UNIVERSITY OF AMSTERDAM

University of Amsterdam

Homework Assignment 2

Machine Learning I

2024

Pedro M. P. Curvo
15713725

Contents

1 Naive Bayes Modeling of Climate

a)

Our dataset consists of N samples, each of which composed by $x \in \mathbb{R}^D$ features and a target variable $t \in \mathbb{R}^K$, where K is the set of possible classes. t is a one-hot encoded vector, where the k-th element is 1 if the sample belongs to class k and 0 otherwise. Thus, we can write the likelihood of the data, without assuming Naive Bayes, as:

$$
\begin{aligned}
p(\boldsymbol{T}, \boldsymbol{X}|\boldsymbol{M}, \boldsymbol{B}) &= \prod_{n=1}^{N} p(\boldsymbol{t}_n, \boldsymbol{x}_n|\boldsymbol{M}, \boldsymbol{B}) \quad \text{assuming i.i.d samples} \\
&= \prod_{n=1}^{N} p(x_n|t_n, \boldsymbol{M}, \boldsymbol{B}) p(t_n|\boldsymbol{M}, \boldsymbol{B}) \\
&= \prod_{n=1}^{N} \prod_{k=1}^{K} [p(x_n|t_{nk} = 1, \boldsymbol{M}, \boldsymbol{B}) p(t_{nk} = 1|\boldsymbol{M}, \boldsymbol{B})]^{t_{nk}} \\
&= \prod_{n=1}^{N} \prod_{k=1}^{K} [p(x_n|C_k, \boldsymbol{M}, \boldsymbol{B}) p(C_k|\boldsymbol{M}, \boldsymbol{B})]^{t_{nk}} \quad \text{, since} \quad p(C_k|\boldsymbol{M}, \boldsymbol{B}) = p(t_{nk} = 1|\boldsymbol{M}, \boldsymbol{B}) \\
&= \prod_{n=1}^{N} \prod_{k=1}^{K} [\pi_k p(x_n|C_k, \boldsymbol{M}, \boldsymbol{B})]^{t_{nk}} \quad \text{, since} \quad p(C_k|\boldsymbol{M}, \boldsymbol{B}) = \pi_k
\end{aligned}
$$

Now, assuming the Naive Bayes assumption, we have that the features are conditionally independent given the class, i.e. $p(x_n|C_k, \boldsymbol{M}, \boldsymbol{B}) = \prod_{d=1}^{D} p(x_{nd}|C_k, \boldsymbol{M}, \boldsymbol{B})$, which allows us to write the likelihood as:

$$
p(\boldsymbol{T}, \boldsymbol{X}|\boldsymbol{M}, \boldsymbol{B}) = \prod_{n=1}^{N} \prod_{k=1}^{K} [\pi_k \prod_{d=1}^{D} p(x_{nd}|C_k, \boldsymbol{M}, \boldsymbol{B})]^{t_{nk}}
$$

b)

Assuming that the features are modelled as Gaussian Distributions, then we can assume the following for each one of the assumptions:

Bayes Classifier:

Since the features in x are not independent, then their relation would be explained by a multivariate Gaussian distribution. This multivariate gaussian distribution, for each class, would have a mean $\boldsymbol{\mu}_k$ and a covariance matrix $\boldsymbol{\Sigma}_k$. $\boldsymbol{\mu}_k \in \mathbb{R}^D$ and $\boldsymbol{\Sigma}_k \in \mathbb{R}^{D \times D}$, however since $\boldsymbol{\Sigma}_k$ is a covariance matrix, it is symmetric. Thus, the total number of parameters per class would be $D + \frac{D(D+1)}{2}$, where the first term corresponds to the mean and the second term to the covariance matrix.

Thus, for the entire number of classes, the total number of parameters would be $K \times (D + \frac{D(D+1)}{2})$. Adding to this, we would need to consider the prior probabilities for each class, which would add $K - 1$ parameters, since the last class can be inferred from the others. Thus, the total number of parameters would be:

$K \times (D + \frac{D(D+1)}{2}) + K - 1$.

Naive Bayes Classifier:

Now, if we assume that the features are independent, and thus $p(x_n|C_k, \boldsymbol{M}, \boldsymbol{B}) = \prod_{d=1}^{D} p(x_{nd}|C_k, \boldsymbol{M}, \boldsymbol{B})$, which becomes $p(x_{nd}|C_k, \boldsymbol{M}, \boldsymbol{B}) = \mathcal{N}(x_{nd}|\mu_{kd}, \sigma_{kd}^2)$, assuming that the features are modelled as Gaussian distributions, which is the same as saying that the covariance matrix $\boldsymbol{\Sigma}_k$ is diagonal. Then, the total number of parameters per class would be $2D$, D parameters that correspond to the mean for each feature and D parameters that correspond to the variance for each feature.

Thus, for the entire number of classes, the total number of parameters would be $K \times 2D$.

For the prior probabilities, we would need $K - 1$ parameters, as in the previous case, since the last class can be inferred from the others.

This brings the total number of parameters to:

$K \times 2D + K - 1$.

Naive

The term "naive" in the Naive Bayes classifier comes from the assumption that the features are independent given the class, meaning that the features are not correlated with each other. This assumption is not always true, actually, in the real world, most experiments would have correlated features. Even in the example of the climate dataset, the regions might have a degree of correlation, if, for example, you assume that one region is in the south hemisphere and the other in the north hemisphere, and thus the seasons would be inverted, you could have a negative correlation between the features. Another example that can be used to illustrate this is, for example, the detection of spam. i.e., imagine that you have a features that are the presence of certain words in the email, for example, "win", "money", etc. You know that when they appear together, the email is more likely to be spam, thus the features are correlated.

Note: In the case of the temperature above, the features might be correlated, as I said in the example before, so basically we are using a multivariate Gaussian distribution with a diagonal covariance matrix, meaning that in the hyperspace of the features, the distribution aligns with the axis and cannot have a diagonal elipse shape. Considering the Bayes classifier, we would have a full covariance matrix, which would allow the distribution to have a diagonal elipse shape. In the case of the Gaussian, we could approximate the full covariance case with a multivariate combination of diagonal covariance matrices. Imagine you have a 2D space, and you want to approximate a diagonal elipse, you could use two 1D elipses, one for each axis, and combine them to form the diagonal elipse. This could be a average case in the number of parameters since you would multipy the number of parameters of the bayes classifier by the number of gaussians you would use. Which, for a certain number of gaussians, could be less than the number of parameters of the bayes classifier.

c)

Assuming the Naive Bayes assumption, we can write the likelihood as we showed in the first question:

$$p(\boldsymbol{T}, \boldsymbol{X}|\boldsymbol{M}, \boldsymbol{B}) = \prod_{n=1}^{N} \prod_{k=1}^{K} [\pi_k \prod_{d=1}^{D} p(x_{nd}|C_k, \boldsymbol{M}, \boldsymbol{B})]^{t_{nk}}$$

Assuming the features are modelled as Gaussian distributions, we can write the likelihood as:

$$p(\boldsymbol{T}, \boldsymbol{X}|\boldsymbol{M}, \boldsymbol{B}) = \prod_{n=1}^{N}\prod_{k=1}^{K}[\pi_k \prod_{d=1}^{D} \frac{\beta_{dk}^{\frac{1}{2}}2}{\sqrt{2\pi}} \exp\left(-\frac{\beta_{dk}}{2}(x_{nd} - \mu_{dk})^2\right)]^{t_{nk}}$$

Taking the logarithm of the likelihood, we have:

$$\log p(\boldsymbol{T}, \boldsymbol{X}|\boldsymbol{M}, \boldsymbol{B}) = \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\left[\log \pi_k + \sum_{d=1}^{D}\left[\frac{1}{2}\log \beta_{dk} - \frac{1}{2}\log 2\pi - \frac{\beta_{dk}}{2}(x_{nd} - \mu_{dk})^2\right]\right]$$

d)

Solving for the MLE estimator for μ_{dk}, we have:

$$\begin{aligned}
\frac{\partial \log p(\boldsymbol{T}, \boldsymbol{X}|\boldsymbol{M}, \boldsymbol{B})}{\partial \mu_{dk}} &= 0 \\
\sum_{n=1}^{N} t_{nk}\left[\beta_{dk}(x_{nd} - \mu_{dk})\right] &= 0 \\
\sum_{n=1}^{N} t_{nk}\beta_{dk}x_{nd} - \sum_{n=1}^{N} t_{nk}\beta_{dk}\mu_{dk} &= 0 \\
\sum_{n=1}^{N} t_{nk}\beta_{dk}x_{nd} &= \sum_{n=1}^{N} t_{nk}\beta_{dk}\mu_{dk} \\
\sum_{n=1}^{N} t_{nk}x_{nd} &= \sum_{n=1}^{N} t_{nk}\mu_{dk} \\
\mu_{dk} &= \frac{\sum_{n=1}^{N} t_{nk}x_{nd}}{\sum_{n=1}^{N} t_{nk}}
\end{aligned}$$

This, if we think that t_{nk} is a binary variable that takes the value 1 when $x_n \in C_k$, we can see that the MLE estimator for μ_{dk} is the average of samples that belong to class k.

e)

Specifying $p(C_1|x)$ for the general k classes of the Naive Bayes classifier, we have:

$$\begin{aligned}
p(C_1|x) &= \frac{p(x|C_1)p(C_1)}{p(x)} \\
&= \frac{p(x|C_1)p(C_1)}{\sum_{k=1}^{K} p(x|C_k)p(C_k)} \\
&= \frac{\pi_1 \prod_{d=1}^{D} p(x_d|C_1)}{\sum_{k=1}^{K} \pi_k \prod_{d=1}^{D} p(x_d|C_k)} \\
&= \frac{\pi_1 \prod_{d=1}^{D} p(x_d|C_1, \mu_{d1}, \beta_{d1})}{\sum_{k=1}^{K} \pi_k \prod_{d=1}^{D} p(x_d|C_k, \mu_{dk}, \beta_{dk})}
\end{aligned}$$

f)

Firstly, we need to bare in mind that, contrary to a logistic regression, which is a discriminative model and models the posterior probability $p(C_k|x)$ without the need to model the likelihood $p(x|C_k)$, caring only about the decision boundary, the Naive Bayes classifier models the joint probability $p(C_k, x)$. Summing this idea: the Naive Bayes classifier learns the joint probability which allows to then sample from it, while the logistic regression does not care about the distribution of the data.

Looking at our previous answer, we can see that the Naive Bayes Classifier models the joint probability $p(x, C_1) = p(x|C_1)p(C_1)$ by first generating a class target C_1 based on the information of the prior π_1 and then generating the features x based on the information of the likelihood $p(x|C_1)$. This generates a distribution for the joint probability $p(x, C_1)$, which we can use to sample from it. Basically, if we want to generate a new sample that belongs to class C_1, we would just need to sample from the likelihood $p(x|C_1)$.

g)

Writing $p(C_1|x)$ for the normally distributed model explicitly, we have:

$$\begin{aligned} p(C_1|x) &= \frac{\pi_1 \prod_{d=1}^{D} p(x_d|C_1, \mu_{d1}, \beta_{d1})}{\sum_{k=1}^{K} \pi_k \prod_{d=1}^{D} p(x_d|C_k, \mu_{dk}, \beta_{dk})} \\ &= \frac{\pi_1 \prod_{d=1}^{D} \frac{\beta_{d1}^{\frac{1}{2}}}{\sqrt{2\pi}} \exp\left(-\frac{\beta_{d1}}{2}(x_d - \mu_{d1})^2\right)}{\sum_{k=1}^{K} \pi_k \prod_{d=1}^{D} \frac{\beta_{dk}^{\frac{1}{2}}}{\sqrt{2\pi}} \exp\left(-\frac{\beta_{dk}}{2}(x_d - \mu_{dk})^2\right)} \end{aligned}$$

h)

A datapoint x is classified as C_1 if $p(C_1|x) > p(C_k|x)$, $\forall k \neq 1$.

i)

Considering the inequality $p(C_1|x) > p(C_k|x)$, $\forall k \neq 1$, we can write it as:

$$\begin{aligned} p(C_1|x) &> p(C_k|x) \\ \frac{\pi_1 \prod_{d=1}^{D} p(x_d|C_1, \mu_{d1}, \beta_{d1})}{\sum_{k=1}^{K} \pi_k \prod_{d=1}^{D} p(x_d|C_k, \mu_{dk}, \beta_{dk})} &> \frac{\pi_k \prod_{d=1}^{D} p(x_d|C_k, \mu_{dk}, \beta_{dk})}{\sum_{k=1}^{K} \pi_k \prod_{d=1}^{D} p(x_d|C_k, \mu_{dk}, \beta_{dk})} \\ \pi_1 \prod_{d=1}^{D} p(x_d|C_1, \mu_{d1}, \beta_{d1}) &> \pi_k \prod_{d=1}^{D} p(x_d|C_k, \mu_{dk}, \beta_{dk}) \quad \text{, since the denominator is the same for both sides a} \end{aligned}$$

To have quadratic inequality in the form $ax^2 + bx + c > 0$, we can take the logarithm of both sides, since if $f(x) > g(x)$, then $\log f(x) > \log g(x) \forall x > 0$:

$$\log \pi_1 + \sum_{d=1}^{D} \left[\frac{1}{2} \log \beta_{d1} - \frac{1}{2} \log 2\pi - \frac{\beta_{d1}}{2} (x_d - \mu_{d1})^2 \right] > \log \pi_k + \sum_{d=1}^{D} \left[\frac{1}{2} \log \beta_{dk} - \frac{1}{2} \log 2\pi - \frac{\beta_{dk}}{2} (x_d - \mu_{dk})^2 \right]$$

$$\log \pi_1 + \sum_{d=1}^{D} \left[\frac{1}{2} \log \beta_{d1} - \frac{1}{2} \log 2\pi - \frac{\beta_{d1}}{2} (x_d - \mu_{d1})^2 \right] - \log \pi_k - \sum_{d=1}^{D} \left[\frac{1}{2} \log \beta_{dk} - \frac{1}{2} \log 2\pi - \frac{\beta_{dk}}{2} (x_d - \mu_{dk})^2 \right] >$$

$$\log \frac{\pi_1}{\pi_k} + \sum_{d=1}^{D} \left[\frac{1}{2} \log \frac{\beta_{d1}}{\beta_{dk}} - \frac{\beta_{d1}}{2} (x_d - \mu_{d1})^2 + \frac{\beta_{dk}}{2} (x_d - \mu_{dk})^2 \right] > 0$$

$$2 \log \frac{\pi_1}{\pi_k} + \sum_{d=1}^{D} \left[\log \frac{\beta_{d1}}{\beta_{dk}} - \beta_{d1} (x_d - \mu_{d1})^2 + \beta_{dk} (x_d - \mu_{dk})^2 \right] > 0$$

$$\sum_{d=1}^{D} \left[-\beta_{d1} (x_d - \mu_{d1})^2 + \beta_{dk} (x_d - \mu_{dk})^2 \right] > -2 \log \frac{\pi_1}{\pi_k} - \sum_{d=1}^{D} \left[\log \frac{\beta_{d1}}{\beta_{dk}} \right]$$

$$\sum_{d=1}^{D} \left[-\beta_{d1} x_d^2 + 2\beta_{d1} x_d \mu_{d1} - \beta_{d1} \mu_{d1}^2 + \beta_{dk} x_d^2 - 2\beta_{dk} x_d \mu_{dk} + \beta_{dk} \mu_{dk}^2 \right] > -2 \log \frac{\pi_1}{\pi_k} - \sum_{d=1}^{D} \left[\log \frac{\beta_{d1}}{\beta_{dk}} \right]$$

$$\sum_{d=1}^{D} \left[x_d^2 (\beta_{dk} - \beta_{d1}) + 2 x_d (\beta_{d1} \mu_{d1} - \beta_{dk} \mu_{dk}) + \beta_{d1} \mu_{d1}^2 - \beta_{dk} \mu_{dk}^2 \right] > -2 \log \frac{\pi_1}{\pi_k} - \sum_{d=1}^{D} \left[\log \frac{\beta_{d1}}{\beta_{dk}} \right]$$

$$\sum_{d=1}^{D} \left[x_d^2 (\beta_{dk} - \beta_{d1}) + 2 x_d (\beta_{d1} \mu_{d1} - \beta_{dk} \mu_{dk}) \right] > -2 \log \frac{\pi_1}{\pi_k} - \sum_{d=1}^{D} \left[\log \frac{\beta_{d1}}{\beta_{dk}} - \beta_{d1} \mu_{d1}^2 + \beta_{dk} \mu_{dk}^2 \right]$$

Which considering the constant terms as c, we have:

$$\sum_{d=1}^{D} \left[x_d^2 (\beta_{dk} - \beta_{d1}) + 2 x_d (\beta_{d1} \mu_{d1} - \beta_{dk} \mu_{dk}) \right] > c$$

j)

Considering the inequality showed before, we should expect that since it is a quadratic inequality, the decision boundary would create a region that is not convex. If we eliminate the quadratic term, we would have a linear inequality, which would create a linear decision boundary, which is true if we consider the case where $\beta_{dk} = \beta_{dl}$ for $k \neq l$. However, I'm going to show this in the next paragraphs.

The decision boundary is the set of points where $p(C_1|x) = p(C_k|x)$, $\forall k \neq 1$.

Now, considering the inequality showed before we can say that the boundary for the region that a point x belongs to class C_1 is the set of points where the inequality becomes:

$$\sum_{d=1}^{D} \left[x_d^2 (\beta_{dk} - \beta_{d1}) + 2 x_d (\beta_{d1} \mu_{d1} - \beta_{dk} \mu_{dk}) \right] = c$$

Now, to evaluate the shape of the decision boundary, we can take the Hessian of the condition:

$$H = \begin{bmatrix} \frac{\partial^2}{\partial x_1^2} & \frac{\partial^2}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2}{\partial x_1 \partial x_D} \\ \frac{\partial^2}{\partial x_2 \partial x_1} & \frac{\partial^2}{\partial x_2^2} & \cdots & \frac{\partial^2}{\partial x_2 \partial x_D} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial x_D \partial x_1} & \frac{\partial^2}{\partial x_D \partial x_2} & \cdots & \frac{\partial^2}{\partial x_D^2} \end{bmatrix}$$

$$H = \begin{bmatrix} 2(\beta_{1k} - \beta_{1d}) & 0 & \cdots & 0 \\ 0 & 2(\beta_{2k} - \beta_{21}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 2(\beta_{Dk} - \beta_{D1}) \end{bmatrix}$$

Now, considering the eigenvalues of the Hessian:

- If the eigenvalues are all positive, then the decision region inside the boundary is a convex region.
- If the eigenvalues are all negative, then the decision region inside the boundary is a concave region.
- If the eigenvalues are mixed, then the decision region inside the boundary is a saddle region.
- If the eigenvalues are all zero, then the decision region inside the boundary is a linear region.

This lead to the conclusion, only for the part of the decision regions that separate the classes, that:

- if $\beta_{dk} > \beta_{d1} \forall d$, then the decision region for class 1 is convex and the decision region for class k is non concave.
- if $\beta_{dk} < \beta_{d1} \forall d$, then the decision region for class 1 is non convex and the decision region for class k is convex.
- if $\beta_{dk} = \beta_{d1} \forall d$, then the decision region for class 1 is linear and the decision region for class k is linear.

To conclude, we can then see that for a set of K desision regions, for the case where $\beta_{dk}! = \beta_{dl}$ for $k \neq l$, we can have at best 1 convex region and $K - 1$ concave regions, since if region L is convex than all the other regions are concave. Saying this, we can conclude that the generality of the problem is that the decision regions are not convex, considering the case where $\beta_{dk}! = \beta_{dl}$ for $k \neq l$.

If we consider the case where $\beta_{dk} = \beta_{dl}$ for $k \neq l$, then the decision regions are linear, and, hence, we would end with a set of linear decision regions which would be convex.

This also makes sense relating to the previous question that was meant to show the quadratic inequality that would define the decision boundary. If we consider the case where $\beta_{dk}! = \beta_{dl}$ for $k \neq l$, then the inequality would be a quadratic inequality. Whereas, if we consider the case where $\beta_{dk} = \beta_{dl}$ for $k \neq l$, then the inequality would be a linear inequality.

k)

Same as before.

2 Binary Classification

a)

i)

The dataset above **cannot** be classified using a logistic regression model, since the data is **not linearly separable**, meaning that there is no line that can separate the two classes.

Assuming we have points $A(1,1)$, $B(-1,-1)$, $C(1,-1)$ and $D(-1,1)$, we can see that the points A and B belong to class *Orange* and the points C and D belong to class *Blue*. If we now assume a line that separates the two classes, this boundary could be defined as $ax+by+c=0$, where a, b and c are the parameters of the line. We would know that for the points to be on the same side of the line, then:

$$(ax_1 + by_1 + c) \cdot (ax_2 + by_2 + c) > 0$$

This because if a point belongs to a class, let's say the class above the line, putting in the equation of the line would give a positive value, and the same for the other point. For the other class, the value would be negative. Hence, we know that the product of the two values should be positive, if the points belong to the same class. Saying this, considering points A and B, we would have:

$$(a + b + c) \cdot (-a - b + c) > 0$$

Now, considering points A and D, we would have:

$$(a + b + c) \cdot (a - b + c) > 0$$

Since, both of them are positive, we can combain them to have:

$$(a + b + c)^2 \cdot (-a - b + c) \cdot (a - b + c) > 0$$

Meaning, that $(-a-b+c)$ and $(a-b+c)$ should have the same sign, which gives us a contradiction. Meaning that the data is not linearly separable.

ii)

The data **can** be classified using a logistic regression model with **non-linear basis functions**, depending on the basis functions that are used.

For example, consider the function $\Phi \in \mathbb{R}^2 \rightarrow \mathbb{R}^3$ defined as $\Phi(x_1, x_2) = (x_1, x_2, x_1x_2)$. Now, we know that to belong to class *Orange*, the value of x_1x_2 should be negative, and to belong to class *Blue*, the value of x_1x_2 should be positive, meaning that in this higher space the data is linearly separable by

the plane $x_1 x_2 = 0$, which is the third dimension of the space. We could also $\Phi \in \mathbb{R}^2 \to \mathbb{R}^1$ defined as $\Phi(x_1, x_2) = (x_1 x_2)$, which would also make the data linearly separable, and others. Basically, the idea is to transform the data into a different space where it is linearly separable, and then use the logistic regression model to classify the data.

iii)

The data **can** be classified using a **Multilayer Perceptron with one hidden layer**. While a single Perceptron is not able to classify the data, since the data is not linearly separable as shown before, a Multilayer Perceptron with one hidden layer can classify the data, since it can learn non-linear decision boundaries. For example, let's look at the architecture of a MLP with one hidden layer with 2 neurons. Neuron 1 could activate for the inputs that belong to class *Orange* and neuron 2 could activate for the inputs that belong to class *Blue*. i.e., two neurons would create two linear decision boundaries, which would allow the MLP to classify the data.

b)

The data *cannot* be classified using a *Naive Bayes Classifier*. This happens because the Naive Bayes Classifier assumes that the features are independent given the class. However, in this case, if know the class and know the value of x_1, we can infer the value of x_2, and vice-versa, i.e., the features are not independent. For example, if we know that x belongs to class *Orange*, and we know the sign of x_1, we can infer the sign of x_2.

Given the subset of points $A(1,1)$, $B(-1,-1)$, $C(1,-1)$ and $D(-1,1)$, we can see that the points A and B belong to class *Blue* and the points C and D belong to class *Orange*. Now, assuming only this subset of points and knowing that Naive Bayes assumes that:

$$p(x_1, x_2 | C_k) = p(x_1 | C_k) p(x_2 | C_k)$$

Then, for class orange and assuming a frequentist approach on the probabilities, we have that:

$$p(x_1 = 1 | C_{\text{Orange}}) = \frac{1}{2} \quad \text{, since we have 1 sample with } x_1 = -1 \text{ and 1 sample with } x_1 = 1$$

$$p(x_2 = 1 | C_{\text{Orange}}) = \frac{1}{2} \quad \text{, since we have 1 sample with } x_2 = -1 \text{ and 1 sample with } x_2 = 1$$

$$p(x_1 = 1, x_2 = 1 | C_{\text{Orange}}) = p(x_1 = 1 | C_{\text{Orange}}) p(x_2 = 1 | C_{\text{Orange}}) = \frac{1}{4} \quad \text{, assuming the Naives Bayes assumption}$$

However, the probability of $p(x_1 = 1, x_2 = 1 | C_{\text{Orange}})$ is 0, since we do not have any sample that belongs to class *Orange*.

This shows that the assumption of the Naive Bayes Classifier falls, hence we cannot use it to classify the data.

c)

The dataset consists of a inner circle of points that belong to class *Orange* and an outer annulus of points that belong to class *Blue*. If we perform a change of basis to polar coordinates, keeping the dimension of the problem, we can see that the data is linearly separable, since the radius of the points

that belong to class *Orange* is smaller than the radius of the points that belong to class *Blue* and the angle of the points does not matter. Hence, we can say that knowing the angle of the point gives no information about tthe radius and vice-versa, which makes the data linearly separable in the polar coordinates.

Putting in on a mathematical form:

$$r = \sqrt{x^2 + y^2}$$
$$\theta = \arctan\left(\frac{y}{x}\right)$$

For class *Orange*, we have that:

$$r \in [0, r_1]$$
$$\theta \in [0, 2\pi]$$

For class *Blue*, we have that:

$$r \in [r_1, r_2]$$
$$\theta \in [0, 2\pi]$$

Assuming we have uniform distributions for the radius and the angle, we now have that:

$$p(r|C_{\text{Orange}}) = \begin{cases} \frac{1}{r_1} & \text{if } r \in [0, r_1] \\ 0 & \text{otherwise} \end{cases}$$
$$p(\theta|C_{\text{Orange}}) = \frac{1}{2\pi}$$
$$p(r|C_{\text{Blue}}) = \begin{cases} \frac{1}{r_2 - r_1} & \text{if } r \in [r_1, r_2] \\ 0 & \text{otherwise} \end{cases}$$
$$p(\theta|C_{\text{Blue}}) = \frac{1}{2\pi}$$

Now, in Naive Bayes we assume that the features are independent given the class, hence we have that:

$$p(r, \theta|C_{\text{Orange}}) = p(r|C_{\text{Orange}})p(\theta|C_{\text{Orange}})$$
$$p(r, \theta|C_{\text{Blue}}) = p(r|C_{\text{Blue}})p(\theta|C_{\text{Blue}})$$

Now, applying the Bayes rule, we have that:

$$p(C_{\text{Orange}}|r,\theta) = \frac{p(r,\theta|C_{\text{Orange}})p(C_{\text{Orange}})}{p(r,\theta)}$$
$$p(C_{\text{Blue}}|r,\theta) = \frac{p(r,\theta|C_{\text{Blue}})p(C_{\text{Blue}})}{p(r,\theta)}$$

Assuming Naive Bayes, we have that:

$$p(C_{\text{Orange}}|r,\theta) = \frac{p(r|C_{\text{Orange}})p(\theta|C_{\text{Orange}})p(C_{\text{Orange}})}{p(r,\theta)}$$
$$p(C_{\text{Blue}}|r,\theta) = \frac{p(r|C_{\text{Blue}})p(\theta|C_{\text{Blue}})p(C_{\text{Blue}})}{p(r,\theta)}$$

Now, substituting with the distributions, we have that:

$$p(C_{\text{Orange}}|r,\theta) = \begin{cases} \frac{\frac{1}{r_1}\frac{1}{2\pi}p(C_1)}{p(r,\theta)} & \text{if } r \in [0, r_1] \\ 0 & \text{otherwise} \end{cases}$$
$$p(C_{\text{Blue}}|r,\theta) = \begin{cases} \frac{\frac{1}{r_2-r_1}\frac{1}{2\pi}p(C_2)}{p(r,\theta)} & \text{if } r \in [r_1, r_2] \\ 0 & \text{otherwise} \end{cases}$$

Which simplifies to the fact that we only need to know the radius of the point to classify it and the angle does not give any information about the class of the point.

Different from the question (b), in this problem the features are independent given the class by the change of basis to polar coordinates, which makes the Naive Bayes Classifier applicable to the problem. In contrast to the previous question, where the features were not independent given the class and, even with a change of basis, the data was not linearly separable. We can think of this as the following: in the previous question, by knowing x_1 and the class we could infer x_2, which makes the features dependent. In this question, by knowing the radius we cannot infer nothing about the angle and vice-versa, which makes the features independent.

d)

In **Logistic Regression with Nonlinear Basis Functions**, $\phi(x)$ is a fixed and predetermined function that is applied to the input features, i.e., this transformation does not change during the training process and the weights are learned in the transformed space, meaning it only learns the weights associated with the transformed features.

In **Multilayer Perceptron with One Hidden Layer**, the ϕ transformation is learned during the training process, along its hidden layers. While the activation functions (such as ReLU or sigmoid) are predefined, the MLP adjusts the weights and biases within each layer which allows it to learn complex decision boundaries and not only the weights associated with the transformed features.

3 Regularized Logistic Regression

3.1 a)

Considering a logistic regression for K classes with N training vectores x_n mapped by a function $\phi(x_n)$ to M-dimensional space. And given that $p(C_k|\phi(x), w_1, \cdots, w_K) = y_k(\phi) = \frac{\exp(w_k^T \phi)}{\sum_{j=1}^{K} \exp(w_j^T \phi)}$. And given an assumption of a Gaussian prior on the vectors $w_1, \cdots, w_K$:

$$p(w_1, \cdots, w_k|\alpha) = \mathcal{N}(w_k|0, \alpha^{-1} I)$$

We can then write the likelihood $p(T|\Phi, w_1, \cdots, w_K)$ as:

$$\begin{aligned}
p(T|\Phi, w_1, \cdots, w_K) &= \prod_{n=1}^{N} p(t_n|\phi_n, w_1, \cdots, w_K) \\
&= \prod_{n=1}^{N} \prod_{k=1}^{K} p(t_{nk} = 1|\phi_n, w_1, \cdots, w_K)^{t_{nk}} \\
&= \prod_{n=1}^{N} \prod_{k=1}^{K} p(C_k|\phi_n, w_1, \cdots, w_K)^{t_{nk}} \\
&= \prod_{n=1}^{N} \prod_{k=1}^{K} y_k(\phi_n)^{t_{nk}} \\
&= \prod_{n=1}^{N} \prod_{k=1}^{K} \left(\frac{\exp(w_k^T \phi_n)}{\sum_{j=1}^{K} \exp(w_j^T \phi_n)} \right)^{t_{nk}}
\end{aligned}$$

Writing down the log likelihood, we have:

$$\begin{aligned}
\log p(T|\Phi, w_1, \cdots, w_K) &= \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log y_k(\phi_n) \\
&= \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log \left(\frac{\exp(w_k^T \phi_n)}{\sum_{j=1}^{K} \exp(w_j^T \phi_n)} \right) \\
&= \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \left(w_k^T \phi_n - \log \sum_{j=1}^{K} \exp(w_j^T \phi_n) \right)
\end{aligned}$$

3.2 b)

The prior in the question is given as:

$$
\begin{aligned}
p(w_1, \cdots, w_K|\alpha) &= \prod_{k=1}^{K} \mathcal{N}(w_k|0, \alpha^{-1}I) \\
&= \prod_{k=1}^{K} \left(\frac{\alpha}{2\pi}\right)^{\frac{M}{2}} \exp\left(-\frac{\alpha}{2} w_k^T w_k\right) \\
&= \left(\frac{\alpha}{2\pi}\right)^{\frac{M}{2}K} \exp\left(-\frac{\alpha}{2} \sum_{k=1}^{K} w_k^T w_k\right)
\end{aligned}
$$

Writing down the log prior, we have:

$$
\begin{aligned}
\log p(w_1, \cdots, w_K|\alpha) &= \sum_{k=1}^{K} \left(\frac{M}{2} \log \frac{\alpha}{2\pi} - \frac{\alpha}{2} w_k^T w_k\right) \\
&= \sum_{k=1}^{K} \left(\frac{M}{2} \log \frac{\alpha}{2\pi} - \frac{\alpha}{2} \sum_{m=0}^{M-1} w_{km}^2\right) \\
&= \frac{M}{2} K \log \frac{\alpha}{2\pi} - \frac{\alpha}{2} \sum_{k=1}^{K} \sum_{m=0}^{M-1} w_{km}^2
\end{aligned}
$$

Computing the logarithm helps us in two ways when we doing the computations. Firstly, it avoids numerical underflows specially when dealing with small numbers that can arise due to small probabilities, and secondly, it helps in the computations since the logarithm of a product is the sum of the logarithms of the terms, which makes the computations easier and, sometimes, reduce the problem to a multiplication of matrices.

3.3 c)

The posterior distribution is given by:

$$
\begin{aligned}
p(w_1, \cdots, w_K|T, \Phi, \alpha) &\propto p(T|\Phi, w_1, \cdots, w_K) p(w_1, \cdots, w_K|\alpha) \\
&= \prod_{n=1}^{N} \prod_{k=1}^{K} y_k(\phi_n)^{t_{nk}} \prod_{k=1}^{K} \left(\frac{\alpha}{2\pi}\right)^{\frac{M}{2}} \exp\left(-\frac{\alpha}{2} w_k^T w_k\right) \\
&= \prod_{n=1}^{N} \prod_{k=1}^{K} \left(\frac{\exp(w_k^T \phi_n)}{\sum_{j=1}^{K} \exp(w_j^T \phi_n)}\right)^{t_{nk}} \prod_{k=1}^{K} \left(\frac{\alpha}{2\pi}\right)^{\frac{M}{2}} \exp\left(-\frac{\alpha}{2} w_k^T w_k\right) \\
&= \prod_{n=1}^{N} \prod_{k=1}^{K} \left(\frac{\exp(w_k^T \phi_n)}{\sum_{j=1}^{K} \exp(w_j^T \phi_n)}\right)^{t_{nk}} \left(\frac{\alpha}{2\pi}\right)^{\frac{M}{2}K} \exp\left(-\frac{\alpha}{2} \sum_{k=1}^{K} w_k^T w_k\right) \\
&= \prod_{n=1}^{N} \prod_{k=1}^{K} \left(\frac{\exp(w_k^T \phi_n)}{\sum_{j=1}^{K} \exp(w_j^T \phi_n)}\right)^{t_{nk}} \left(\frac{\alpha}{2\pi}\right)^{\frac{M}{2}K} \exp\left(-\frac{\alpha}{2} \sum_{k=1}^{K} \sum_{m=0}^{M-1} w_{km}^2\right)
\end{aligned}
$$

3.4 d)

Taking the logarithm of the posterior distribution, we have:

$$\log p(w_1, \cdots, w_K | T, \Phi, \alpha) = \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log \left(\frac{\exp(w_k^T \phi_n)}{\sum_{j=1}^{K} \exp(w_j^T \phi_n)} \right) + \frac{M}{2} K \log \frac{\alpha}{2\pi} - \frac{\alpha}{2} \sum_{k=1}^{K} \sum_{m=0}^{M-1} w_{km}^2$$

To find the maximum of the posterior distribution, we would need to find:

$$\begin{aligned} &argmax_{w_1, \cdots, w_K} \log p(w_1, \cdots, w_K | T, \Phi, \alpha) \\ &= argmax_{w_1, \cdots, w_K} \left(\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log \left(\frac{\exp(w_k^T \phi_n)}{\sum_{j=1}^{K} \exp(w_j^T \phi_n)} \right) + \frac{M}{2} K \log \frac{\alpha}{2\pi} - \frac{\alpha}{2} \sum_{k=1}^{K} \sum_{m=0}^{M-1} w_{km}^2 \right) \end{aligned}$$

Now, we can drop the constant terms since they do not affect the maximum of the posterior distribution, and we have:

$$\begin{aligned} &argmax_{w_1, \cdots, w_K} \log p(w_1, \cdots, w_K | T, \Phi, \alpha) \\ &= argmax_{w_1, \cdots, w_K} \left(\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log \left(\frac{\exp(w_k^T \phi_n)}{\sum_{j=1}^{K} \exp(w_j^T \phi_n)} \right) - \frac{\alpha}{2} \sum_{k=1}^{K} \sum_{m=0}^{M-1} w_{km}^2 \right) \end{aligned}$$

Taking the argmax of the log posterior distribution is equivalent to taking the argmin of the negative log posterior distribution, hence we have:

$$\begin{aligned} &argmin_{w_1, \cdots, w_K} - \log p(w_1, \cdots, w_K | T, \Phi, \alpha) \\ &= argmin_{w_1, \cdots, w_K} \left(- \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log \left(\frac{\exp(w_k^T \phi_n)}{\sum_{j=1}^{K} \exp(w_j^T \phi_n)} \right) + \frac{\alpha}{2} \sum_{k=1}^{K} \sum_{m=0}^{M-1} w_{km}^2 \right) \end{aligned}$$

Which reduces the problem of minimizing the function:

$$E(w_1, \cdots, w_K) = - \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log \left(\frac{\exp(w_k^T \phi_n)}{\sum_{j=1}^{K} \exp(w_j^T \phi_n)} \right) + \frac{\alpha}{2} \sum_{k=1}^{K} \sum_{m=0}^{M-1} w_{km}^2$$

with respect to $w_1, \cdots, w_K$ as we wanted to show.

3.5 e)

If I'm certain that my weights should lie around zero, then I would choose a high value for α. This because if I'm certain about my weights and if they are modelled as a Gaussian distribution, then I want the most part of the distribution to be around zero, which means a smaller variance. Since the variance is inversely proportional to α, then I would choose a high value for α. This would make the weights to be close to zero, because since we are trying to minimize the function in the previous question, the term $\frac{\alpha}{2} \sum_{k=1}^{K} \sum_{m=0}^{M-1} w_{km}^2$ would also be minimized. The term is always positive, hence if α is high, then the weights would need to be close to zero to minimize the function and balance the

increase in the term $\frac{\alpha}{2}\sum_{k=1}^{K}\sum_{m=0}^{M-1} w_{km}^2$.

3.6 f)

If I'm very uncertain about my weights, then the distribution for the weights should have a high variance, resembling a uniform distribution. This would mean that α should be small, since the variance is inversely proportional to α. The more uncertain I'm the more α is small, so as $\alpha \to 0$, the equation at (d):

$$\lim_{\alpha\to 0} -\log p(w_1, \cdots, w_K|T, \Phi, \alpha) = \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \log\left(\frac{\exp(w_k^T\phi_n)}{\sum_{j=1}^{K}\exp(w_j^T\phi_n)}\right)$$

Which is actually the MLE solution.

3.7 g)

Taking the gradient of the log-likelihood with respect to w_j:

$$\begin{aligned}
\nabla_{w_j}\log p(T, \Phi, w_1, \cdots, w_K) &= \nabla_{w_j}\left(\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\log y_k(\phi_n)\right) \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\nabla_{w_j}\log y_k(w_k^T\phi_n) \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\frac{1}{y_k(w_k^T\phi_n)}\nabla_{w_j} y_k(w_k^T\phi_n) \quad \text{, since y is the softmax function} \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\frac{1}{y_k(w_k^T\phi_n)} y_k(w_k^T\phi_n)(1 - y_k(w_k^T\phi_n))\nabla_{w_j} w_k^T\phi_n \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}(1 - y_k(w_k^T\phi_n))\phi_n\delta_{jk} \\
&= \sum_{n=1}^{N}(t_{nj} - y_j(w_j^T\phi_n))\phi_n
\end{aligned}$$

3.8 h)

Now taking the gradient of the log-prior:

$$\begin{aligned}
\nabla_{w_j}\log p(w_1, \cdots, w_K|\alpha) &= \nabla_{w_j}\left(\frac{M}{2}K\log\frac{\alpha}{2\pi} - \frac{\alpha}{2}\sum_{k=1}^{K}\sum_{m=0}^{M-1} w_{km}^2\right) \\
&= -\alpha\sum_{m=0}^{M-1} w_{jm}
\end{aligned}$$

Now, taking the gradient of the log-posterior is just the sum of the gradients of the log-likelihood and the log-prior, since the derivative is a linear operator:

$$\begin{aligned}\nabla_{w_j} \log p(w_1, \cdots, w_K | T, \Phi, \alpha) &= \nabla_{w_j} \log p(T, \Phi, w_1, \cdots, w_K) + \nabla_{w_j} \log p(w_1, \cdots, w_K | \alpha) \\ &= \sum_{n=1}^{N} (t_{nj} - y_j(w_j^T \phi_n))\phi_n - \alpha \sum_{m=0}^{M-1} w_{jm}\end{aligned}$$