COMP9417 - Machine Learning
Tutorial: Classification

Weekly Problem Set: Please submit questions 2c, 3a and 3b on Moodle by 12pm Tuesday 8th March, 2022. Please only submit these requested questions and no others.

Question 1 (Bayes Rule)

Assume that the probability of a certain disease is 0.01. The probability of testing positive given that a person is infected with the disease is 0.95, and the probability of testing positive given that the person is not infected with the disease is 0.05.

(a) Calculate the probability of testing positive.

Solution:

Let T denote the event that the test returns a positive, and let D denote the event that an individual has the disease, and $\overline{D}$ the complement of D. Then we wish to find $P(T)$. It follows by the law of total probability that

$$\begin{aligned} P(T) &= P(T|D)P(D) + P(T|\overline{D})P(\overline{D}) \\ &= 0.95 \times 0.01 + 0.05 \times (1 - 0.01) \\ &= 0.059. \end{aligned}$$

In other words, the probability that a randomly chosen individual tests positive is 5.9%.

(b) Calculate the probability of being infected with the disease, given that the test is positive.

Solution:

We want to compute the probability: $P(D|T)$. From Bayes rule, we have

$$\begin{aligned} P(D|T) &= \frac{P(D \cap T)}{P(T)} \\ &= \frac{P(T|D)P(D)}{P(T)} \\ &= \frac{0.95 \times 0.01}{0.059} \\ &= 0.16, \end{aligned}$$

so the probability that someone has the disease given that the test is positive is only 16%. This result is somewhat surprising, but is due to the fact that the disease is so rare, that it pops up

only in 1% of the population. In other words, our prior information on the disease pushes the probability of having the disease even after testing positive to be quite low.

(c) Now assume that you test the individual a second time, and the test comes back positive (so two tests, two positives). Assume that conditional on having the disease, the outcomes of the two tests are independent, what is the probability that the individual has the disease? (note, conditional independence in this case means that $P(TT|D) = P(T|D)P(T|D)$, and not $P(TT) = P(T)P(T)$.) You may also assume that the test outcomes are conditionally independent given not having the disease.

Solution:
We now want to compute the probability $P(D|TT)$. First note that

$$\begin{aligned} P(TT) &= P(TT|D)P(D) + P(TT|\overline{D})P(\overline{D}) \\ &= P(T|D)^2P(D) + P(T|\overline{D})^2P(\overline{D}) \\ &= (0.95)^2 \times 0.01 + (0.05)^2 \times (1 - 0.01) \\ &= 0.0115. \end{aligned}$$

Then,

$$\begin{aligned} P(D|TT) &= \frac{P(TT|D)P(D)}{P(TT)} \\ &= \frac{(0.95)^2 \times 0.01}{0.0115} \\ &= 0.7848. \end{aligned}$$

Now we see that our data (the two positives) is starting to contribute more and more evidence, and so our prior belief about the disease contributes less to the probability.

Question 2 (Lecture Review)
In this question, we will review some important ideas from the lecture.

(a) What is probabalistic classification? How does it differ from non-probabalistic classification methods?

Solution:
In a classification problem, we are given a set of classes, $C = \{c_1, \ldots, c_K\}$, a set of examples $D = \{(\mathbf{x}_i, c_i) : i = 1, \ldots, n\}$, where $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^T$ is a vector of features for the i-th subject. The goal of classification is, given a new datum, $\mathbf{x}_\star$, to predict the 'best' class, $c_\star$. The definition of 'best' varies across methods, for example, under kNN (which is not a probabalistic classification method), 'best' meant the class in agreement with the k neighbours of $\mathbf{x}_\star$. Under the probabalistic approach 'best' is taken to mean 'most probable'. In order to talk about probabilities, we need to talk about probability distributions. In particular, we are most interested in the conditional probability distribution of a class given the data:

$$p(c_k|\mathbf{x}_\star).$$

We would like to compute this probability for each of the k classes, and return our prediction as

$$c_\star = \arg\max_k p(c_k|\mathbf{x}_\star).$$

There are a couple of approaches we could take to learning this conditional distribution. One approach, called the discriminative approach, is to learn the distribution $p(c_k|\mathbf{x})$ directly - this is what Logistic regression does for example. An alternative approach that we will focus on here would be the generative approach. To understand this second approach, recall Bayes rule:

$$p(c_k|\mathbf{x}) = \frac{p(\mathbf{x}|c_k)p(c_k)}{p(\mathbf{x})},$$

which tells us that instead of focusing on $P(c_k|\mathbf{x})$, we could instead model $p(\mathbf{x}|c_k)$ and $p(c_k)$, for each k - this is called a generative approach. $p(\mathbf{x}|c_k)$ is termed the class conditional distribution, and $p(c_k)$ is termed the class prior distribution. Note that we do not need to model $p(\mathbf{x})$ since it can be computed from knowledge of the class conditional and prior distributions, i.e.

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(\mathbf{x}|c_k)p(c_k).$$

Now, what we need to do under a generative approach is to choose the form of the class conditional distribution (e.g. continuous, multinomial, multivariate bernoulli, etc.) and class prior distribution (e.g. discrete uniform, categorical, etc). The prior is usually taken to be a discrete distribution since the number of classes is finite, whilst the class conditional distribution can be either continuous or discrete. One example of this approach is Gaussian (linear) discriminant analysis.

(b) What is the Naive Bayes assumption and why do we need it?

Solution:

In the previous section we saw that in order to do generative classification, we must learn a class conditional distribution for each of the k classes in our data. Now, if our observations live in a high-dimensional space (the number of features p is very large), then learning these distributions can be very difficult. For example, if $p = 1000$, and we assume that $x|c_k \sim N(\mu_k, \sigma_k^2)$, then we would have to learn a 1000-dimensional Gaussian for each class. Learning high dimensional distributions can be very tricky, and we often need a large amount of data (relative to p) to be able to do it well. Naive Bayes gives us a way out of this problem. Before

describing it, let's go back and rewrite the quantity we are trying to estimate:

$$\begin{aligned} p(\mathbf{x}|c_k)p(c_k) &= p(\mathbf{x}, c_k) \\ &= p(x_1, x_2, \ldots, x_p, c_k) \\ &= p(x_1|x_2, \ldots, x_p, c_k)p(x_2, x_3, \ldots, x_p, c_k) \\ &= p(x_1|x_2, \ldots, x_p, c_k)p(x_2|x_3, \ldots, x_p, c_k)p(x_3, x_4 \ldots, x_p, c_k) \\ &\vdots \\ &= p(x_1|x_2, \ldots, x_p, c_k)p(x_2|x_3, \ldots, x_p, c_k)p(x_3|x_4 \ldots, x_p, c_k) \times \cdots \times p(x_p|c_k)p(c_k). \end{aligned}$$

These computations are valid for any probability distribution, and we have so far made zero assumptions. Naive Bayes introduces the assumption that the features are conditionally independent, this is written as

$$x_i \perp x_j | c_k \quad \text{for all } j \neq i.$$

Now, let's interpret this assumption intuitively. Let's assume that we are trying to classify whether a patient has or doesn't have diabetes, and I tell you that a particular patient, $\mathbf{x}$, in our dataset belongs to the class diabetes. Let's also assume that x_1 describes the blood pressure of the patient, and x_2 describes the weight of the patient. Now, the naive bayes assumption tells us that if we know the patient is diabetic, then also knowing that the patient has high blood pressure tells us nothing about their weight. Realistically, we expect that for patients with diabetes, there exists a positive correlation between blood pressure and weight. Under Naive Bayes however, we ignore this correlation for the sake of mathematical tractability.
Mathematically, the Naive Bayes assumption means that $p(x_i|x_j, c_k) = p(x_i|c_k)$. Using this, we can rewrite:

$$\begin{aligned} p(\mathbf{x}|c_k)p(c_k) &= p(\mathbf{x}, c_k) \\ &= p(x_1|x_2, \ldots, x_p, c_k)p(x_2|x_3, \ldots, x_p, c_k)p(x_3|x_4 \ldots, x_p, c_k) \times \cdots \times p(x_p|c_k)p(c_k) \\ &= p(x_1|c_k)p(x_2|c_k)p(x_3|c_k) \times \cdots \times p(x_p|c_k)p(c_k) \\ &= p(c_k) \prod_{i=1}^{p} p(x_i|c_k). \end{aligned}$$

This means that instead of estimating a p-dimensional distribution, we now just have to estimate p different 1-dimensional distributions (for each of the k classes) instead - since estimating 1-dimensional distributions is much more straight forward, this is a much easier task!

(c) Consider the problem from lectures of classifying emails as spam or ham, with training data summarised below: Each row represents an email, and each email is a combination of words taken from the set $\{a, b, c, d, e\}$. We treat the words d, e as stop words - these are words that are not useful for classification purposes, for example, the word 'the' is too common to be useful for classifying documents as spam or ham. We therefore define our vocabulary as $V = \{a, b, c\}$. Note that in this case we have two classes, so $k = 2$, and we will assume a uniform prior, that is:

$$p(c_+) = p(c_-) = \frac{1}{2},$$

e_1	b	d	e	b	b	d	e			
e_2	b	c	e	b	b	d	d	e	c	c
e_3	a	d	a	d	e	a	e	e		
e_4	b	a	d	b	e	d	a	b		
e_5	a	b	a	b	a	b	a	e	d	
e_6	a	c	a	c	a	c	a	e	d	
e_7	e	a	e	d	a	e	a			
e_8	d	e	d	e	d					

where $c_+ =$ spam, $c_- =$ ham. Review the multivariate Bernoulli Naive Bayes set-up and classify the test example: assume we get a new email that we want to classify: $e_\star = abbdebb$

Solution:

Under the multivariate Bernoulli NB set-up, we encode the first email (e_1) as $\mathbf{x}_1 = (x_{1a} = 0, x_{1b} = 1, x_{1c} = 0)$, so that each of the features are binary and represent whether a word is present or not in a given email. Carrying this out for all emails in our dataset gives us:

	x_{ia}	x_{ib}	x_{ic}
e_1	0	1	0
e_2	0	1	1
e_3	1	0	0
e_4	1	1	0
e_5	1	1	0
e_6	1	0	1
e_7	1	0	0
e_8	0	0	0

So, under this model, we are choosing to ignore the frequency of words in the email and just consider whether a word appears or not in an email. This is equivalent to modelling the class conditional distribution as

$$p(\mathbf{x}|c_k) = \prod_{j \in V} p(x_j|c_k),$$

where $x_j|c_k \sim$ Bernoulli(p_j^k). In other words:

$$p(x_j|c_k) = (p_j^k)^{x_j}(1 - p_j^k)^{1-x_j}.$$

Note that p_j^k is the probability that a document of class k contains the word j or not. So, under this approach, we must estimate the following parameters:

$$p_a^+, p_b^+, p_c^+, p_a^-, p_b^-, p_c^-.$$

We will not go into detail here, but under the hood we are using maximum likelihood estimation to estimate these probabilities. It turns out that maximum likelihood gives us an intuitive estimate

$$p_j^k = \frac{\text{number of documents of class } k \text{ that contain the word } j}{\text{number of documents in class } k},$$

so, based on our data, this gives us:

$$p_a^+ = \frac{2}{4}, \quad p_b^+ = \frac{3}{4}, \quad p_c^+ = \frac{1}{4},$$
$$p_a^- = \frac{3}{4}, \quad p_b^- = \frac{1}{4}, \quad p_c^- = \frac{1}{4}.$$

Now, for $e_\star = abbdebb$, we have the encoding $\mathbf{x}_\star = (1, 1, 0)$, and we classify according to

$$c_\star = \arg \max_{k \in \{+,-\}} p(c_k) p(\mathbf{x}_\star | c_k).$$

First, for $k = +$, we have:

$$\begin{aligned}
p(c_+)p(\mathbf{x}_\star|c_+) &\overset{\text{(NB)}}{=} p(c_+) \prod_{j \in V} p(x_j = x_{\star j}|c_+) \\
&= p(c_+) \times p(x_a = x_{\star a}|c_+) \times p(x_b = x_{\star b}|c_+) \times p(x_c = x_{\star c}|c_+) \\
&= p(c_+) \times p(x_a = 1|c_+) \times p(x_b = 1|c_+) \times p(x_c = 0|c_+) \\
&= p(c_+) \times p_a^+ \times p_b^+ \times (1 - p_c^+) \\
&= \frac{1}{2} \times \frac{2}{4} \times \frac{3}{4} \times \left(1 - \frac{1}{4}\right) \\
&= \frac{9}{64},
\end{aligned}$$

where (NB) is to signify that we are making the Naive Bayes assumption. A similar computation gives us:

$$\begin{aligned}
p(c_-)p(\mathbf{x}_\star|c_-) &= p(c_-) \times p_a^- \times p_b^- \times (1 - p_c^-) \\
&= \frac{1}{2} \times \frac{3}{4} \times \frac{1}{4} \times \left(1 - \frac{1}{4}\right) \\
&= \frac{9}{128}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
c_\star &= \arg \max_{k \in \{+,-\}} p(c_k) p(\mathbf{x}_\star | c_k) \\
&= \arg\max \left\{ \frac{9}{64}, \frac{9}{128} \right\} \\
&= c_+.
\end{aligned}$$

(d) Next, review Smoothing for the multivariate Bernoulli case. Why do we need smoothing What happens to our previous classification under the smoothed multvariate Bernoulli model?

Solution:
In the previous section, we saw how to make predictions under the Naive Bayes Multivariate Bernoulli model. Note that for each prediction, we have to multiply various probabilities with each other. These probabilities are of the form p_j^k or $(1 - p_j^k)$. Since we estimate these probabilities based on occurences in the data, this can be problematic if we have not seen particular observations before. For example, assume that we never observe the word a in any spam emails, so that:

	x_{ia}	x_{ib}	x_{ic}
e_1	0	1	0
e_2	0	1	1
e_3	0	0	0
e_4	0	1	0
e_5	1	1	0
e_6	1	0	1
e_7	1	0	0
e_8	0	0	0

Now, our estimate of $p_a^+ = 0$. Therefore, any new data point, $x_\star$, which has $x_{\star a} = 1$, regardless of the values of the other features, gets $p(\mathbf{x}_\star|+) = 0$. Further note that if $p_a^+ = 1$, which means every spam email has the word a, we get similar behaviour, since any new email that does not have the word a gets probability of spam equal to zero, as $(1 - p_a^+) = (1 - 1) = 0$. This is not ideal behaviour, since we allow one feature to override the entire model, and in the case of document classification, the model is completely unable to deal with new words not before seen in our vocabulary. Therefore, the modification is often made to Naive Bayes in which we smooth our probability estimates. This amounts to altering our estimating equation slightly to

$$p_j^k = \frac{\text{number of documents of class } k \text{ that contain the word } j + 1}{\text{number of documents in class } k + \text{number of possible values of } X}.$$

Note that in the bernoulli model, the number of possible values of X is 2 since we can only have $X = 0$ or $X = 1$.
This modification ensures that $p(\mathbf{x}|c_k)$ is never equal to zero, even if the number of times a word j appears in a document of class k is actually zero. So, our smoothed estimates now would be:

$$p_a^+ = \frac{2+1}{4+2} = \frac{3}{6}, \quad p_b^+ = \frac{3+1}{4+2} = \frac{4}{6}, \quad p_c^+ = \frac{1+1}{4+2} = \frac{2}{6},$$
$$p_a^- = \frac{3+1}{4+2} = \frac{4}{6}, \quad p_b^- = \frac{1+1}{4+2} = \frac{2}{6}, \quad p_c^- = \frac{1+1}{4+2} = \frac{2}{6}.$$

Estimation is done identically as before, except with our new estimates.
Note that smoothing can be thought of in a different way. We want our probability estimates to never be zero, which is equivalent to observing every possible combination of words and classes. In other words, we need to observe documents in each class that contain each word in our vocabulary, as well as documents in each class that do not contain each word in our vocabulary. We can therefore introduce 'pseudo-documents', that ensure we have seen every combination of outcomes at least once. This amounts to adding $e_{p1}, e_{p2}, e_{p3}, e_{p4}$ as follows

	x_{ia}	x_{ib}	x_{ic}
e_1	0	1	0
e_2	0	1	1
e_3	0	0	0
e_4	0	1	0
e_{p1}	1	1	1
e_{p2}	0	0	0
e_5	1	1	0
e_6	1	0	1
e_7	1	0	0
e_8	0	0	0
e_{p1}	1	1	1
e_{p2}	0	0	0

If this was our dataset, then none of our probability estimates based on the original estimation procedure would be zero.

(e) Redo the previous analysis for the Multinomial Naive Bayes model without smoothing. Use the following test email: $e_\star = abbdebbcc$

Solution:
Consider a k-sided die, with the probability of each side being θ_i, so that $\sum_{i=1}^{k} \theta_i = 1$. Now, if we throw the die n times (the n throws are independent) and record the frequency of each face of the die as a vector $X = (X_1, X_2, \ldots, X_k)$, where $X_i =$ number of times face i came up, then we say that X follows a multinomial distribution with parameters $\theta_1, \ldots, \theta_k$ and n. We summarise this as:

$$X \sim \text{Multinomial}(n, \theta_1, \ldots, \theta_k), \quad \sum_{i=1}^{k} \theta_i = 1, \ \theta_i > 0 \forall i.$$

The probability mass function of the multinomial is:

$$\begin{aligned} P(X = (x_1, x_2, \ldots, x_k)) &= \frac{n!}{x_1! \times x_2! \times \cdots \times x_k!} \theta_1^{x_1} \times \cdots \times \theta_k^{x_k} \\ &= \frac{n!}{\prod_{i=1}^{k} x_i!} \prod_{i=1}^{k} \theta_i^{x_i} \end{aligned}$$

Now, under this model, we choose to encode emails as count vectors, where now the features represent the number of occurences of a word, rather than whether or not they were present.

	x_{ia}	x_{ib}	x_{ic}
e_1	0	3	0
e_2	0	3	3
e_3	3	0	0
e_4	2	3	0
e_5	4	3	0
e_6	4	0	3
e_7	3	0	0
e_8	0	0	0

We now model the class conditional distribution as a multinomial, which implicitly uses the Naive Bayes assumption since the multinomial distribution assumes independence across trials.

$$p(\mathbf{x}|c_k) = \frac{n!}{\prod_{j\in V} x_j!} \prod_{j\in V} p(x_j|c_k)^{x_j},$$

and we write $\theta_j^k = p(x_j|c_k)$ to denote the probability that a word in document of class k takes the value j. Note the subtle difference to the bernoulli model. Previously, the probabilities were to do with the probability that a particular word existed in a given class. The probabilities here are to do with how likely a randomly chosen word in the class is a particular word. Now, we need to estimate the following parameters for our model:

$$\theta_a^+, \theta_b^+, \theta_c^+, \theta_a^-, \theta_b^-, \theta_b^-.$$

We once more omit the details of how to derive the estimates of these parameters, and as one might expect, the estimates are:

$$\theta_j^k = \frac{\text{number of times word } j \text{ appears in class } k}{\text{number of words that appear in class } k}$$

so, based on our data, this gives us:

$$\begin{aligned} \theta_a^+ &= \frac{5}{17}, \quad \theta_b^+ = \frac{9}{17}, \quad \theta_c^+ = \frac{3}{17}, \\ \theta_a^- &= \frac{11}{17}, \quad \theta_b^- = \frac{3}{17}, \quad \theta_c^- = \frac{3}{17}. \end{aligned}$$

Now, given $\mathbf{x}_\star = (1, 4, 2)$, we can compute

$$\begin{aligned} p(c_+)p(x_\star|c_+) &= p(c_+)\frac{7!}{1!\times 4!\times 2!} \times \left(\theta_a^1 \times \theta_b^4 \times \theta_c^2\right) \\ &= \frac{1}{2} \times \frac{7!}{4!\times 2!} \times \left(\frac{5}{17} \times \left(\frac{9}{17}\right)^4 \times \left(\frac{3}{17}\right)^2\right) \end{aligned}$$

and similarly for $p(c_-)p(\mathbf{x}_\star|c_-)$.

(f) Repeat the analysis for the smoothed Multinomial Naive Bayes model.

Solution:
In our predictions here, just as in the multivariate bernoulli case, we must multiply various probabilities. Therefore, this model suffers the same potential problems when probabilities

take the value zero. The difference here is that this only occurs when the *frequency* of a word in a particular class is zero. To get around this, we apply smoothing once more, which involves adding 1 to each of the numerators of our estimates. Note that for each class k, this means adding 1 to the numerator $|V|$ times, and so we add $|V|$ to the denominator to account for the $|V|$ extra (pseudo-documents) that we are artifically adding. Our estimates therefore become

$$\theta_j^k = \frac{\text{number of times word } j \text{ appears in class } k + 1}{\text{number of words that appear in class } k + |V|}.$$

Now, our estimates for the spam problem become

$$\theta_a^+ = \frac{5+1}{17+3} = \frac{6}{20}, \quad \theta_b^+ = \frac{9+1}{17+3} = \frac{10}{20}, \quad \theta_c^+ = \frac{3+1}{17+3} = \frac{4}{20},$$
$$\theta_a^- = \frac{11+1}{17+3} = \frac{12}{20}, \quad \theta_b^- = \frac{3+1}{17+3} = \frac{4}{20}, \quad \theta_c^- = \frac{3+1}{17+3} = \frac{4}{20}.$$

Question 3. Binary Logistic Regression, two perspectives

Recall from previous weeks that we can view least squares regression as a purely optimisation based problem (minimising MSE), or as a statistical problem (using MLE). We now discuss two perspectives of the Binary Logistic Regression problem. In this problem, we are given a dataset $D = \{(x_i, y_i)\}_{i=1}^n$ where the x_i's represent the feature vectors, just as in linear regression, but the y_i's are now binary. The goal is to model our output as a probability that a particular data point belongs to one of two classes. We will denote this predicted probability by

$$P(y = 1|x) = p(x)$$

and we model it as

$$\hat{p}(x) = \sigma(\hat{w}^T x), \qquad \sigma(z) = \frac{1}{1 + e^{-z}},$$

where $\hat{w}$ is our estimated weight vector. We can then construct a classifier by assigning the class that has the largest probability, i.e.:

$$\hat{y} = \arg\max_{k=0,1} P(\hat{y} = k|x) = \begin{cases} 1 & \text{if} \quad \sigma(\hat{w}^T x) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

note: do not confuse the function $\sigma(z)$ with the parameter σ which typically denotes the standard deviation.

(a) What is the role of the logistic sigmoid function $\sigma()$ in the logistic regression formulation? Why are we not able to simply use linear regression here? (a plot of $\sigma(z)$ may be helpful here).

Solution:

The problem with just modelling $\hat{p}(x) = \hat{w}^T x$ is that $\hat{w}^T x$ is an unbounded quantity, which means that $\hat{w}^T x$ can be any number on the real line. This doesn't make sense since we wish to

model $\hat{p}$ as a probability, and probabilities belong to the interval $[0, 1]$. The role of the logistic sigmoid, also referred to as a *squashing function* is to squash the *raw score* $w^T x$ into the desired interval.

(b) We first consider the statistical view of logistic regression. Recall in the statistical view of linear regression, we assumed that $y|x \sim N(x^T\beta^*, \sigma^2)$. Here, we are working with binary valued random variables and so we assume that

$$y|x \sim \text{Bernoulli}(p^*), \qquad p^* = \sigma(x^T w^*)$$

where $p^* = \sigma(x^T w^*)$ is the true unknown probability of a response belonging to class 1, and we assume this is controlled by some true weight vector w^*. Write down the log-likelihood of the data D (as a function of w), and further, write down the MLE objective (but do not try to solve it).

Solution:
The assumption implies that

$$P(y = 1|x) = p^y(1 - p)^{1-y},$$

so we can write down the log-likelihood as

$$\begin{aligned}
\ln L(w) &= \ln P(y_1, \ldots, y_n|x_1, \ldots, x_n) \\
&= \ln \left\{ \prod_{i=1}^{n} P(y_i|x_i) \right\} \\
&= \sum_{i=1}^{n} \ln P(y_i|x_i) \\
&= \sum_{i=1}^{n} \ln(p_i^{y_i}(1 - p_i)^{1-y_i}) \\
&= \sum_{i=1}^{n} [y_i \ln p_i + (1 - y_i) \ln(1 - p_i)].
\end{aligned}$$

At this point, we recall that

$$p_i = \sigma(w^T x_i) = \frac{1}{1 + e^{-w^T x_i}},$$

and so

$$\begin{aligned}
\ln L(w) &= \sum_{i=1}^{n} [y_i \ln \sigma(w^T x_i) + (1 - y_i) \ln(1 - \sigma(w^T x_i))] \\
&= \sum_{i=1}^{n} \left[y_i \ln \left(\frac{\sigma(w^T x_i)}{1 - \sigma(w^T x_i)} \right) + \ln(1 - \sigma(w^T x_i)) \right].
\end{aligned}$$

The MLE optimisation is then

$$\hat{w}_{\text{MLE}} = \arg \max_{w \in \mathbb{R}^p} \ln L(w).$$

Note that we cannot solve this problem by hand as in the linear regression case (you should try this for yourself). This is an example in which we rely solely on numerical methods to find a solution.

(c) An alternative approach to the logistic regression problem is to view it purely from the optimisation perspective. This requires us to pick a loss function and solve for the corresponding minimizer. Write down the MSE objective for logistic regression and discuss whether you think this loss is appropriate.

Solution:
The MSE objective is simply

$$\arg\min_w \frac{1}{n}\|y - \sigma(Xw)\|_2^2$$

where $y = [y_1, \ldots, y_n]^T$ and $\sigma(Xw)$ is the element-wise application of σ to the vector Xw, or we could equivalently write:

$$\arg\min_w \frac{1}{n}\sum_{i=1}^{n}(y_i - \sigma(w^T x_i))^2.$$

This objective makes sense mathematically, but the squared error does not seem like a particularly good way to measure the distance between the resposne y_i and our prediction $\sigma(w^T x_i)$. The reason being that y_i is binary, whereas our prediction is real-valued. There are more technical reasons for not wanting to use the MSE, in particular the MSE is not convex in the parameter vector w for this problem, and this makes it harder to optimize, but this is beyond the scope of the course.

(d) (**optional**) Consider the following problem: you are given two discrete probability distributions, P and Q, and you are asked to quantify how far Q is from P. This is a very common task in statistics and information theory. The most common way to measure the discrepancy between the two is to compute the Kullback-Liebler (KL) divergence, also known as the relative entropy, which is defined by:

$$D_{\mathrm{KL}}(P\|Q) = \sum_{x\in\mathcal{X}} P(x)\ln\frac{P(x)}{Q(x)},$$

where we are summing over all of the possible values of the underlying random variable. A good way to think of this is that we have a true distribution P, an estimate Q, and we are trying to figure out how bad our estimate is. Write down the KL divergence between two bernoulli distributions $P = \text{Bernoulli}(p)$ and $Q = \text{Bernoulli}(q)$.

Solution:

A Bernoulli random variable can take on only two values (namely $x = 0, 1$) and so we have:

$$\begin{aligned} D_{\mathrm{KL}}(P\|Q) &= \sum_{x=0}^{1} P(x) \ln \frac{P(x)}{Q(x)} \\ &= P(X=0) \ln \frac{P(X=0)}{Q(X=0)} + P(X=1) \ln \frac{P(X=1)}{Q(X=1)} \\ &= (1-p) \ln \frac{1-p}{1-q} + p \ln \frac{p}{q}. \end{aligned}$$

(e) (**optional**) Continuing with the optimisation based view: In our set-up, one way to quantify the discrepancy between our prediction $\hat{p}_i$ and the true label y_i is to look at the KL divergence between the two bernoulli distributions $P_i = \text{Bernoulli}(y_i)$ and $Q_i = \text{Bernoulli}(\hat{p}_i)$. Use this to write down an appropriate minimization for the logistic regression problem.

Solution:
We can simply define the KL loss as the sum of the KL divergences:

$$\begin{aligned} \mathcal{L}_{\mathrm{KL}}(w) &= \sum_{i=1}^{n} D_{\mathrm{KL}}(P_i\|Q_i) \\ &= \sum_{i=1}^{n} D_{\mathrm{KL}}(\text{Bernoulli}(y_i)\|\text{Bernoulli}(\hat{p}_i)) \\ &= \sum_{i=1}^{n} (1-y_i) \ln \frac{1-y_i}{1-\hat{p}_i} + y_i \ln \frac{y_i}{\hat{p}_i}. \end{aligned}$$

Note that we can expand the individual terms as:

$$\begin{aligned} (1-y_i) \ln \frac{1-y_i}{1-\hat{p}_i} + y_i \ln \frac{y_i}{\hat{p}_i} &= (1-y_i)\ln(1-y_i) - (1-y_i)\ln(1-\hat{p}_i) + y_i \ln y_i - y_i \ln \hat{p}_i \\ &\overset{w}{\propto} -(1-y_i)\ln(1-\hat{p}_i) - y_i \ln \hat{p}_i. \end{aligned}$$

In the last step above, we have noted that only terms that involve $\hat{p}_i$ are functions of w, and so we can safely remove all other terms from consideration. This is because we are optimising with respect to w, and so only need to consider terms that involve w. Plugging this back in shows that

$$\begin{aligned} \mathcal{L}_{\mathrm{KL}}(w) &\overset{w}{\propto} -\sum_{i=1}^{n} [(1-y_i)\ln(1-\hat{p}_i) + y_i \ln \hat{p}_i] \\ &= -\sum_{i=1}^{n} \left[y_i \ln \frac{\hat{p}_i}{1-\hat{p}_i} + \ln(1-\hat{p}_i) \right] \\ &= -\sum_{i=1}^{n} \left[y_i \ln \left(\frac{\sigma(w^T x_i)}{1-\sigma(w^T x_i)} \right) + \ln(1-\sigma(w^T x_i)) \right]. \end{aligned}$$

(f) (**optional**) In logistic regression (and other binary classification problems), we commonly use the cross-entropy loss, defined by

$$\mathcal{L}_{XE}(a, b) = -a \ln b - (1 - a) \ln(1 - b).$$

Using your result from the previous part, discuss why the XE loss is a good choice, and draw a connection between the statistical and optimisation views of logistic regression.

Solution:
We can easily see that minimizing the cross entropy loss is equivalent to minimizing the KL loss from the previous question, so the XE loss is a valid objective function. For concreteness, we have

$$\mathcal{L}_{\text{KL}}(w) \overset{w}{\propto} \mathcal{L}_{\text{XE}}(w),$$

and so it follows immediately that

$$\arg\min_w \mathcal{L}_{\text{KL}}(w) = \arg\min_w \mathcal{L}_{\text{XE}}(w),$$

so the XE gives us the same solution as does minimizing the KL divergence. Now, using our result from the previous question and comparing it to our MLE derivation above, we see that

$$\begin{aligned}
\hat{w}_{\text{MLE}} &= \arg\max_{w \in \mathbb{R}^p} \ln L(w) \\
&= \arg\max_{w \in \mathbb{R}^p} \sum_{i=1}^{n} \left[y_i \ln \left(\frac{\sigma(w^T x_i)}{1 - \sigma(w^T x_i)} \right) + \ln(1 - \sigma(w^T x_i)) \right] \\
&= \arg\min_{w \in \mathbb{R}^p} -\sum_{i=1}^{n} \left[y_i \ln \left(\frac{\sigma(w^T x_i)}{1 - \sigma(w^T x_i)} \right) + \ln(1 - \sigma(w^T x_i)) \right] \\
&= \arg\min_{w \in \mathbb{R}^p} \mathcal{L}_{\text{KL}}(w) \\
&= \arg\min_{w \in \mathbb{R}^p} \mathcal{L}_{\text{XE}}(w).
\end{aligned}$$

So, what we see is that minimizing the XE loss gives us the same solution as would maximizing the log-likelihood. This is similar to what we saw in the linear regression case: using the MSE is equivalent to maximizing the likelihood under the Gaussian assumption. We can think of this as: 'MSE is to Gaussians as XE is to Bernoullis'