Solutions for week 03

1. Using the table of joint probabilities, we can explicitly compute the joint distribution $p(a, b)$ for all possible values of the variables a and b and the product of their marginals:

a	b	$p(a, b) \times 1000$	$p(a) \times 1000$	$p(b) \times 1000$	$p(a)p(b) \times 1000$
0	0	336	600	592	355.2
0	1	264	600	408	244.8
1	0	256	400	592	236.8
1	1	144	400	408	163.2

   Clearly $p(a, b) \neq p(a)p(b)$.

   Similarly we can check for conditional independence given c by writing down a similar table for conditional joint and marginal distributions.

c	a	b	$p(a, b\|c) \times 1000$	$p(a\|c) \times 1000$	$p(b\|c) \times 1000$	$p(a\|c)p(b\|c) \times 1000$
0	0	0	400	500	800	400
0	0	1	100	500	200	100
0	1	0	400	500	800	400
0	1	1	100	500	200	100
1	0	0	276.9	692.3	400	276.9
1	0	1	415.4	692.3	600	415.4
1	1	0	123.1	307.7	400	123.1
1	1	1	184.6	307.7	600	184.6

   Observing the table, $p(a, b|c) = p(a|c)p(b|c)$ for all values of a, b and c.

2. Consider a formulation of the logical-OR function over the set of binary variables $x_i \in \{0, 1\}$ where $i \in \{1, \ldots, M\}$.

$$y = 1 - \prod_{i=1}^{M} I_0(x_i)$$

   where I_0 is an indicator function defined by

$$\begin{aligned} I_0(x) &= 1, x = 0 \\ &= 0, x = 1 \end{aligned}$$

It is easy to see that the product term goes to zero when at least one of the x_i is 1 leading to a logical OR function. This can be converted to a soft-OR function by considering each x_i as a binary random variable taking values in {0, 1} with $p(x_i = 1) = \mu_i$. By replacing the indicator functions with the probabilities $p(x_i = 0) = (1 - \mu_i)$, we obtain a function which looks similar to the above. But the left hand side would now indicate the probability of at least one of the x_i assuming 1. This is because the second term is the probability of all of the x_i being zero assuming they are independent. The noisy-OR function is obtained by considering only those variables in the product term whose value has been observed to be 1. Then the left hand side would denote the probability of observing at least one of those x_i to be 1. Further an additional binary random variable x_0 is introduced whose value is always forced to 1. But it can be interpreted as having the probability $p(x_0 = 1) = \mu_0$. By including this additional random variable and writing out the soft version of the OR function

$$p(y = 1|x_0, x_1, \dots, x_M) = 1 - p(x_0 = 0)^1 \prod_{i=1}^{M} p(x_i = 0)^{x_i}$$

Since x_0 is always forced to 1, we obtain a noisy soft-OR function as

$$p(y = 1|x_1, \dots, x_M) = 1 - (1 - \mu_0) \prod_{i=1}^{M} (1 - \mu_i)^{x_i}$$

When all the x_i are zero, then $p(y = 1|x_1, \dots, x_M) = \mu_0$. It is nothing but the probability of getting atleast one 1 among all the observed x_i that are 1. In this case, the only variable x_0 is 1 and hence the result. The value of μ_0 can also be interpreted as the value of the noisy-OR function when all the variables $x_i \in \{1, \dots, M\}$ are observed to be 0.

Solution from a second student is in figure 1.

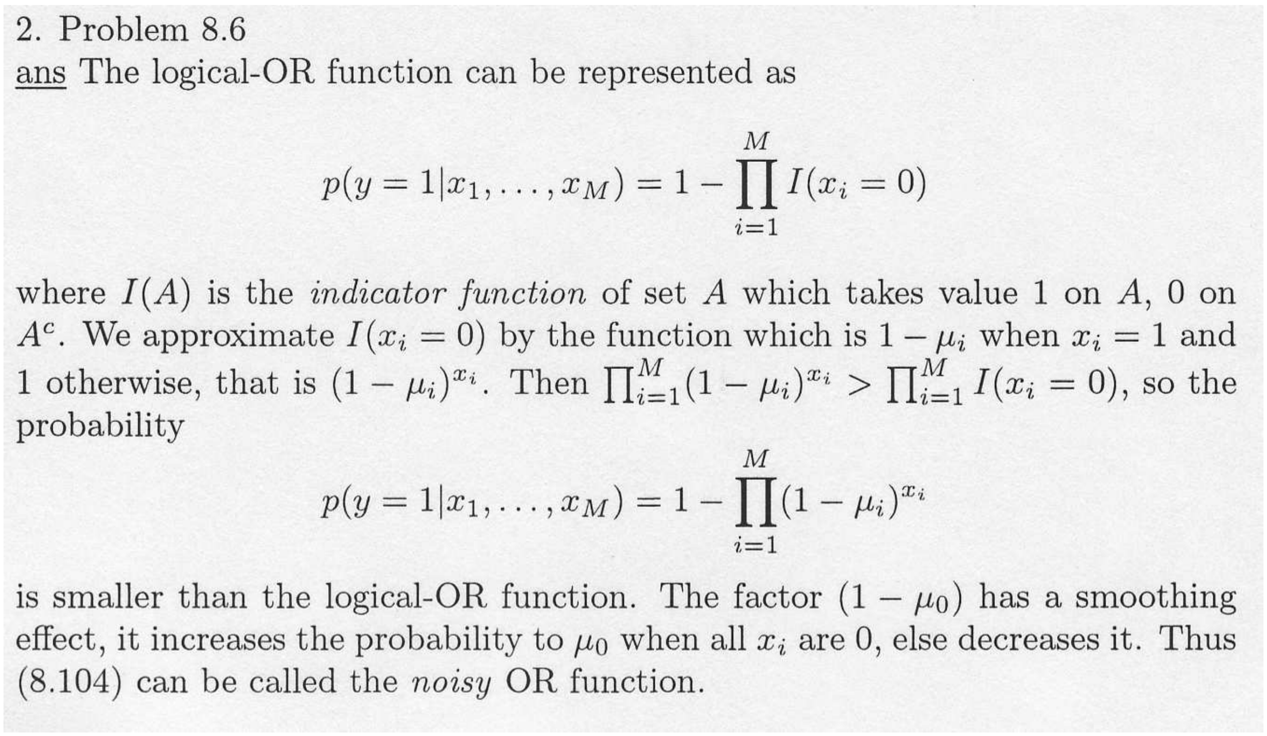
Solution from a third student is in figure 2.

3. The joint distribution implied by the graph is given by

$$p(a, b, c, d) = p(a)p(b)p(c|a, b)p(d|c)$$

The distribution $p(a, b)$ is obtained by marginalizing the above joint distribution w.r.t. the variables c and d

$$\begin{aligned} p(a, b) &= \sum_c \sum_d p(a, b, c, d) \\ &= \sum_c \sum_d p(a)p(b)p(c|a, b)p(d|c) \\ &= p(a)p(b) \sum_c p(c|a, b) \sum_d p(d|c) \\ &= p(a)p(b) \sum_c p(c|a, b) \\ &= p(a)p(b) \end{aligned}$$



2. Problem 8.6

<u>ans</u> The logical-OR function can be represented as

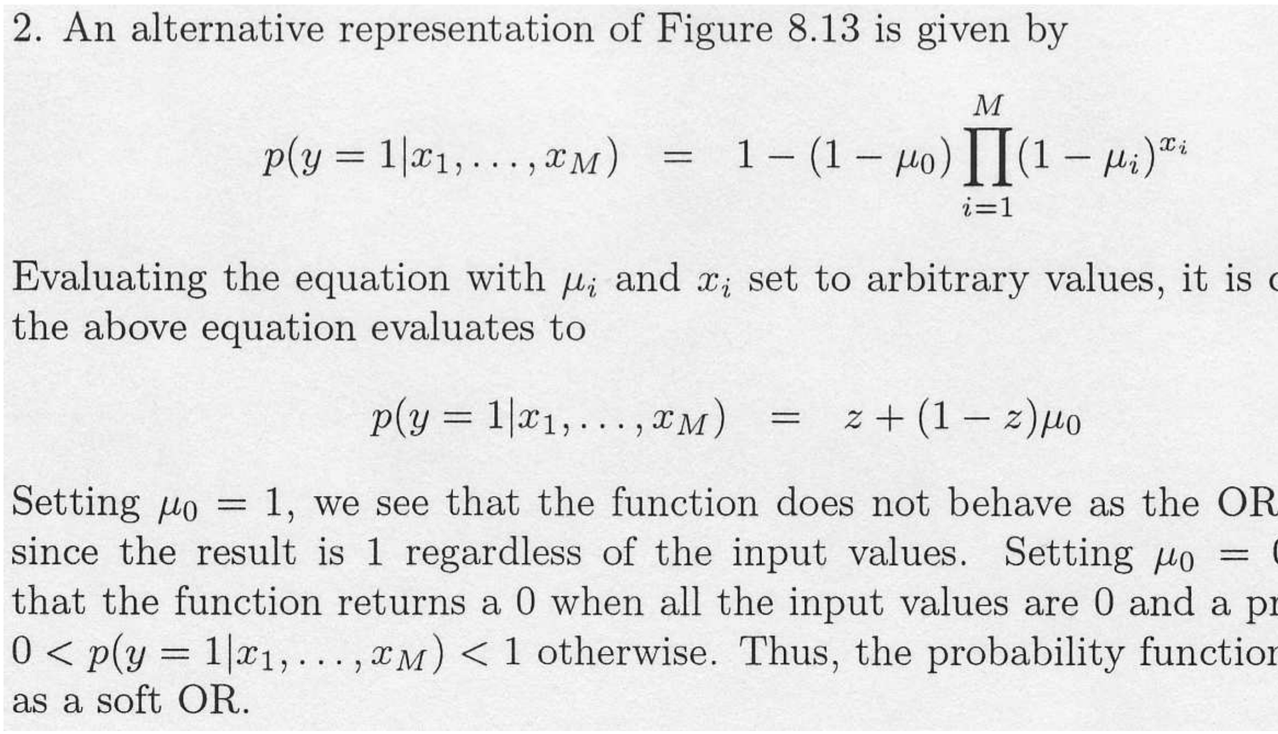
$$p(y=1|x_1,\dots,x_M) = 1 - \prod_{i=1}^{M} I(x_i = 0)$$

where $I(A)$ is the *indicator function* of set A which takes value 1 on A, 0 on A^c. We approximate $I(x_i = 0)$ by the function which is $1-\mu_i$ when $x_i = 1$ and 1 otherwise, that is $(1-\mu_i)^{x_i}$. Then $\prod_{i=1}^{M}(1-\mu_i)^{x_i} > \prod_{i=1}^{M} I(x_i = 0)$, so the probability

$$p(y=1|x_1,\dots,x_M) = 1 - \prod_{i=1}^{M}(1-\mu_i)^{x_i}$$

is smaller than the logical-OR function. The factor $(1-\mu_0)$ has a smoothing effect, it increases the probability to μ_0 when all x_i are 0, else decreases it. Thus (8.104) can be called the *noisy* OR function.

Figure 1: An approach to problem 2.


2. An alternative representation of Figure 8.13 is given by

$$p(y=1|x_1,\ldots,x_M) \quad = \quad 1-(1-\mu_0)\prod_{i=1}^{M}(1-\mu_i)^{x_i}$$

Evaluating the equation with μ_i and x_i set to arbitrary values, it is
the above equation evaluates to

$$p(y=1|x_1,\ldots,x_M) \quad = \quad z+(1-z)\mu_0$$

Setting $\mu_0 = 1$, we see that the function does not behave as the OR
since the result is 1 regardless of the input values. Setting $\mu_0 =$
that the function returns a 0 when all the input values are 0 and a p
$0 < p(y=1|x_1,\ldots,x_M) < 1$ otherwise. Thus, the probability functio
as a soft OR.

Figure 2: An approach to problem 2.

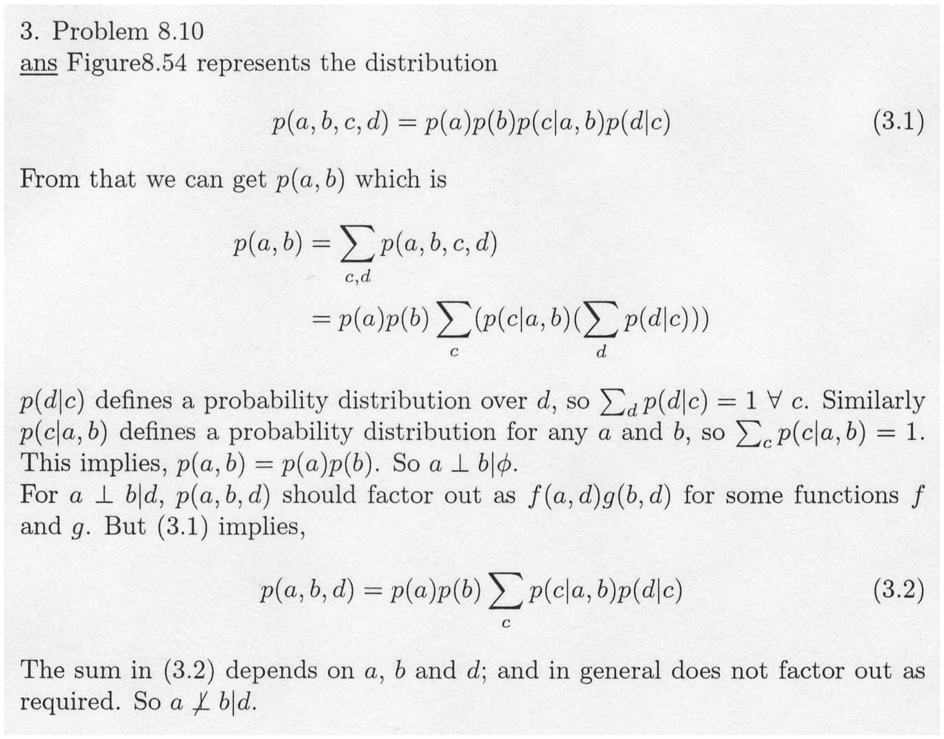
Therefore $a \perp\!\!\!\perp b \mid \emptyset$.

Similarly the conditional joint distribution $p(a,b|d)$ can be obtained as

$$\begin{aligned}
p(a,b|d) &= \frac{p(a,b,d)}{p(d)} \\
&= \frac{\sum_c p(a,b,c,d)}{p(d)} \\
&= \frac{\sum_c p(a)p(b)p(c|a,b)p(d|c)}{p(d)} \\
&= \frac{p(a)p(b)\sum_c p(c|a,b)p(d|c,a,b)}{p(d)} \\
&= \frac{p(a)p(b)\sum_c p(c,d|a,b)}{p(d)} \\
&= \frac{p(a)p(b)p(d|a,b)}{p(d)}
\end{aligned}$$

So in general, $p(a,b|d) \neq p(a|d)p(b|d)$. Hence $a \not\perp\!\!\!\perp b \mid d$

Solution from a second student is in figure 3.



3. Problem 8.10

<u>ans</u> Figure8.54 represents the distribution

$$p(a,b,c,d) = p(a)p(b)p(c|a,b)p(d|c) \tag{3.1}$$

From that we can get $p(a,b)$ which is

$$\begin{aligned} p(a,b) &= \sum_{c,d} p(a,b,c,d) \\ &= p(a)p(b)\sum_{c}(p(c|a,b)(\sum_{d} p(d|c))) \end{aligned}$$

$p(d|c)$ defines a probability distribution over d, so $\sum_d p(d|c) = 1 \ \forall \ c$. Similarly $p(c|a,b)$ defines a probability distribution for any a and b, so $\sum_c p(c|a,b) = 1$. This implies, $p(a,b) = p(a)p(b)$. So $a \perp b|\phi$.

For $a \perp b|d$, $p(a,b,d)$ should factor out as $f(a,d)g(b,d)$ for some functions f and g. But (3.1) implies,

$$p(a,b,d) = p(a)p(b)\sum_{c} p(c|a,b)p(d|c) \tag{3.2}$$

The sum in (3.2) depends on a, b and d; and in general does not factor out as required. So $a \not\perp b|d$.

Figure 3: An approach to problem 3.

4. Writing down the expression for the complete energy function

$$E(\mathbf{x}, \mathbf{y}) = h \sum_i x_i - \beta \sum_{\{i,j\}} x_i x_j - \eta \sum_i x_i y_i$$

Retaining only the terms that contain x_j as a factor

$$E_j = hx_j - x_j(\beta \sum_k x_k) - \eta x_j y_j$$

where k goes over all the neighbors of x_j. The difference between the energy function for the two different values of x_j (+1, -1) is given by

$$\begin{aligned}\Delta E &= E_j|_{x_j=1} - E_j|_{x_j=-1} \\ &= 2h - 2\beta \sum_k x_k - 2\eta y_j\end{aligned}$$

The above expression shows that the difference in the values of the energy function associated with two states of a given variable x_j depends only on the quantities that are local to x_j.

5. **Bayes Classifier Method**

Bayes' theorem specifies that the posterior is proportional to likelihood times prior,

$$p(C|\mathbf{d}) = \frac{p(\mathbf{d}|C) \cdot p(C)}{p(\mathbf{d})}$$

To construct our Naïve-Bayes classifier, we assume that the grid cells are conditionally independent given the class ("face" or "no-face") of the training data. We define a likelihood function $p(C|d_i)$ for each cell, presuming Gaussian distribution of the data in each of the $7x7 = 49$ cells. This yields a total of 98 sets of conditional Gaussian parameters. The parameters are determined by a maximum-likelihood fitting of the Gaussian to the data points in the 100 training samples in each class. That is, we fit the Gaussian to the 100 sample points corresponding to each of the 49 cells of the "face" and "no face" training groups, respectively. In the case of max-likelihood, the best-fitting mean and variance are determined as simply the sample mean and variance of the training points in each cell. Our prior, $p(C)$, is defined trivially as the number of images falling into each class.

We can simply take the ratio of the posteriors for the "face" and "no face" classes, such that the normalization terms cancel. Also, working with the (natural) log-posterior avoids numerical difficulties due to limited machine precision. This yields the composite log-posterior ratio

$$\log \frac{p(C_f|\mathbf{d})}{p(C_n|\mathbf{d})} = \log \frac{p(\mathbf{d}|C_f)}{p(\mathbf{d}|C_n)} + \log \frac{p(C_f)}{p(C_n)}$$
$$= \log \frac{\sum_i^{cells} p(d_i|C_f)}{\sum_i^{cells} p(d_i|C_n)} + \log \frac{p(C_f)}{p(C_n)}$$

When this ratio is greater than 0, the given test image is considered a face.

Assumptions

(a) The most significant assumption is that of strong conditional independence: given the class of the data, all training images will be of that class. In other words, if our class is "face," our training data should represent only faces.

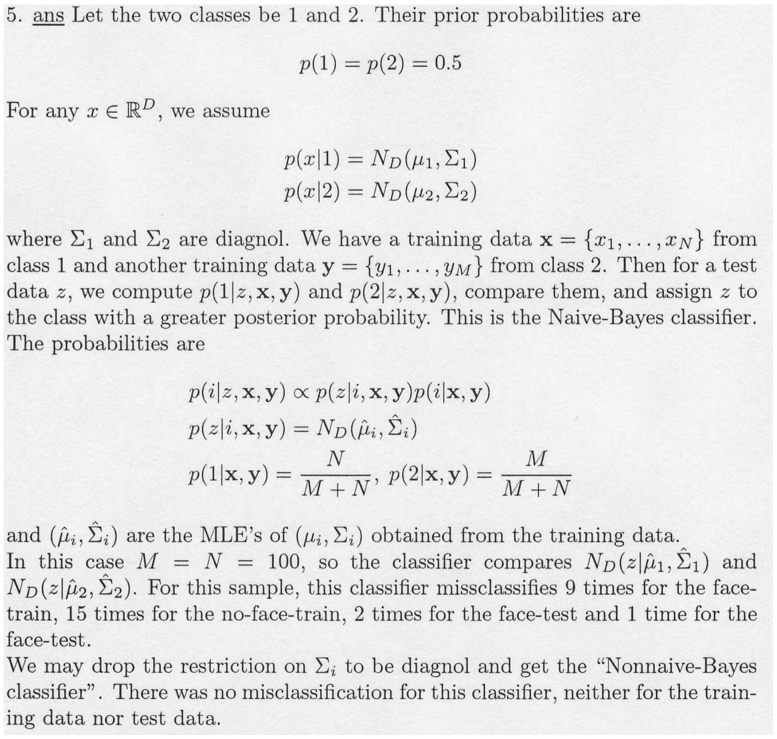
(b) As there are equal numbers of training points for the two classes, our prior is rather uninformative. We can only conjecture that either image class will be seen with 50% probability; this effectively cancels the influence of the prior in our Bayesian calculation.

(c) We assume a Gaussian density function is representative of the data for each cell, and that our maximum likelihood obtained from the training sample is a reasonable approximation of the "true," unobservable mean and variance of each cell.

Results

Our Naïve-Bayes classifier properly detected 11 of 13 faces in the "face" test group, while improperly detecting 1 face out of 13 samples in the "no face" test group. Upon running the classifier against the original training sets, we found that 9 of 100 samples in the "face" group were improperly classified, while 15 of 100 samples in the "no face" group were misclassified.

Solution from a second student is is in figure 4.



5. ans Let the two classes be 1 and 2. Their prior probabilities are

$$p(1) = p(2) = 0.5$$

For any $x \in \mathbb{R}^D$, we assume

$$p(x|1) = N_D(\mu_1, \Sigma_1)$$
$$p(x|2) = N_D(\mu_2, \Sigma_2)$$

where Σ_1 and Σ_2 are diagnol. We have a training data $\mathbf{x} = \{x_1, \ldots, x_N\}$ from class 1 and another training data $\mathbf{y} = \{y_1, \ldots, y_M\}$ from class 2. Then for a test data z, we compute $p(1|z, \mathbf{x}, \mathbf{y})$ and $p(2|z, \mathbf{x}, \mathbf{y})$, compare them, and assign z to the class with a greater posterior probability. This is the Naive-Bayes classifier. The probabilities are

$$p(i|z, \mathbf{x}, \mathbf{y}) \propto p(z|i, \mathbf{x}, \mathbf{y}) p(i|\mathbf{x}, \mathbf{y})$$
$$p(z|i, \mathbf{x}, \mathbf{y}) = N_D(\hat{\mu}_i, \hat{\Sigma}_i)$$
$$p(1|\mathbf{x}, \mathbf{y}) = \frac{N}{M+N}, \; p(2|\mathbf{x}, \mathbf{y}) = \frac{M}{M+N}$$

and $(\hat{\mu}_i, \hat{\Sigma}_i)$ are the MLE's of (μ_i, Σ_i) obtained from the training data.
In this case $M = N = 100$, so the classifier compares $N_D(z|\hat{\mu}_1, \hat{\Sigma}_1)$ and $N_D(z|\hat{\mu}_2, \hat{\Sigma}_2)$. For this sample, this classifier missclassifies 9 times for the face-train, 15 times for the no-face-train, 2 times for the face-test and 1 time for the face-test.
We may drop the restriction on Σ_i to be diagnol and get the "Nonnaive-Bayes classifier". There was no misclassification for this classifier, neither for the training data nor test data.

Figure 4: An approach to problem 5.