Problem 1

Through the proof shown at https://en.wikipedia.org/wiki/Maximum_likelihood_estimation#Relation_to_minimizing_Kullback%E2%80%93Leibler_divergence_and_cross_entropy, we know finding θ that maximizes likelihood is equivalent to finding $\arg\min_\theta D_{KL}(P_\theta || P_{\theta^0})$.

As such, we want $\arg\min_\theta D_{KL}(P_\theta || P_{\theta^0}) = \arg\min_\theta \sum_{i=1}^{N} P(y^i|x^i,\theta) \log \frac{P(y^i|x^i,\theta)}{P(y^i|x^i,\theta^0)}$

The minimum value of KL-divergence is 0.

To get a KL-divergence of 0, we need the value in the logarithm to be 1. In other words:

$\frac{P(y^i|x^i,\theta)}{P(y^i|x^i,\theta^0)} = 1$
$\implies \log \frac{P(y^i|x^i,\theta)}{P(y^i|x^i,\theta^0)} = 0$
$\implies P(y^i|x^i,\theta) \log \frac{P(y^i|x^i,\theta)}{P(y^i|x^i,\theta^0)} = 0$
$\implies \sum_{i=1}^{N} P(y^i|x^i,\theta) \log \frac{P(y^i|x^i,\theta)}{P(y^i|x^i,\theta^0)} = 0$

As such, if we have $\theta = \theta^0$, then we compute the KL-divergence as so:

$D_{KL}(P_\theta || P_{\theta^0})$
$= \sum_{i=1}^{N} P(y^i|x^i,\theta) \log \frac{P(y^i|x^i,\theta)}{P(y^i|x^i,\theta^0)}$
$= \sum_{i=1}^{N} P(y^i|x^i,\theta) \log \frac{\frac{P(x^i,y^i|\theta)}{P(x^i|\theta)}}{\frac{P(x^i,y^i|\theta^0)}{P(x^i|\theta^0)}}$
$= \sum_{i=1}^{N} P(y^i|x^i,\theta) \log \frac{P(x^i,y^i|\theta)P(x^i|\theta^0)}{P(x^i,y^i|\theta^0)P(x^i|\theta)}$
$= \sum_{i=1}^{N} P(y^i|x^i,\theta^0) \log \frac{P(x^i,y^i|\theta^0)P(x^i|\theta^0)}{P(x^i,y^i|\theta^0)P(x^i|\theta^0)}$ (substitute $\theta = \theta^0$)
$= \sum_{i=1}^{N} P(y^i|x^i,\theta^0) \log 1$
$= \sum_{i=1}^{N} P(y^i|x^i,\theta^0)(0)$
$= \sum_{i=1}^{N} 0$
$= 0$

As such, when $\theta = \theta^0$, we achieve the minimum KL-divergence possible, which is an optimum by definition. □

Problem 2

We have the "younger than 60" samples as follows: 0110 and 1110.

We have the "older than 60" samples as follows: 1000, 1001, 1111, 0001.

We produce MLE estimates of the probabilities as follows:

P(cornflakes | younger than 60) $= \frac{1}{2} = 0.5$

P(frosties | younger than 60) $= \frac{2}{2} = 1.0$

P(sugar puss | younger than 60) $= \frac{2}{2} = 1.0$

P(branflakes | younger than 60) $= \frac{0}{2} = 0.0$

P(younger than 60) $= \frac{2}{6} = \frac{1}{3}$

P(cornflakes | older than 60) $= \frac{3}{4} = 0.75$

P(frosties | older than 60) $= \frac{1}{4} = 0.25$

P(cornflakes | older than 60) $= \frac{1}{4} = 0.25$

P(branflakes | older than 60) $= \frac{3}{4} = 0.75$

P(older than 60) $= \frac{4}{6} = \frac{2}{3}$

So, we get the following probabilities of the sample 0110 belonging to each class as the following:

P(younger than 60 | 0110)

= P(not cornflakes | younger than 60)P(frosties | younger than 60)P(sugar puss | younger than 60)P(not branflakes | younger than 60)P(younger than 60)

$= 0.5(1)(1)(1)\frac{1}{3} = \frac{1}{6} = 0.1667$

P(older than 60 | 0110)

= P(not cornflakes | older than 60)P(frosties | older than 60)P(sugar puss | older than 60)P(not branflakes | older than 60)P(older than 60) $= 0.25(0.25)(0.25)(0.25)\frac{2}{3}$

$= \frac{2}{768} = 0.0026$

So, after normalizing the probabilities, we get:

P(younger than 60 | 0110) $= \frac{\text{P(younger than 60 | 0110)}}{\text{P(younger than 60 | 0110)+P(older than 60 | 0110)}} = 0.984615$

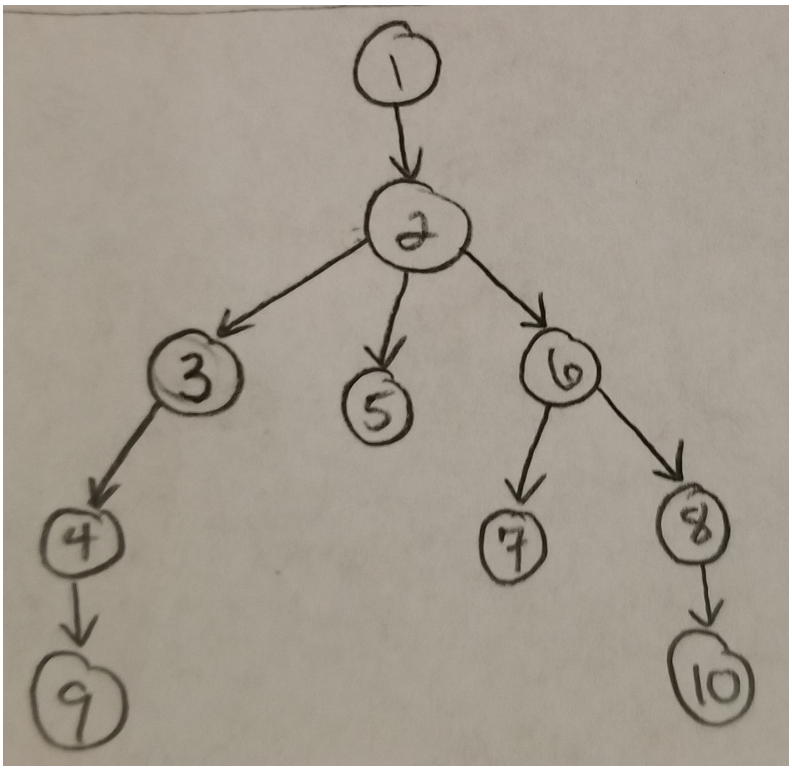
P(older than 60 | 0110) $= \frac{\text{P(older than 60 | 0110)}}{\text{P(younger than 60 | 0110)+P(older than 60 | 0110)}} = 0.015385$

As such, the probability that she is younger than 60 is **0.9846**.

Problem 3

To run my program, execute "q3.m".

Below is the maximum likelihood Chow Liu tree:



Problem 4

With a Naive Bayes classifier, we classify a sample to class 1, as opposed to class 0, if the following condition holds (let y represent "class"):

$\frac{P(y=1|x)}{P(y=0|x)} > 1$

As such, we can expand this as follows:

$\frac{P(y=1|x)}{P(y=0|x)} > 1$
$\implies \log \frac{P(y=1|x)}{P(y=0|x)} > \log(1)$
$\implies \log P(y=1|x) - \log P(y=0|x) > 0$
$\implies \log P(x|y=1)P(y=1) - \log P(x|y=0)P(y=0) > 0$
$\implies \log P(x|y=1) + \log P(y=1) - \log P(x|y=0) - \log P(y=0) > 0$
$\implies (\log P(x|y=1) - \log P(x|y=0)) + (\log P(y=1) - \log P(y=0)) > 0$
$\implies (\sum_{i=1}^{N} [x_i \log(\theta_i^1) + (1-x_i)\log(1-\theta_i^1)] - \sum_{i=1}^{N} [x_i \log(\theta_i^0) + (1-x_i)\log(1-\theta_i^0)]) + (\log P(y=1) - \log P(y=0)) > 0$
$\implies (\sum_{i=1}^{N} [x_i(\log(\theta_i^1) - \log(1-\theta_i^1)) + \log(1-\theta_i^1)] - \sum_{i=1}^{N} [x_i(\log(\theta_i^0) - \log(1-\theta_i^0)) + \log(1-\theta_i^0)]) + (\log P(y=1) - \log P(y=0)) > 0$
$\implies (\sum_{i=1}^{N} [x_i(\log(\theta_i^1) - \log(\theta_i^0) - \log(1-\theta_i^1) + \log(1-\theta_i^0)) + \log(1-\theta_i^1) - \log(1-\theta_i^0)] + (\log P(y=1) - \log P(y=0)) > 0$
$\implies (\sum_{i=1}^{N} \left[x_i(\log(\frac{\theta_i^1(1-\theta_i^0)}{\theta_i^0(1-\theta_i^1)})) + \log(1-\theta_i^1) - \log(1-\theta_i^0)\right] + (\log P(y=1) - \log P(y=0)) > 0$
$\implies \sum_{i=1}^{N} \left[x_i(\log(\frac{\theta_i^1(1-\theta_i^0)}{\theta_i^0(1-\theta_i^1)}))\right] + (\log P(y=1) - \log P(y=0) + \sum_{i=1}^{N} [\log(1-\theta_i^1) - \log(1-\theta_i^0)]) > 0$
$\implies \log(\frac{\theta^1(1-\theta^0)}{\theta^0(1-\theta^1)})^T x + (\log P(y=1) - \log P(y=0) + \sum_{i=1}^{N} [\log(1-\theta_i^1) - \log(1-\theta_i^0)]) > 0$
$\implies w^T x + b > 0$

As we can see from the above equations, the decision to classify a data point x as class 1 holds if $w^T x + b > 0$ for some w and b.

Specifically, we get:

$w = \log(\frac{\theta^1(1-\theta^0)}{\theta^0(1-\theta^1)})$ (all operations are element-wise)

$b = (\log P(y=1) - \log P(y=0) + \sum_{i=1}^{N} [\log(1-\theta_i^1) - \log(1-\theta_i^0)])$. □

Problem 5

1)

We will derive expressions for the parameters of this model in terms of the training data using maximum likelihood. We have the following:

$P(c^1, \ldots, c^N, x^1, \ldots, x^N)$
$= \prod_{i=1}^{N} P(c^i, x^i)$
$= \prod_{i=1}^{N} \left[P(c^i) \prod_{j=1}^{D} P(x_j^i | c^i) \right]$

If we take the log probability to calculate the MLE, we get:

$\log P(c^1, \ldots, c^N, x^1, \ldots, x^N)$
$= \sum_{i=1}^{N} \log \left[P(c^i) \prod_{j=1}^{D} P(x_j^i | c^i) \right]$
$= \sum_{i=1}^{N} \left[\log P(c^i) + \sum_{j=1}^{D} \log P(x_j^i | c^i) \right]$
$= \sum_{i=1}^{N} \log P(c^i) + \sum_{i=1}^{N} \sum_{j=1}^{D} \log P(x_j^i | c^i) = L$

As such, to solve this optimization problem, we need to maximize L subject to:

$\sum_{i \in \{0,1\}} P(c^i) = 1$
$\sum_{w \in \{0,1\}}^{D} P(x_j^i = w) = 1 \ \ \forall i = 1, \ldots, C$ and $\forall j = 1, \ldots, D$

Claim. To solve this greater proof, we first must show the MLE estimate p_i of a multinomial distribution is $p_i = \frac{c_i}{N}$:

A multinomial distribution is defined as $P(x_1, \ldots, x_k | p_1, \ldots, p_k) = \frac{n!}{\prod_{i=1}^{K} x_i!} \prod_{i=1}^{K} p_i^{x_i}$. As such, we get the following:

$\log P(x_1, \ldots, x_k | p_1, \ldots, p_k)$
$= \log(n!) - \log(\prod_{i=1}^{K} x_i!) + \log(\prod_{i=1}^{K} p_i^{x_i})$
$= \log(n!) - \sum_{i=1}^{K} \log(x_i!) + \sum_{i=1}^{K} \log(p_i^{x_i})$

Then we get the following:

$\arg\max_p \log P(x|p)$
$= \arg\max_p \sum_{i=1}^{K} \log(p_i^{x_i})$
$= \arg\max_p \sum_{i=1}^{K} x_i \log(p_i)$

So, to get the MLE estimate of the multinomial distribution, we must solve the following optimization problem:

maximize $\sum_{y \in Y} c_y \log(p_y)$

s.t. $p_i \geq 0 \ \ \forall i$

$\sum_{w \in \{0,1\}} P(x_j^i = w) = 1 \ \ \forall i = 1, \ldots, C$ and $\forall j = 1, \ldots, D$

We can solve this using the lagrangian: $g(\lambda, p) = \sum_{y \in Y} c_y \log(p_y) - \lambda(\sum_{y \in Y} p_y - 1)$. We get the following:

$\frac{\delta g(\lambda,p)}{\delta p_i} = \frac{c_i}{p_i} - \lambda$
$\implies \frac{c_i}{p_i} - \lambda = 0$ (Setting derivative to 0)
$\implies p_i = \frac{c_i}{\lambda}$
$\implies p_i = \frac{c_i}{\sum_{y \in Y} c_y}$ (Normalizing to sum to 1 due to its constraint)
$\implies p_i = \frac{c_i}{N}$ **(*)**

As such, we have shown the MLE of a multinomial distribution is $p_i = \frac{c_i}{N}$. □

Now, to solve the remaining proof for the MLE of Naive Bayes. Earlier, we showed the log-likelihood for the MLE estimates was $L = \sum_{i=1}^{N} \log P(c^i) + \sum_{i=1}^{N} \sum_{j=1}^{D} \log P(x_j^i | c^i)$. We will reduce the log-likelihood even further:

$L = \sum_{i=1}^{N} \log P(c^i) + \sum_{i=1}^{N} \sum_{j=1}^{D} \log P(x_j^i | c^i)$
$= \sum_{y \in \{0,1\}} \text{count}(c_y) \log P(c_y) + \sum_{j=1}^{D} \sum_{y \in \{0,1\}} \sum_{w \in \{0,1\}} \text{count}(x_j = w | c_y) \log P(x_j = w | c_y)$

In the above, $\text{count}(c_i) = \sum_{j=1}^{N} I[c^j = c_i]$ and $\text{count}(x_j = w | c_y) = \sum_{i=1}^{M} I[c^i = c_y \text{ and } x_j^i = w]$.

Now, since we have simplified the log-likelihood to a summation of two terms, we can see in order to maximize the log-likelihood, we want to maximize each term.

To maximize the first term, we see it is in the form of the multinomial distribution optimization problem as shown above, so by (*) the MLE estimate is $P(c_i) = \frac{\text{count}(c_i)}{N}$.

To maximize the second term, we also see it is in the form of the multinomial distribution optimization problem, so by (*) the MLE estimate is $P(x_i = x | c_i) = \frac{\text{count}(x_i = x | c_i)}{\sum_{w \in \{0,1\}} \text{count}(x_i = w | c_i)}$.

In the above two sentences, we have derived expressions in order to calculate the parameters of this Naive Bayes model using the MLE. □

2)

To form the classifier $p(c|x)$, we can do the following:

$p(c|x)$
$\propto p(x|c)p(c)$
$= p(x_i, \ldots, x_n | c)p(c)$
$= p(x_i|c) \ldots p(x_n|c)p(c)$ (Due to feature independence assumption of naive bayes)

So, we simply need to use the parameters of the model, as calculated in part (1), and with help from the feature independence assumption of naive bayes, multiply the probabilities of a sample's features together to get its final class probability.

Then, assuming there are two classes, we can calculate $p(c = 0|x)$ and $p(c = 1|x)$ and assign the sample to the class with the higher probability.

3)

Since 'viagra' never appears in the spam training data, we will assign a probability of 0 to it. This means, given a test sample, $p(x_i = \text{viagra}|c) = 0$. Since we multiply that 0 probability with the probability of all the other features, the final probability of the sample belonging to either class will be 0. Obviously, we cannot assign a 0 probability of the sample belonging to both classes.

To counter this effect, we need to add smoothing through add-1 smoothing. This is a very common practice in natural language processing and slightly modifies the way the parameters of the model are computed. In essence, it assigns slightly higher probabilities to words/features with few or zero counts, and slightly reduces the probability of more common words/features in the samples. So, it smooths the distribution across all features and can be computed as follows:

$$P(x_i|c) = \frac{\text{count}(x_i|c)+1}{|V|+\sum_{x_w \in V} \text{count}(x_w|c)}$$

Please note, this above notation is a bit specific to the language domain. As such, w represents a word within the vocabulary, V represents the vocabulary of the samples, and $|V|$ represents the cardinality of the vocabulary.

One way a spammer might try to fool a naive bayes spam filter is to estimate the distribution of words in the training data. They can do this by registering an account with the email service provider and sending emails to themself. Then, since the email service provider automatically tags which emails are spam or not, the attacker/spammer can compute the distribution of words within each class. Once they know the distribution of words in spam/non-spam, they can create a normal spam email as they normally would, but then fill the email footer with tiny font with the appropriate count of non-spam word to match the training distribution. As such, the the spam filter will read across the entire email, including the email footer, see there are far more non-spam words than spam words (which all happen to be located in the footer), and classify the email as non-spam.

Problem 6

You can run my program by executing "q6.m". The CPTs found using EM are as follows:

P(fuse = false)	P(fuse = true)
0.7739	0.2261

P(drum = false)	P(drum = true)
0.6153	0.3847

P(toner = false)	P(toner = true)
0.6149	0.3851

P(paper = false)	P(paper = true)
0.3446	0.6554

P(roller = false)	P(roller = true)
0.7164	0.2836

	P(burning = false)	P(burning = true)
fuse = false	0.9999	0.0001
fuse = true	0.0010	0.9990

	P(quality = false)	P(quality = true)
drum = false, toner = false, paper = false	0.9995	0.0005
drum = false, toner = false, paper = true	0.9993	0.0007
drum = false, toner = true, paper = false	0.0029	0.9971
drum = false, toner = true, paper = true	0.0024	0.9976
drum = true, toner = false, paper = false	0.0017	0.9983
drum = true, toner = false, paper = true	0.0013	0.9987
drum = true, toner = true, paper = false	0.0010	0.9990
drum = true, toner = true, paper = true	0.0007	0.9993

	P(wrinkled = false)	P(wrinkled = true)
fuse = false, paper = false	0.9997	0.0003
fuse = false, paper = true	0.5873	0.4127
fuse = true, paper = false	0.5195	0.4805
fuse = true, paper = true	0.0012	0.9988

	P(mult.pages = false)	P(mult.pages = true)
paper = false, roller = false	0.9997	0.0003
paper = false, roller = true	0.0039	0.9961
paper = true, roller = false	0.6353	0.3647
paper = true, roller = true	0.4210	0.5790

	P(paper jam = false)	P(paper jam = true)
fuse = false, roller = false	0.4683	0.5317
fuse = false, roller = true	0.0005	0.9995
fuse = true, roller = false	0.2728	0.7272
fuse = true, roller = true	0.9969	0.0031

Using the computed CPTs, given no wrinkled pages, no burning smell, and poor print quality, the probability there is a drum unit problem is **0.6190**.