Homework 1

Luis Vitor Zerkowski - 14895730

October 4, 2023

1 Naive Bayes Modeling of Climate

1.a

Before starting, I make explicit the dimensions of each variable used for the question:

$$T \in \mathbb{R}^{N\times K}, \quad X \in \mathbb{R}^{N\times D}, \quad \Lambda \in \mathbb{R}^{D\times K}, \quad t_n \in \mathbb{R}^{K\times 1}, \quad x_n \in \mathbb{R}^{D\times 1}, \quad \lambda_k \in \mathbb{R}^{D\times 1},$$

$$t_{nk} \in \{0,1\}, \quad x_{nd} \in \mathbb{R}, \quad \lambda_{dk} \in \mathbb{R}$$

Without the Naive Bayes conditional independence assumption we cannot ignore correlations between features. But using the considering data i.i.d, we can still make use of the fact that the measurements are independent for each individual region. So the likelihood $p(T, X|\Lambda)$ is:

$$p(T, X|\Lambda) = \prod_{n=1}^{N} p(t_n, x_n|\Lambda) =$$

Using Bayes rule:

$$= \prod_{n=1}^{N} p(x_n|t_n, \Lambda)p(t_n|\Lambda) = \prod_{n=1}^{N}\prod_{k=1}^{K}(p(x_n|t_{nk}=1, \lambda_k)p(t_{nk}=1|\lambda_k))^{t_{nk}}$$

We also understand that the class C_k is encoded by $t_{nk} = 1$ and that it does not depend on the parameter vector λ_k, so $p(t_{nk} = 1|\lambda_k) = p(C_k|\lambda_k) = p(C_k) = \pi_k$. Using this expression, we get:

$$p(T, X|\Lambda) = \prod_{n=1}^{N}\prod_{k=1}^{K}(p(x_n|C_k, \lambda_k)\pi_k)^{t_{nk}}$$

Now going back to considering the Naive Bayes assumption, we can ignore the correlation between features and understand the class-conditional likelihood as a product of the univariate distributions for each feature. So continuing with the likelihood computations for the last equation, we have:

$$p(T, X|\Lambda) = \prod_{n=1}^{N}\prod_{k=1}^{K}(p(x_n|C_k, \lambda_k)\pi_k)^{t_{nk}}$$

By the Naive Bayes assumption:

$$p(T, X|\Lambda) = \prod_{n=1}^{N}\prod_{k=1}^{K}(\pi_k \prod_{d=1}^{D} p(x_{nd}|C_k, \lambda_{dk}))^{t_{nk}}$$

1.b

The number of parameters can indeed be affected by the Naive Bayes assumption depending on the class-conditioned distribution of the data. In the case of the exponential distribution we are working with, it's hard to compute the exact difference of parameters since this distribution does not have a direct multivariate analogous. But in general the idea is that by not using the Naive Bayes assumption, you need to consider the relationship between features, which increases the number of parameters a lot. In the Gaussian case, for example, it's the difference between using one variance term for each feature (Naive Bayes assumption and $\mathcal{O}(D)$) to using a covariance matrix for each feature indicating how the depend on one another (no Naive Bayes assumption and $\mathcal{O}(D^2)$).

The assumption is called naive because we are working as if we had no correlation between features, which is a strong assumption and commonly not the case. One good example is the task of classifying emails as normal or spam. For this case, our features are the presence or absence of a word in an email and the Naive Bayes assumption here is that the presence of a word is not correlated to the presence of another word. Even though the modeling leads to a good result in email classification, it is definitely inaccurate to assume that words are not correlated. Some words are more likely to appear in an email given that some other words are already there.

Two examples of this relationship are the word 'a' given the word 'lot' and the word 'you' given the word 'thank'. If an email has the word 'lot', it is very likely that it also has the word 'a' and if the email has the word 'thank' it is also very likely it contains 'you', so in reality the presence of these pairs are not uncorrelated.

1.c

Considering the likelihood we got from item **1.a**, we have:

$$\ln p(T, X|\Lambda) = \ln \prod_{n=1}^{N} \prod_{k=1}^{K} (\pi_k \prod_{d=1}^{D} p(x_{nd}|C_k, \lambda_{dk}))^{t_{nk}} =$$

$$= \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} (\ln \pi_k + \sum_{d=1}^{D} \ln p(x_{nd}|C_k, \lambda_{dk})) =$$

Now using the given likelihood model, we get:

$$= \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} (\ln \pi_k + \sum_{d=1}^{D} \ln(\lambda_{dk} e^{-\lambda_{dk} x_{nd}})) = \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} (\ln \pi_k + \sum_{d=1}^{D} (\ln(\lambda_{dk}) - \lambda_{dk} x_{nd}))$$

So the log-likelihood is given by $\ln p(T, X|\Lambda) = \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} (\ln \pi_k + \sum_{d=1}^{D} (\ln(\lambda_{dk}) - \lambda_{dk} x_{nd}))$.

1.d

To compute the MLE estimator for λ_{dk} we have to take the derivative of our log-likelihood expression an equal it to zero. So we have:

$$\frac{\partial}{\partial \lambda_{dk}} \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} (\ln \pi_k + \sum_{d=1}^{D} (\ln(\lambda_{dk}) - \lambda_{dk} x_{nd})) =$$

By the sum rule and considering t_{nk} is a constant:

$$= \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} (\frac{\partial}{\partial \lambda_{dk}} \ln \pi_k + \sum_{d=1}^{D} \frac{\partial}{\partial \lambda_{dk}} (\ln(\lambda_{dk}) - \lambda_{dk} x_{nd})) = \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \sum_{d=1}^{D} (\frac{\partial}{\partial \lambda_{dk}} \ln \lambda_{dk} - \frac{\partial}{\partial \lambda_{dk}} \lambda_{dk} x_{nd}) =$$

$$= \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \sum_{d=1}^{D} \delta_{dk} (\frac{1}{\lambda_{dk}} - x_{nd}) = \sum_{n=1}^{N} \frac{t_{nk} - t_{nk} \lambda_{dk} x_{nd}}{\lambda_{dk}} =$$

$$= \frac{1}{\lambda_{dk}} \sum_{n=1}^{N} (t_{nk} - t_{nk} \lambda_{dk} x_{nd})$$

Now equaling it to zero, we get:

$$\frac{1}{\lambda_{dk}} \sum_{n=1}^{N} (t_{nk} - t_{nk} \lambda_{dk} x_{nd}) = 0 \implies$$

$$\implies \sum_{n=1}^{N} t_{nk} = \sum_{n=1}^{N} t_{nk} \lambda_{dk} x_{nd} \implies$$

$$\implies \lambda_{dk} = \frac{\sum_{n=1}^{N} t_{nk}}{\sum_{n=1}^{N} t_{nk} x_{nd}}$$

Considering our computed MLE estimator, we can interpret the numerator $N_k = \sum_{n=1}^{N} t_{nk}$ as the number of targets belonging to class k and the denominator $N_{dk} = \sum_{n=1}^{N} t_{nk} x_{nd}$ as the sum of the temperatures at time d for class k. So the value that maximizes the likelihood for λ_{dk} is exactly the inverse of the mean temperature at time d for a class k.

1.e

$$p(C_1|\bar{x}) = \frac{p(\bar{x}|C_1, \lambda) p(C_1)}{p(\bar{x})} =$$

Given the Naive Bayes assumption and the given prior, we have:

$$= \frac{\pi_1 \prod_{d=1}^{D} p(\bar{x}_d|C_1, \lambda)}{p(\bar{x})} =$$

The evidence will be given by the summation of the class conditional likelihood times the prior for every class. Considering the the K classes, we have:

$$= \frac{\pi_1 \prod_{d=1}^{D} p(\bar{x}_d|C_1, \lambda)}{\sum_{k=1}^{K} p(\bar{x}|C_k, \lambda_k) p(C_k)} = \frac{\pi_1 \prod_{d=1}^{D} p(\bar{x}_d|C_1, \lambda)}{\sum_{k=1}^{K} \pi_k \prod_{d=1}^{D} p(\bar{x}_d|C_k, \lambda_k)}$$

So the posterior is given by $p(C_1|\bar{x}) = \frac{\pi_1 \prod_{d=1}^{D} p(\bar{x}_d|C_1, \lambda)}{\sum_{k=1}^{K} \pi_k \prod_{d=1}^{D} p(\bar{x}_d|C_k, \lambda_k)}$.

1.f

The Naive Bayes is considered a generative model because we can sample new data points for each class. In other words, since we were able to compute the joint distribution given by $p(C_k, \bar{x}) = p(\bar{x}|C_k) p(C_k)$, we can use a new feature vector $\bar{x}'$ and from the class-conditional likelihood distribution and the prior distribution assign it to a class. This result is shown in the previous exercise exactly because we are able to compute the joint distribution $p(C_1, \bar{x})$ in terms of λ and π.

1.g

Considering the obtained $p(C_1|\bar{x})$ from question **1.e**, we have:

$$p(C_1|\bar{x}) = \frac{\pi \prod_{d=1}^{D} p(\bar{x}_d|C_1, \lambda)}{\sum_{k=1}^{K} p(C_k) \prod_{d=1}^{D} p(\bar{x}_d|C_k, \lambda_k)} =$$

And now using the exponential model and the given prior distributions, we have:

$$= \frac{\pi_1 \prod_{d=1}^{D} \lambda_{d1} e^{-\lambda_{d1} \bar{x}_d}}{\sum_{k=1}^{K} \pi_k \prod_{d=1}^{D} \lambda_{dk} e^{-\lambda_{dk} \bar{x}_d}}$$

So the posterior is given by $p(C_1|\bar{x}) = \frac{\pi_1 \prod_{d=1}^{D} \lambda_{d1} e^{-\lambda_{d1} \bar{x}_d}}{\sum_{k=1}^{K} \pi_k \prod_{d=1}^{D} \lambda_{dk} e^{-\lambda_{dk} \bar{x}_d}}$.

1.h

The condition that defines that $\bar{x}$ is in class C_1 is simply that the probability of class C_1 given $\bar{x}$ is greater than the probability of any other class given $\bar{x}$. Mathematically, $p(C_1|\bar{x}) > p(C_k|\bar{x})$ for any $k \in \{2, \dots, K\}$.

1.i

Considering the expression found for $p(C_1|\bar{x})$ in exercise **1.g** and the inequality expressed in **1.h**, but changing k to j so we don't overload the notation, we have:

$$p(C_1|\bar{x}) > p(C_j|\bar{x}) \implies$$

$$\implies \frac{\pi_1 \prod_{d=1}^{D} \lambda_{d1} e^{-\lambda_{d1}\bar{x}_d}}{\sum_{k=1}^{K} \pi_k \prod_{d=1}^{D} \lambda_{dk} e^{-\lambda_{dk}\bar{x}_d}} > \frac{\pi_j \prod_{d=1}^{D} \lambda_{dj} e^{-\lambda_{dj}\bar{x}_d}}{\sum_{k=1}^{K} \pi_k \prod_{d=1}^{D} \lambda_{dk} e^{-\lambda_{dk}\bar{x}_d}} \implies$$

Noticing we have the same denominator, we get:

$$\implies \pi_1 \prod_{d=1}^{D} \lambda_{d1} e^{-\lambda_{d1}\bar{x}_d} > \pi_j \prod_{d=1}^{D} \lambda_{dj} e^{-\lambda_{dj}\bar{x}_d} \implies$$

Now to get to the specified form, we use the natural logarithm:

$$\implies \ln(\pi_1 \prod_{d=1}^{D} \lambda_{d1} e^{-\lambda_{d1}\bar{x}_d}) > \ln(\pi_j \prod_{d=1}^{D} \lambda_{dj} e^{-\lambda_{dj}\bar{x}_d}) \implies$$

$$\implies \ln \pi_1 + \sum_{d=1}^{D} (\ln \lambda_{d1} - \lambda_{d1}\bar{x}_d) > \ln \pi_j + \sum_{d=1}^{D} (\ln \lambda_{dj} - \lambda_{dj}\bar{x}_d) \implies$$

$$\implies \sum_{d_1}^{D} (\lambda_{dj} - \lambda_{d1})\bar{x}_d > \ln \pi_j - \ln \pi_1 + \sum_{d=1}^{D} (\ln \lambda_{dj} - \ln \lambda_{d1}) \implies$$

Using the specified form:

$$\implies \bar{x}^T \bar{a} > c$$

With $\bar{a} = (\lambda_j - \lambda_1) \in \mathbb{R}^{D \times 1}$ and $c = \ln \frac{\pi_j}{\pi_i} + \sum_{d=1}^{D} \ln \frac{\lambda_{dj}}{\lambda_{d1}}$.

1.j

I expect the decision regions to be convex. As shown in **1.i**, we have a boundary condition given by a linear equation $\bar{x}^T$, which means we are generating a linear boundary - a hyperplane on a D-dimensional space. Since for this problem we are not using any non-linear basis functions on the data points, just their direct features, we end up with linear decision boundaries for the regions. Considering the given convex definition, linear boundaries indeed yield convex decision regions.

1.k

Given two generic points $\bar{x}_1$, $\bar{x}_2$ in region C_1 and considering the results from **1.i**, we have:

$$\bar{x}_1^T \bar{a} > c \ \text{(I)} \ ; \ \bar{x}_2^T \bar{a} > c \ \text{(II)}$$

Now we can use the convex definition from **1.j** to get a generic point in the straight line defined by $\bar{x}_1$, $\bar{x}_2$:

$$\bar{x}_i = \bar{x}_1(1-\alpha) + \alpha \bar{x}_2 \ ; \ \text{with } \alpha \in (0,1)$$

So to prove that C_1 is convex we just need to show that any $\bar{x}_i$ also respects $\bar{x}_i^T \bar{a} > c$. We have:

$$\bar{x}_i^T \bar{a} = (\bar{x}_1(1-\alpha) + \alpha \bar{x}_2)^T \bar{a} = (1-\alpha)\bar{x}_1^T \bar{a} + \alpha \bar{x}_2^T \bar{a}$$

But using equations (I) and (II), we get:

$$(1-\alpha)\bar{x}_1^T \bar{a} + \alpha \bar{x}_2^T \bar{a} > (1-\alpha)c + \alpha c = c$$

So we just proved that the generic point $\bar{x}_i = \bar{x}_1(1-\alpha) + \alpha \bar{x}_2$; with $\alpha \in (0,1)$ indeed respects the condition $\bar{x}_i^T \bar{a} > c$, which proves the region C_1 is convex.

2 Binary Classification

2.a

2.a.1

The logistic regression method with linear functions cannot be used to classify the given dataset. This is because the dataset is clearly not linearly separable w.r.t. to the features x_1, x_2. The non-linearity introduced by the activation function is used primarily to map $\bar{w}^T\bar{x}$ to an interval $]0,1[$ so it has a probabilistic interpretation, but also to make the gradient of the error function easier to compute. In other words, even though we have a non-linear activation function, the data classification is not determined by this non-linearity, but rather by the hyperplane given by the parameter vector $\bar{w}$ on the feature space x_1, x_2.

2.a.2

On the other hand, logistic regression with non-linear basis function can be used to classify the given dataset. As said in **2.a.1**, the classification for this method is given by an hyperplane on the feature space. So if non-linearity is introduced to the features via a non-linear basis functions, we can project the initially non-linearly separable data onto a feature space in which the data is linearly separable. In this new feature space, the logistic regression method can be used to find a hyperplane that perfectly splits the data into their groups, thus correctly classifying every point.

2.a.3

The Multilayer Perceptron with one hidden layer can be used to classify the given dataset. This is because the hidden layers have adjustable linear parameters combined with a non-linear activation function, making the neurons basically trainable non-linear basis-functions for the input. With one hidden layer thus we can already project our data onto a feature space with non-linear basis functions and then, on the output layer, we use the projected data exactly like the logistic regression would do, computing a hyperplane that splits the data into their classes. Equation **5.9** on Bishop's book makes it easier to visualize: $y_k(\bar{x},\bar{w}) = \sigma(\sum_{j=0}^{M} w_{kj}^{(2)} h(\sum_{i=1}^{D} w_{ji}^{(1)} x_i))$, in which $w_{ji}^{(1)}$ are the trainable parameters for the basis function, $h(.)$ introduces its non-linearity and the rest is basically logistic regression with the computed non-linear basis functions.

2.a.4

For the linear regression with linear features case, similarly to logistic regression with linear features, it cannot be used to classify the dataset perfectly. On the other hand linear regression with non-linear basis function, similarly to logistic regression with non-linear basis functions, could be used to classify the dataset. Linear regression is really sensitive to outliers though, so if the data projected onto the feature space is not evenly spaced around a hyperplane, this method would probably fail classifying the dataset perfectly too.

As and addition to both **2.a.2** and **2.a.3**, it's worth noting that being able to correctly classify the dataset points doesn't mean it's an easy task. They would probably require a lot of effort to make the classification work properly, which means designing a proper basis-functions for the logistic regression case and using enough data points, enough neurons for the hidden layer and proper hyperparameters for the Multilayer Perceptron training.

2.b

The given dataset cannot be perfectly classified using Naive Bayes. This is because in this approach we ignore the correlations between features and compute the class-conditional likelihood $p(\bar{x}|C_k)$ as a product of univariate distributions, but our data is strongly not separable on each feature. This means we are considering the features x_1, x_2 separately and we end up with two features for which the classes overlap a lot and the overlap for classes on different features do not coincide much. Thus

if we fit distributions to both features, their multiplication cannot account for all the overlap, making it really difficult for us to perfectly split the data.

Mathematically, if you think about the posterior distribution $p(C_k|\bar{x})$ as proportional to the class-conditional likelihood $p(\bar{x}|C_k)$ times the prior $p(C_k)$, ignoring the normalization factor of the evidence, we won't be able to find a distribution that perfectly classifies the data simply by multiplying the class-conditional distributions for the features.

2.c

This dataset, on the other hand, can be classified using Naive Bayes. This is because now if we treat the features separately and fit distributions to them, the multiplication of these distributions can account for the class overlap on both the features since the overlaps coincide. To better understand it, let's say the inner points belong to class one. Since class one is concentrated around zero for both features, if we fit, for example, a gaussian to indicate class one on both features, their multiplication will lead to indicate class one when both features are around zero, and indicate class two when at least one of the features is not close to zero. This result exactly accounts for the overlap of classes on both features and indicates correct classification of the data.

2.d

The main difference is that in logistic regression case we use fixed pre designed non-linear basis functions and in the multilayer perceptron case we learn the non-linear basis functions by parameterizing them and iteratively updating the parameters' values.

3 Regularized Logistic Regression

3.a

We start by using the i.i.d property of the data to get:

$$p(T|\Phi, \bar{w}_1, \ldots, \bar{w}_K) = \prod_{n=1}^{N} p(\bar{t}_n|\bar{\phi}_n, \bar{w}_1, \ldots, \bar{w}_K) =$$

Now we use the one-hot-encoded vectors $\bar{t}_n$ as class selectors and then the fact that these vectors encode the actual class. So we have:

$$= \prod_{n=1}^{N}\prod_{k=1}^{K} p(t_{nk}|\bar{\phi}_n, \bar{w}_1, \ldots, \bar{w}_K)^{t_{nk}} = \prod_{n=1}^{N}\prod_{k=1}^{K} p(C_k|\bar{\phi}_n, \bar{w}_1, \ldots, \bar{w}_K)^{t_{nk}} =$$

Using the given posterior class probabilities, we get:

$$= \prod_{n=1}^{N}\prod_{k=1}^{K} y_k(\bar{\phi}_n)^{t_{nk}} = \prod_{n=1}^{N}\prod_{k=1}^{K} \left(\frac{e^{a_k}}{\sum_{j=1}^{K} e^{a_j}}\right)^{t_{nk}} = \prod_{n=1}^{N}\prod_{k=1}^{K} \left(\frac{e^{\bar{w}_k^T\bar{\phi}_n}}{\sum_{j=1}^{K} e^{\bar{w}_j^T\bar{\phi}_n}}\right)^{t_{nk}}$$

With the likelihood $p(T|\Phi, \bar{w}_1, \ldots, \bar{w}_K) = \prod_{n=1}^{N}\prod_{k=1}^{K} \left(\frac{e^{\bar{w}_k^T\bar{\phi}_n}}{\sum_{j=1}^{K} e^{\bar{w}_j^T\bar{\phi}_n}}\right)^{t_{nk}}$ in hands, we proceed to compute the log-likelihood:

$$\ln p(T|\Phi, \bar{w}_1, \ldots, \bar{w}_K) = \ln\left(\prod_{n=1}^{N}\prod_{k=1}^{K} \left(\frac{e^{\bar{w}_k^T\bar{\phi}_n}}{\sum_{j=1}^{K} e^{\bar{w}_j^T\bar{\phi}_n}}\right)^{t_{nk}}\right) =$$

$$= \sum_{n=1}^{N}\sum_{k=1}^{K} \ln\left(\frac{e^{\bar{w}_k^T\bar{\phi}_n}}{\sum_{j=1}^{K} e^{\bar{w}_j^T\bar{\phi}_n}}\right)^{t_{nk}} = \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\ln\left(\frac{e^{\bar{w}_k^T\bar{\phi}_n}}{\sum_{j=1}^{K} e^{\bar{w}_j^T\bar{\phi}_n}}\right) =$$

$$= \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\left(\ln e^{\bar{w}_k^T\bar{\phi}_n} - \ln\sum_{j=1}^{K} e^{\bar{w}_j^T\bar{\phi}_n}\right) = \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\left(\bar{w}_k^T\bar{\phi}_n - \ln\sum_{j=1}^{K} e^{\bar{w}_j^T\bar{\phi}_n}\right)$$

So the log-likelihood is given by $\ln p(T|\Phi, \bar{w}_1, \ldots, \bar{w}_K) = \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\left(\bar{w}_k^T\bar{\phi}_n - \ln\sum_{j=1}^{K} e^{\bar{w}_j^T\bar{\phi}_n}\right)$.

3.b

The prior is given by:

$$p(\bar{w}_1,\ldots,\bar{w}_K|\alpha)=\prod_{k=1}^{K}N(\bar{w}_k|\bar{0},\alpha^{-1}I)=\prod_{k=1}^{K}\frac{1}{\sqrt{(2\pi)^M|\alpha^{-1}I|}}e^{-\frac{1}{2}(\bar{w}_k-\bar{0})^T\alpha I(\bar{w}_k-\bar{0})}=$$

Considering the fact that the determinant of a diagonal matrix is given by the multiplication of its diagonal entries and removing the zero vector, we get:

$$=\prod_{k=1}^{K}\frac{1}{\sqrt{\left(\frac{2\pi}{\alpha}\right)^M}}e^{-\frac{1}{2}\bar{w}_k^T\alpha I\bar{w}_k}=\prod_{k=1}^{K}\frac{1}{\sqrt{\left(\frac{2\pi}{\alpha}\right)^M}}e^{-\frac{\alpha}{2}\bar{w}_k^T\bar{w}_k}$$

With the explicit form of the prior $p(\bar{w}_1,\ldots,\bar{w}_K|\alpha)=\prod_{k=1}^{K}\frac{1}{\sqrt{\left(\frac{2\pi}{\alpha}\right)^M}}e^{-\frac{\alpha}{2}\bar{w}_k^T\bar{w}_k}$ in hands, we compute the logarithm of the prior:

$$\ln p(\bar{w}_1,\ldots,\bar{w}_K|\alpha)=\ln\left(\prod_{k=1}^{K}\frac{1}{\sqrt{\left(\frac{2\pi}{\alpha}\right)^M}}e^{-\frac{\alpha}{2}\bar{w}_k^T\bar{w}_k}\right)=$$

$$=\sum_{k=1}^{K}\ln\left(\frac{e^{-\frac{\alpha}{2}\bar{w}_k^T\bar{w}_k}}{\sqrt{\left(\frac{2\pi}{\alpha}\right)^M}}\right)=\sum_{k=1}^{K}\left(-\frac{\alpha}{2}\bar{w}_k^T\bar{w}_k-\frac{M}{2}\ln\left(\frac{2\pi}{\alpha}\right)\right)$$

So the logarithm of the prior is given by $\ln p(\bar{w}_1,\ldots,\bar{w}_K|\alpha)=\sum_{k=1}^{K}\left(-\frac{\alpha}{2}\bar{w}_k^T\bar{w}_k-\frac{M}{2}\ln\left(\frac{2\pi}{\alpha}\right)\right)$.

The idea of computing this logarithm is directly related to computing the Maximum A Posteriori. Since it is essentially an optimization problem, we are going to take the derivative of the posterior expression at some point, which involves both the likelihood and the prior. When we finally compute the derivative of the posterior, it is much easier to work with the logarithm of the whole expression, which does not affect the optimization. Thus we are computing in advance two expressions, the log-likelihood and the log-prior, that are going to be useful for future computations.

3.c

Applying Bayes rule, the posterior is given by the likelihood times the prior over the evidence:

$$p(\bar{w}_1,\ldots,\bar{w}_K|\Phi,T,\alpha)=\frac{p(T|\Phi,\bar{w}_1,\ldots,\bar{w}_K)p(\bar{w}_1,\ldots,\bar{w}_K|\alpha)}{p(T|\Phi)}=$$

$$=\frac{\left(\prod_{n=1}^{N}\prod_{k=1}^{K}\left(\frac{e^{\bar{w}_k^T\bar{\phi}_n}}{\sum_{j=1}^{K}e^{\bar{w}_j^T\bar{\phi}_n}}\right)^{t_{nk}}\right)\left(\prod_{k=1}^{K}\frac{1}{\sqrt{\left(\frac{2\pi}{\alpha}\right)^M}}e^{-\frac{\alpha}{2}\bar{w}_k^T\bar{w}_k}\right)}{p(T|\Phi)}=$$

Now we just need to expand the evidence:

$$=\frac{\left(\prod_{n=1}^{N}\prod_{k=1}^{K}\left(\frac{e^{\bar{w}_k^T\bar{\phi}_n}}{\sum_{j=1}^{K}e^{\bar{w}_j^T\bar{\phi}_n}}\right)^{t_{nk}}\right)\left(\prod_{k=1}^{K}\frac{1}{\sqrt{\left(\frac{2\pi}{\alpha}\right)^M}}e^{-\frac{\alpha}{2}\bar{w}_k^T\bar{w}_k}\right)}{\int\cdots\int\left(\prod_{n=1}^{N}\prod_{k=1}^{K}\left(\frac{e^{\bar{w}_k^T\bar{\phi}_n}}{\sum_{j=1}^{K}e^{\bar{w}_j^T\bar{\phi}_n}}\right)^{t_{nk}}\right)\left(\prod_{k=1}^{K}\frac{1}{\sqrt{\left(\frac{2\pi}{\alpha}\right)^M}}e^{-\frac{\alpha}{2}\bar{w}_k^T\bar{w}_k}\right)\partial\bar{w}_1,\ldots,\partial\bar{w}_K}$$

3.d

Since we are interested in computing the MAP, we would take the derivatives of the posterior w.r.t to the parameters $\bar{w}_1, \ldots, \bar{w}_K$ and equal them to zero. In this scenario, we can ignore the evidence, since it's simply a normalization constant and we would multiply it by zero:

$$\frac{\partial}{\partial \bar{w}_k} \frac{\text{likelihood} * \text{prior}}{\text{evidence}} = 0 \implies \frac{1}{\text{evidence}} \frac{\partial}{\partial \bar{w}_k} \text{likelihood} * \text{prior} = 0 \implies \frac{\partial}{\partial \bar{w}_k} \text{likelihood} * \text{prior} = 0$$

We then start working on the MAP computations by taking the natural logarithm of the likelihood times the prior, so we have:

$$\ln(p(T|\Phi, \bar{w}_1, \ldots, \bar{w}_K) p(\bar{w}_1, \ldots, \bar{w}_K|\alpha)) =$$

$$= \ln\left(\left(\prod_{n=1}^{N}\prod_{k=1}^{K}\left(\frac{e^{\bar{w}_k^T \bar{\phi}_n}}{\sum_{j=1}^{K} e^{\bar{w}_j^T \bar{\phi}_n}}\right)^{t_{nk}}\right)\left(\prod_{k=1}^{K}\frac{1}{\sqrt{\left(\frac{2\pi}{\alpha}\right)^M}} e^{-\frac{\alpha}{2}\bar{w}_k^T \bar{w}_k}\right)\right) =$$

$$= \ln\left(\prod_{n=1}^{N}\prod_{k=1}^{K}\left(\frac{e^{\bar{w}_k^T \bar{\phi}_n}}{\sum_{j=1}^{K} e^{\bar{w}_j^T \bar{\phi}_n}}\right)^{t_{nk}}\right) + \ln\left(\prod_{k=1}^{K}\frac{1}{\sqrt{\left(\frac{2\pi}{\alpha}\right)^M}} e^{-\frac{\alpha}{2}\bar{w}_k^T \bar{w}_k}\right) =$$

Using the expression we already computed on questions **3.a** and **3.b**:

$$= \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\left(\bar{w}_k^T \bar{\phi}_n - \ln\sum_{j=1}^{K} e^{\bar{w}_j^T \bar{\phi}_n}\right) + \sum_{k=1}^{K}\left(-\frac{\alpha}{2}\bar{w}_k^T \bar{w}_k - \frac{M}{2}\ln\left(\frac{2\pi}{\alpha}\right)\right)$$

In this expression, there are two important details to notice. The first one is that there's a summation that does not depend on the parameter vectors, $\sum_{k=1}^{K} -\frac{M}{2}\ln\left(\frac{2\pi}{\alpha}\right)$. Since it would simply go to zero when we take the derivative w.r.t the parameter vectors, we can simply disregard it from our computations. The second point is related to an expression reduction. Given the definition of $y_k(\bar{\phi})$ provided, we compute its logarithm:

$$\ln y_k(\bar{\phi}) = \ln\left(\frac{e^{\bar{w}_k^T \bar{\phi}}}{\sum_{j=1}^{K} e^{\bar{w}_j^T \bar{\phi}}}\right) = \ln e^{\bar{w}_k^T \bar{\phi}} - \ln\sum_{j=1}^{K} e^{\bar{w}_j^T \bar{\phi}} = \bar{w}_k^T \bar{\phi} - \ln\sum_{j=1}^{K} e^{\bar{w}_j^T \bar{\phi}}$$

So we can reduce our equation to:

$$\ln(p(T|\Phi, \bar{w}_1, \ldots, \bar{w}_K) p(\bar{w}_1, \ldots, \bar{w}_K|\alpha)) = \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\ln y_k(\bar{\phi}_n) - \sum_{k=1}^{K}\frac{\alpha}{2}\bar{w}_k^T \bar{w}_k =$$

Expanding the vector multiplication $\bar{w}_k^T \bar{w}_k$ to a sum and moving the constant $\frac{\alpha}{2}$ that does not depend on the summation over k:

$$= \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\ln y_k(\bar{\phi}_n) - \frac{\alpha}{2}\sum_{k=1}^{K}\sum_{m=0}^{M-1} |w_{km}|^2$$

Finally, if we want to compute the MAP, instead of computing the parameters that maximize the above equation, we can choose to compute the parameters that minimize the negative of the equation, leading to:

$$-\ln(p(T|\Phi, \bar{w}_1, \ldots, \bar{w}_K) p(\bar{w}_1, \ldots, \bar{w}_K|\alpha)) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\ln y_k(\bar{\phi}_n) + \frac{\alpha}{2}\sum_{k=1}^{K}\sum_{m=0}^{M-1} |w_{km}|^2$$

So we just proved that obtaining the MAP estimate for the parameter vectors is indeed equivalent to minimize $-\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\ln y_k(\bar{\phi}_n) + \frac{\alpha}{2}\sum_{k=1}^{K}\sum_{m=0}^{M-1} |w_{km}|^2$.

3.e

We regularize our solution for multiple reasons, but here we talk about two. The firs one is overfitting, which happens when we fit our model too perfectly to our training dataset, specializing it on all the examples it has seen, but negatively affecting its performance on unseen data. For this phenomenon to happen, it is pretty common that the weights have super specific high values that make the the hyperplane on the feature space work just perfectly on the training data. Thus penalizing the model for the sum of the squared absolute values of the parameters helps by keeping the parameter values small, since the model pays a cost for increasing the parameters values.

The second one is also related to overfitting, but with a more specific situation. In the process of overfitting, sometimes the model finds the best way of categorizing the data by overusing some weights and almost ignoring others, kind of a parameter cannibalism. By making the model pay quadratically for increasing parameters values and therefore making the values low, we also guarantee that the hyperplane computation will most probably involve more parameters, since it has less freedom to specialize in only one weight. Given a design of basis functions, using more parameters normally leads to a model that better generalizes.

3.f

Without an informative prior, the MAP problem loses the regularization term and is basically reduced to a MLE problem.

3.g

For computing the gradient $\nabla_{\bar{w}_r} \ln p(T|\Phi, \bar{w}_1, \ldots, \bar{w}_K)$, we start by computing the derivative w.r.t a generic w_{rs}. So we have:

$$\frac{\partial}{\partial w_{rs}} \ln p(T|\Phi, \bar{w}_1, \ldots, \bar{w}_K) = \frac{\partial}{\partial w_{rs}} \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \left(\bar{w}_k^T \bar{\phi}_n - \ln \sum_{j=1}^{K} e^{\bar{w}_j^T \bar{\phi}_n} \right) =$$

By the sum rule:

$$= \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \left(\frac{\partial}{\partial w_{rs}} \bar{w}_k^T \bar{\phi}_n - \frac{\partial}{\partial w_{rs}} \ln \sum_{j=1}^{K} e^{\bar{w}_j^T \bar{\phi}_n} \right) =$$

$$= \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \left(\sum_{m=0}^{M-1} \frac{\partial}{\partial w_{rs}} w_{km}^T \phi_{nm} - \frac{1}{\sum_{j=1}^{K} e^{\bar{w}_j^T \bar{\phi}_n}} \frac{\partial}{\partial w_{rs}} \sum_{j=1}^{K} e^{\bar{w}_j^T \bar{\phi}_n} \right) =$$

$$= \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \left(\sum_{m=0}^{M-1} \delta_{kr} \delta_{ms} \phi_{nm} - \frac{1}{\sum_{j=1}^{K} e^{\bar{w}_j^T \bar{\phi}_n}} \sum_{j=1}^{K} \frac{\partial}{\partial w_{rs}} e^{\bar{w}_j^T \bar{\phi}_n} \right) =$$

$$= \sum_{n=1}^{N} t_{nr} \phi_{ns} - \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \left(\frac{1}{\sum_{j=1}^{K} e^{\bar{w}_j^T \bar{\phi}_n}} \sum_{j=1}^{K} e^{\bar{w}_j^T \bar{\phi}_n} \sum_{m=0}^{M-1} \frac{\partial}{\partial w_{rs}} w_{jm} \phi_{nm} \right) =$$

$$= \sum_{n=1}^{N} t_{nr} \phi_{ns} - \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \left(\frac{1}{\sum_{j=1}^{K} e^{\bar{w}_j^T \bar{\phi}_n}} \sum_{j=1}^{K} e^{\bar{w}_j^T \bar{\phi}_n} \sum_{m=0}^{M-1} \delta_{jr} \delta_{ms} \phi_{nm} \right) =$$

$$= \sum_{n=1}^{N} t_{nr} \phi_{ns} - \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \left(\frac{e^{\bar{w}_r^T \bar{\phi}_n} \phi_{ns}}{\sum_{j=1}^{K} e^{\bar{w}_j^T \bar{\phi}_n}} \right) = \sum_{n=1}^{N} t_{nr} \phi_{ns} - \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} y_r(\bar{\phi}_n) \phi_{ns} =$$

$$= \sum_{n=1}^{N} t_{nr} \phi_{ns} - \sum_{n=1}^{N} y_r(\bar{\phi}_n) \phi_{ns} \sum_{k=1}^{K} t_{nk} = \sum_{n=1}^{N} (t_{nr} - y_r(\bar{\phi}_n)) \phi_{ns}$$

Once the derivative w.r.t a generic w_{rs} is computed, we can simply describe the gradient as $\nabla_{\bar{w}_r} \ln p(T|\Phi, \bar{w}_1, \ldots, \bar{w}_K) = \sum_{n=1}^{N} (t_{nr} - y_r(\bar{\phi}_n)) \bar{\phi}_n^T$.

3.h

Similarly to what we did for the previous question, for computing the gradient $\nabla_{\bar{w}_r} \ln p(\bar{w}_1, \ldots, \bar{w}_K|\alpha)$, we start by computing the derivative w.r.t a generic w_{rs}. So we have:

$$\frac{\partial}{\partial w_{rs}} \sum_{k=1}^{K} \left(-\frac{\alpha}{2} \bar{w}_k^T \bar{w}_k - \frac{M}{2} \ln \left(\frac{2\pi}{\alpha} \right) \right) =$$

Zeroing the term that doesn't depend on $\bar{w}$ and then using the sum rule, we get:

$$= \frac{\partial}{\partial w_{rs}} \sum_{k=1}^{K} -\frac{\alpha}{2} \bar{w}_k^T \bar{w}_k = -\frac{\alpha}{2} \sum_{k=1}^{K} \frac{\partial}{\partial w_{rs}} \bar{w}_k^T \bar{w}_k = -\frac{\alpha}{2} \sum_{k=1}^{K} \sum_{m=0}^{M-1} \frac{\partial}{\partial w_{rs}} w_{km} w_{km} =$$

Now by the product rule:

$$= -\frac{\alpha}{2} \sum_{k=1}^{K} \sum_{m=0}^{M-1} \left(\left(\frac{\partial}{\partial w_{rs}} w_{km} \right) w_{km} + w_{km} \left(\frac{\partial}{\partial w_{rs}} w_{km} \right) \right) = -\frac{\alpha}{2} \sum_{k=1}^{K} \sum_{m=0}^{M-1} (\delta_{kr}\delta_{ms} w_{km} + w_{km}\delta_{kr}\delta_{ms}) =$$

$$= -\frac{\alpha}{2}(w_{rs} + w_{rs}) = -\alpha w_{rs}$$

Once the derivative w.r.t a generic w_{rs} is computed, we can simply describe the gradient as $\nabla_{\bar{w}_r} \ln p(\bar{w}_1, \ldots, \bar{w}_K|\alpha) = -\alpha \bar{w}_r^T$.

For the gradient of the log-posterior, we'll use the Bayes rule to argue it is given by the log-likelihood plus the log-prior minus the log-evidence. Thus by the sum rule, we can compute the derivative w.r.t a generic w_{rs} for the log-posterior simply as the sum of the derivatives of the log-likelihood, the log-posterior and the log-evidence. So we have:

$$\nabla_{\bar{w}_r} p(\bar{w}_1, \ldots, \bar{w}_K|T, \Phi, \alpha) = \nabla_{\bar{w}_r} \ln p(T|\Phi, \bar{w}_1, \ldots, \bar{w}_K) + \nabla_{\bar{w}_r} \ln p(\bar{w}_1, \ldots, \bar{w}_K|\alpha) - \nabla_{\bar{w}_r} p(T|\Phi) =$$

But since the log-evidence is just a constant, it goes zero when computing the derivative w.r.t $\bar{w}_r$. So we get:

$$= \sum_{n=1}^{N} (t_{nr} - y_r(\bar{\phi}_n)) \bar{\phi}_n^T - \alpha \bar{w}_r^T$$

Which indeed makes sense if you consider as a sanity check $\nabla_{\bar{w}_r} \ln p(T|\Phi, \bar{w}_1, \ldots, \bar{w}_K) \in \mathbb{R}^{1\times M}$ and $\nabla_{\bar{w}_r} \ln p(\bar{w}_1, \ldots, \bar{w}_K|\alpha) \in \mathbb{R}^{1\times M}$.