sum_y P(y|x) \log \frac{P(x)}{Q(x)} \right)}_{①} + \underbrace{\sum_x \left(P(x) \sum_y P(y|x) \cdot \log \frac{P(y|x)}{Q(y|x)} \right)}_{②}$

① Note $P(x) \cdot \sum_y P(y|x) = P(x)$

So ① $= \sum_x P(x) \log \frac{P(x)}{Q(x)} = KL(P(x) \| Q(x))$

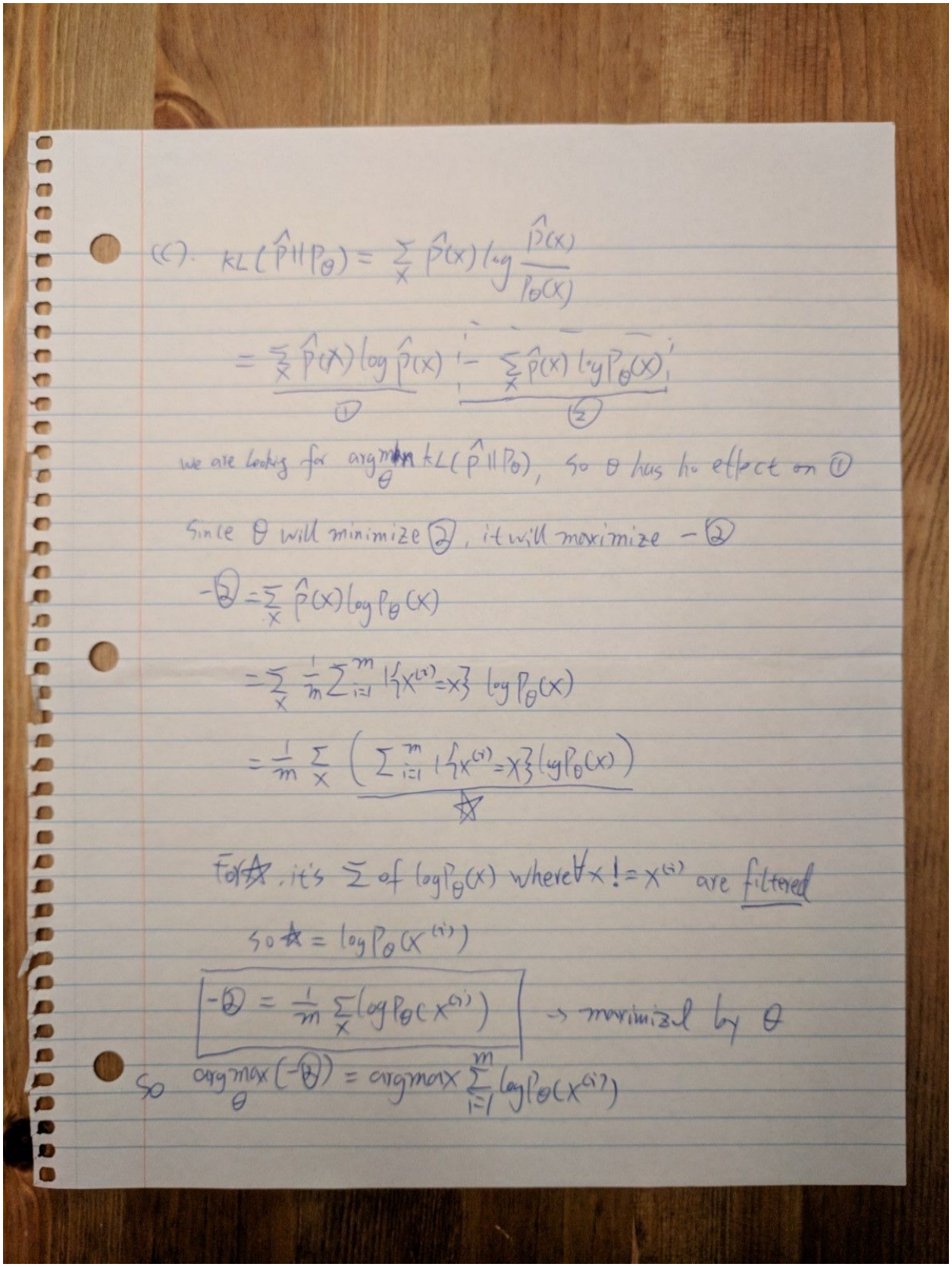
② per KL between 2 conditional distributions,

② $= KL(P(Y|X) \| Q(Y|X))$

Therefore $KL(P(x,y) \| Q(x,y)) = KL(P(x) \| Q(x)) +$

$KL(P(y|x) \| Q(y|x))$

(c)


(c). $KL(\hat{P} \| P_\theta) = \sum_x \hat{P}(x) \log \frac{\hat{P}(x)}{P_\theta(x)}$
$= \sum_x \hat{P}(x) \log \hat{P}(x) - \sum_x \hat{P}(x) \log P_\theta(x)$
① ②
we are looking for $\arg\min_\theta KL(\hat{P} \| P_\theta)$, so θ has no effect on ①
Since θ will minimize ②, it will maximize $-②$
$-② = \sum_x \hat{P}(x) \log P_\theta(x)$
$= \sum_x \frac{1}{m} \sum_{i=1}^m 1\{x^{(i)} = x\} \log P_\theta(x)$
$= \frac{1}{m} \sum_x \left(\sum_{i=1}^m 1\{x^{(i)} = x\} (\log P_\theta(x)) \right)$
☆
For ☆, it's $\sum$ of $\log P_\theta(x)$ where $\forall x \; != x^{(i)}$ are filtered
so ☆ $= \log P_\theta(x^{(i)})$
$-② = \frac{1}{m} \sum_x (\log P_\theta(x^{(i)}))$ → maximized by θ
So $\arg\max_\theta (-②) = \arg\max \sum_{i=1}^m \log P_\theta(x^{(i)})$

5.
(a)

```
def draw_image():
  A = imread('mandrill-large.tiff')
  plt.imshow(A)
  plt.show()
```

(b)

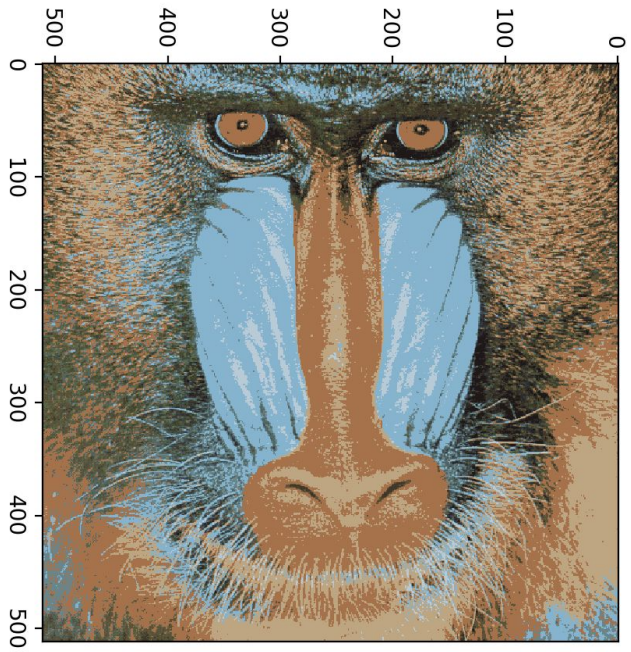
```
class Centroid:
  def __init__(self, image_matrix):
    self.r = random.randint(0, 255)
    self.g = random.randint(0, 255)
    self.b = random.randint(0, 255)
    # a list of index of the pixel assigned to this centroid
    self.assigned_pixels = set()
    self.image_matrix = image_matrix
    (self.height, self.width, c) = image_matrix.shape

  def add_pixel(self, pixel_index):
    self.assigned_pixels.add(pixel_index)

  def remove_pixel(self, pixel_index):
    self.assigned_pixels.remove(pixel_index)

  def contains_pixel(self, pixel_index):
    return pixel_index in self.assigned_pixels

  def update_rgb(self):
    # recalculate this centroids rgb value
    pixel_count = len(self.assigned_pixels)
    if pixel_count == 0:
      return
    red_sum = green_sum = blue_sum = 0
    for index in self.assigned_pixels:
      red_sum += self.get_red_value(index)
      green_sum += self.get_green_value(index)
      blue_sum += self.get_blue_value(index)
    self.r = red_sum / pixel_count
    self.g = green_sum / pixel_count
    self.b = blue_sum / pixel_count

  def distance_from_pixel(self, pixel_index):
    return (self.get_red_value(pixel_index) - self.r) * (self.get_red_value(pixel_index) - self.r) + \
        (self.get_green_value(pixel_index) - self.r) * (self.get_green_value(pixel_index) - self.g) + \
        (self.get_blue_value(pixel_index) - self.r) * (self.get_blue_value(pixel_index) - self.b)

  def get_red_value(self, pixel_index):
    return self.image_matrix[int(pixel_index / self.width), pixel_index % self.width, 0]

  def get_green_value(self, pixel_index):
    return self.image_matrix[int(pixel_index / self.width), pixel_index % self.width, 1]
```

```
  def get_blue_value(self, pixel_index):
    return self.image_matrix[int(pixel_index / self.width), pixel_index % self.width, 2]

  def shout(self):
    print("r: " + str(self.r))
    print("g: " + str(self.g))
    print("b: " + str(self.b))


# return if this pixel is assigned to a different centroid
def assign_pixel_to_centroid(pixel_index, centroids):
  min_cost = sys.maxsize
  new_index = -1
  for i, centroid in enumerate(centroids):
    cost = centroid.distance_from_pixel(pixel_index)
    if cost < min_cost:
      min_cost = cost
      new_index = i

  # remove from previous centroid and add to current centroid
  is_updated = False
  for i, centroid in enumerate(centroids):
    contains_index = centroid.contains_pixel(pixel_index)
    if i == new_index:
      is_updated = not contains_index
      centroid.add_pixel(pixel_index)
    else:
      if contains_index:
        centroid.remove_pixel(pixel_index)

  return is_updated


# problem 5.b
def k_means(k, image_matrix):
  centroids = []
  for i in range(k):
    centroids.append(Centroid(image_matrix))
  (height, width, color) = image_matrix.shape

  pixel_updated = 1
  iterations = 0
  max_iteration = 30
  # convergence check is set to no pixel is updated
  while pixel_updated > 0 and iterations < max_iteration:
    print("pixel_updated: " + str(pixel_updated))
    iterations += 1
    pixel_updated = 0
    for i in range(height):
      for j in range(width):
        pixel_index = i * width + j
        # count # of pixel updated
        if assign_pixel_to_centroid(pixel_index, centroids):
          pixel_updated += 1
    for centroid in centroids:
      centroid.update_rgb()
  if pixel_updated == 0:
    print("converges in " + str(iterations) + " rounds")
  else:
    print("terminates after " + str(max_iteration) + " rounds")
```

```
  return centroids

def update_picture(centroids, image_matrix):
  (height, width, c) = image_matrix.shape

  for centroid in centroids:
    for pixel_index in centroid.assigned_pixels:
      y = int(pixel_index / width)
      x = pixel_index % width
      image_matrix[x, y, 0] = centroid.r
      image_matrix[x, y, 1] = centroid.g
      image_matrix[x, y, 2] = centroid.b


# problem 5.b
def run_k_means():
  # cluster R3 (rgb) to k=16 clusters
  # a) initialize 16 centroid of R3(0-255, 0-255, 0-255) u = int[16]
  # b) for each pixel x: repeat until no centroid moves, no new assignment to existing points
  #   i) find which cluster this pixel belongs to by minimize this cost function:
  #     J = sum((xr-ur)^2 + (xg-ug)^2 + (xb-ub)^2)
  #   ii) update each centroid u by taking the average
  #     let Xs be the pixels assigned to centroid ui
  #     for each ui
  #       ui_r = sum(Xs_r)/len(Xs)
  #       ui_g = sum(Xs_g)/len(Xs)
  #       ui_b = sum(Xs_b)/len(Xs)
  #
  #   Xs_r = sum of r value of each X in Xs

  # A = imread('mandrill-small.tiff')
  # sample outputs
  # pixel_updated: 1
  # pixel_updated: 16384
  # pixel_updated: 15263
  # pixel_updated: 8300
  # pixel_updated: 6210
  # pixel_updated: 5682
  # pixel_updated: 4929
  # pixel_updated: 8413
  # pixel_updated: 5804
  # pixel_updated: 4751
  # pixel_updated: 5533
  # pixel_updated: 6313
  # pixel_updated: 5010
  # pixel_updated: 4593
  # pixel_updated: 5795
  # pixel_updated: 6323
  # pixel_updated: 4351
  # pixel_updated: 5171
  # pixel_updated: 5584
  # pixel_updated: 5355
  # pixel_updated: 4202
  # pixel_updated: 4930
  # pixel_updated: 5971
  # pixel_updated: 5217
  # pixel_updated: 4784
  # pixel_updated: 5400
  # pixel_updated: 5999
```

```
# pixel_updated: 4549
# pixel_updated: 4939
# pixel_updated: 5027
# terminates after 30 rounds

# A = imread('mandrill-large.tiff')
A = imread('mandrill-large.tiff')
A.setflags(write=1)


centroids = k_means(16, A)


update_picture(centroids, A)

plt.imshow(A)
plt.show()
```

(c)



```
#problem 5.c
def update_picture(centroids, image_matrix):
  (height, width, c) = image_matrix.shape

  for centroid in centroids:
    for pixel_index in centroid.assigned_pixels:
      y = int(pixel_index / width)
      x = pixel_index % width
      image_matrix[x, y, 0] = centroid.r
      image_matrix[x, y, 1] = centroid.g
      image_matrix[x, y, 2] = centroid.b
```

(d)