Machine Learning HW5

Q1

We first write down the negative logarithm of the likelihood function

$$E(w, \Sigma) = \frac{1}{2} \sum_{n=1}^{N} \{[y(x_n, w) - t_n]^T \Sigma^{-1} [y(x_n, w) - t_n]\} + const \tag{1}$$

The Σ is unknown and const denotes the term independent of both w and Σ . In the first situation, if Σ is fixed and known, the equation above will reduce to:

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} \{[y(x_n, w) - t_n]^T \Sigma^{-1} [y(x_n, w) - t_n]\} + const \tag{2}$$

We can simply solve w_{ML} by minimizing it. If Σ is unknown, since Σ is in the first term on the right of (∗), solving w_{ML} will involve Σ. Note that in the previous problem, the main reason that they can decouple is due to the independence assumption, i.e., Σ reduces to β^{-1}I, so that we can bring β to the front and view it as a fixed multiplying factor when solving wML.

Q2

We know that the logistic sigmoid function $\sigma(a) \in [0, 1]$, therefore if we perform a linear transformation $h(a) = 2\sigma(a) - 1$, we can find a mapping function $h(a)$ from $(-\infty, +\infty)$ to [-1, 1]. In this case, the conditional distribution of targets given inputs can be similarly written as:

$$p(t|x, w) = [\frac{1 + y(x, w)}{2}]^{(1+t)/2} [\frac{1 - y(x, w)}{2}]^{(1-t)/2} \tag{3}$$

Where $[1 + y(x, w)]/2$ represents the conditional probability $p(C_1|x)$. Since now $y(x, w) \in [-1, 1]$, we also need to perform the linear transformation to make it satisfy the constraint for probability. Then we can further obtain:

$$
\begin{aligned}
E(w) &= -\sum_{n=1}^{N}\{\frac{1+t_n}{2}\ln\frac{1+y_n}{2}+\frac{1-t_n}{2}\ln\frac{1-y_n}{2}\} \\
&= -\frac{1}{2}\sum_{n=1}^{N}\{(1+t_n)\ln(1+y_n)+(1-t_n)\ln(1-y_n)\}+N\ln 2
\end{aligned} \tag{4}
$$

Q3

(a)

$$E[t] = \int tN(t|\mu,\sigma^2 I)dt = \mu \tag{5}$$

And

$$E[\|t\|^2] = \int \|t\|^2 N(t|\mu,\sigma^2 I)dt = L\sigma^2 + \|\mu\|^2 \tag{6}$$

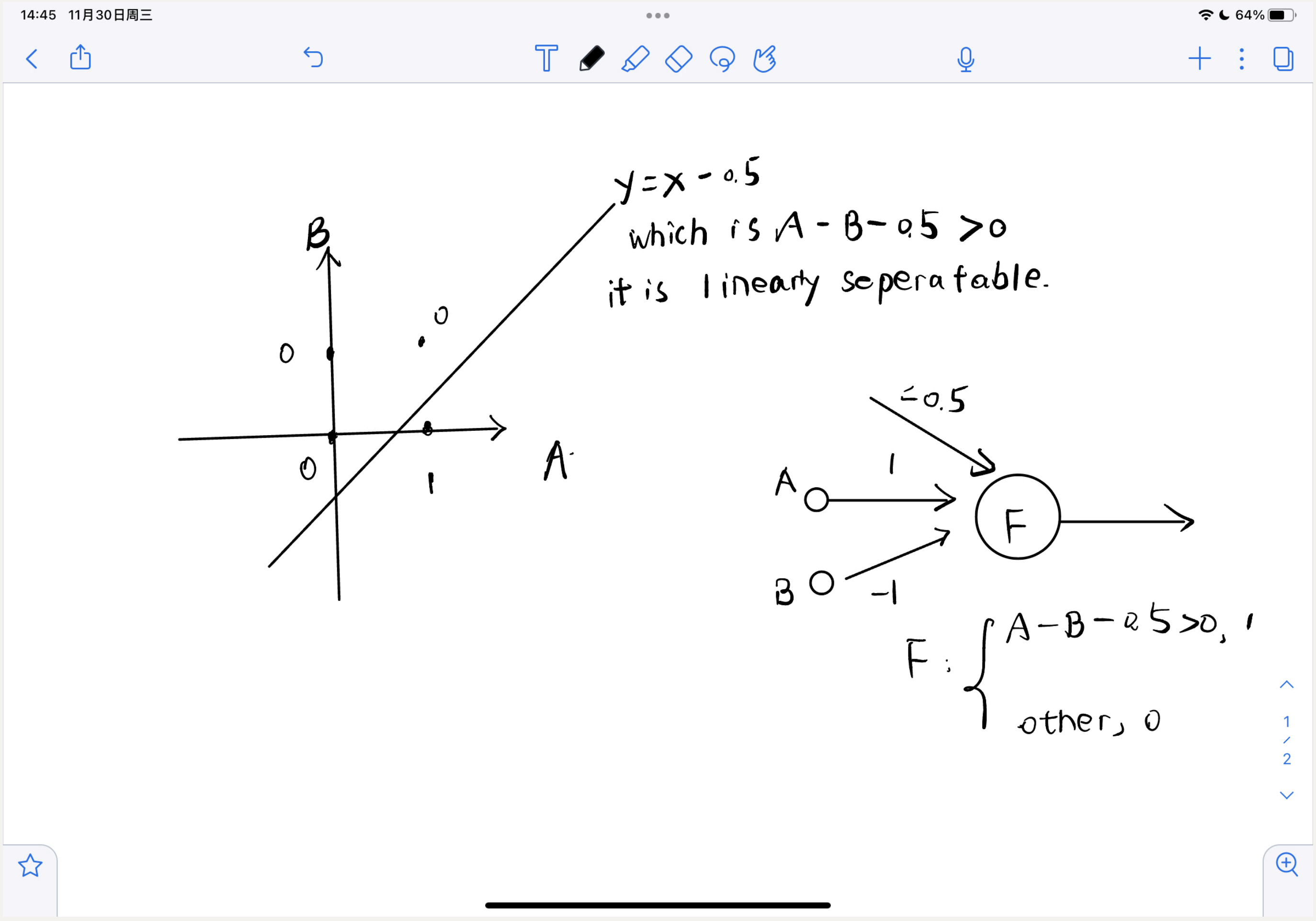
Here L is the dimension of rm we can write.

$$
\begin{aligned}
E[t|x] &= \int tp(t|x)dt \\
&= \int t\sum_{k=1}^{K}\pi_k N(t|\mu_k,\sigma^2)dt \\
&= \sum_{k=1}^{K}\pi_k\int tN(t|\mu_k,\sigma_k^2)dt \\
&= \sum_{k=1}^{K}\pi_k\mu_k
\end{aligned} \tag{7}
$$

(b)

$$
\begin{aligned}
s^2(x) &= E[\|t - E[t|x]\|^2|x] = E[(t^2 - 2tE[t|x] + E[t|x]^2)|x] \\
&= E[t^2|x] - E[2tE[t|x]|x] + E[t|x]^2 = E[t^2|x] - E[t|x]^2 \\
&= \int \|t\|^2 \sum_{k=1}^{K} \pi_k N(\mu_k, \sigma_k^2) dt - \| \sum_{l=1}^{K} \pi_l \mu_l \| \\
&= \sum_{k=1}^{K} \pi_k (L\sigma_k^2 + \|\mu_k\|^2) - \| \sum_{l=1}^{K} \pi_l \mu_l \|^2 \\
&= L \sum_{k=1}^{K} \pi_k \sigma_k^2 + \sum_{k=1}^{K} \pi_k \|\mu_k\|^2 \\
&\quad -2(\sum_{l=1}^{K} \pi_l \mu_l)(\sum_{k=1}^{K} \pi_k \mu_k) + \sum_{k=1}^{K} \pi_k \| \sum_{l=1}^{K} \pi_l \mu_l \| \\
&= L \sum_{k=1}^{K} \pi_k \sigma_k^2 + \sum_{k=1}^{K} \pi_k \|\mu_k - \sum_{l=1}^{K} \pi_l \mu_l\|^2 \\
&= \sum_{k=1}^{K} \pi_k (L\sigma_k^2 + \|\mu_k - \sum_{l=1}^{K} \pi_l \mu_l\|^2)
\end{aligned} \tag{8}
$$

Q4



Q5

(a) $4 * 4 * 3 = 48$

(b)(10/2) * (10/2) * 3 = 75

(c) 48 + 5*5*3*4=348

(d) it's right

(e) It needs to learn more parameters.

Q6

★ **SOLUTION:** Connect the input for X_1 to the output unit with a weight $C * (w_5 * w_1 + w_6 * w_2)$, and connect the input for X_2 to the output unit with weight $C(w_5 * w_3 + w_6 * w_4)$. Then the output unit can use the same activation function it used originally.

(a) Connect the input for X_1 to the output unit with a weight $C * (w_5 * w_1 + w_6 * w_2)$, and connect the input for X_2 to the output unit with weight $C(w_5 * w_3 + w_6 * w_4)$. Then the output unit can use the same activation function it used originally.

(b)This is true. Each layer can be thought of as performing a matrix multiply to find its representation given the representation on the layer that it receives input from. Thus the entire netowrk just performs a chain of matrix multiplies, and therefore we can simply multiply the matrices together to find the weights to represent the function with a single layer.

(c) One solution: w1 = w3 = −10,w2 = w4 = −1,w5 = 5, and w6 = −6. The intuition here is that we can decompose A XOR B into (A OR B) AND NOT (A AND B). We make the upper hidden unit behave like an OR by making it saturate when either of the input units are 1. It isn't possible to make a hidden unit that behaves exactly like AND, but we can at least make the lower hidden unit continue to increase in activation after the upper one has saturated.