# CS 5824/ECE 5424: Advanced Machine Learning
# Final Exam, Fall 2022

**Ankit Parekh**
MS (Non-Thesis) Computer Engineering
ECE Department
ankitparekh@vt.edu

## Section A. Multiple Choice

**A1) (B) Higher.**

The set of data points follow a 4th order process. Therefore, we may observe that the Model 1 (1st order polynomial) underfits on the data while the Model 2 (9th order polynomial) overfits on the same data. This happens because the Model 1 is too simple to grasp the complexity present in the 4th order data, while the Model 2 is too complex and may pick up the noise variable which is present in the data along with the actual variables, overfitting to the training data. Regarding bias-variance, the Model 2 will have higher variance (too sensitive to noise present in the training data), while the Model 1 will have higher bias (unable to identify the key patterns in the training data because it is too simple).

**A2) (C) 3071, 32.**

All features are dependent on each other for a Bayes Classifier because we do not have the conditional independence assumption leading to a large number of parameters. The likelihood probabilities and prior probabilities add up to the total number of independent parameters.

In case of 3 classes, if we have prior probabilities of 2 classes, the prior for the 3rd class can be calculated (as their sum is 1). For 10 binary features and 3 classes, we need $3 * (2^{10} - 1) = 3069$ features (subtracting 1 because we can find the last probability using the previous all probabilities).

In total we have 3071 independent features.

For a Naïve Bayes classifier, we assume features are independent. Therefore, for 3 classes and 10 binary features we have likelihood features $3 * 10 = 30$. Prior probabilities are the same, i.e. 2. In total 32 independent features are required for a Naïve Bayes Classifier.

**A3) (B) 5, 1**

With One-vs-rest approach, we will need 5 classifiers for 5 classes, each corresponding to one of the classes. The predicted class is given by the classifier which has the highest output value. For 2 classes, we will need only 1 classifier since it is binary classification task.

**A4) (C) Maximize, minimize**

The EM algorithm aims to maximize the likelihood of the data by maximizing the variational bound. To do this, the algorithm must minimize the KL divergence between the true posterior distribution and the approximating distribution. Ideally, if the approximating distribution represents the true posterior exactly, the KL divergence will be 0.

**A5) (B) $p(Z|X, \theta)$**

We use q(Z) distribution to simplify the calculation of the posterior $p(Z|X, \theta)$ given X. We use the EM algorithm to try and minimize the distance between these two distributions while maximizing the log-likelihood.

## Section B. True/False

**B1) True.**

K-fold cross validation is a method for evaluating a model by dividing the dataset into k subsets (folds) and training the model on k-1 subsets and evaluating it on the remaining subset. This is repeated for each fold, and the overall performance is the average of the performance on each fold. When k equals the number of samples in the dataset, this is called leave-one-out cross validation, because each sample is used as the validation set once and the rest of the samples are used as the training set.

**B2) False.**

Using the test set to select model parameters can bias the model's error. To get a true, unbiased estimate of the model's performance, we need to use three data splits: a training set to train the model, a validation set to choose the hyperparameters, and a test set to evaluate the model's final performance.

**B3) False.**

We try to find the minimum of a cost function using gradient descent iteratively. The convergence of this algorithm depends on the function of which we are trying to find the global minimum, since there may be a case where the algorithm may get stuck in a local minimum.

**B4) False.**

We need activation functions to inject non-linearity in the classifier and we typically use non-linear transformations. Convolutional layers stacked onto each other will only transform the data linearly as they are just sets of filters that apply convolution operation on input and these filters are learned through training. Therefore, it is not possible to construct a non-linear classification model using just convolutional layers.

**B5) True.**

The model will have less information to train on if the training dataset size is small. If the validation dataset is also small, the error produced on it is less reliable. If the dataset is small, both training and validation sets are small. To avoid this, we use cross-validation, where we create folds from the dataset, using one fold for validation and rest for training, and repeating this for as many folds we have. We average the model performances and get the final performance. In this approach, the model gets to train and validate on a larger number of samples. This approach is computation intensive, and therefore not suggested for a large dataset.

**B6) True.**

For a Naïve Bayes classifier:

$$c = argmax_C \left( p(C = C_j) \prod_i p\left(X = X_i \mid C = C_j\right) \right)$$

In a first-order Markov chain: $p(C = C_j)$ is $p(C = C_j \mid C_{j-1})$

But if we remove the conditional dependence between sequential observations, it becomes $p(C = C_j)$. Therefore, both first-order Markov chain and the Naïve Bayes classifier become the same.

**B7) False.**

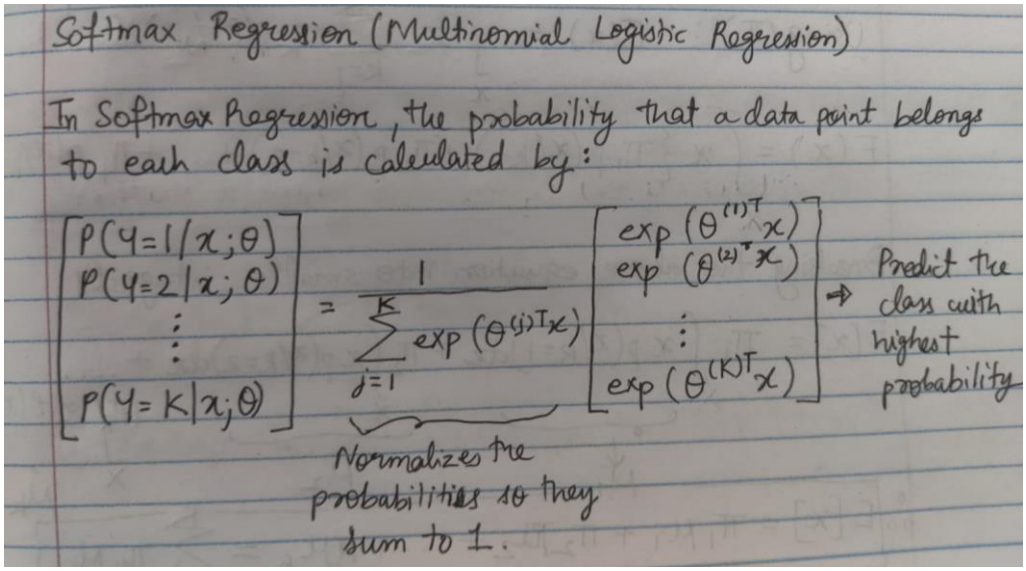
K-means does not follow the same approach to determine cluster locations like GMM, and therefore is not a special application of GMM. In k-means, data points are assigned to the nearest centroid using some distance metric, but in GMM data points are assigned with likelihood of being generated by the current set of clusters.

## Section C. Written Questions

### *1:Logistic Regression/Support Vector Machines*

**Q1)** Logistic regression is inherently designed for binary classification where we model the target using a binomial probability distribution. In its original state, logistic regression cannot be used for multi-class classification and requires tweaking the methodology to support multi-class classification. There are two approaches to apply logistic regression for multi-class classification problems:

1. A popular approach is splitting the multi-class classification problem into multiple binary classification tasks using techniques like **one-vs-rest** and one-vs-one wrapper models.
   In the one-vs-rest technique, taking an example of 5-class classification problem we will train 5 different logistic regression classifiers.
   In the training phase, we will treat input data for a specific class as positive samples (y=1), and input data for rest all the classes as negative samples (y=0). We will repeat this for all the classes.
   In the prediction phase, we will compute the probabilities for all the classes using all the classifiers and the final prediction function will choose the class for which the corresponding classifier gives the highest probability for the class label on the input example.
2. The second approach is using **multinomial logistic regression** based on multinomial probability in order to predict the class probability directly. This is done by changing the loss function to train the model from log-loss to cross-entropy loss, and changing the output from a single probability to an array of probability values (one for each class).




**Q2)** If a logistic regression model achieves small training errors but large validation errors, it is likely overfitting the training data. In this case, increasing the L2 regularization would be appropriate because it would help to reduce the model's complexity and prevent overfitting. This is because L2 regularization adds a penalty to the model's weights, which encourages the model to use only the most important features for making predictions and discourages the use of unnecessary and potentially overfitting features. In terms of the bias-variance trade-off, increasing L2 regularization would help to reduce the model's variance (i.e., the amount by which the model's predictions vary depending on the training data used) and potentially increase its bias (i.e., the error due to the model's assumptions or limitations). This trade-off can be helpful in preventing overfitting and improving the model's generalization performance on unseen data.

**Q3)** We can deal with data that is not linearly separable using SVMs using two different methods:

I) **Soft Margin**: In this approach, we still try to find a hyperplane that separates the data into two classes, but we tolerate misclassifications by introducing a margin. There would be two types of misclassifications after introducing a margin: first where a data point is on the incorrect side of the decision boundary but on the correct side of the margin (first diagram), and second where the data point is on the wrong side of the decision boundary as well as on the wrong side of the margin.

We try to adjust the search for a line for maximum margin and minimum misclassification on training set. This trade-off is an important hyperparameter for SVM and is controlled by the term 'C' in Sklearn. For large values of C, we get a hyperplane with a small margin that gets correctly classifies all training points. Conversely, when C has a very small value, the optimization problem finds a hyperplane with a large margin which may misclassify more training points. Therefore, training an SVM requires one to balance the importance of the regularization term with respect to the loss term using C.

II) **Kernel Trick**: In this approach, we transform the linearly inseparable data by projecting it into a higher dimension and making it linearly separable there, using kernel functions and thus effectively find a non-linear decision boundary. Some of the most popular kernels are the Polynomial and Radial Basis Function(RBF).

## *2:Maximum Likelihood/Maximum A Posteriori Estimation*

**Q1)** Given that the observations are i.i.d., we can estimate the value of the parameter θ using the maximum likelihood estimate (MLE). The likelihood function is the probability of observing the given data given a specific value of the parameter θ. In this case, the likelihood function is given by

$$L(\theta) = \prod_{i=1}^{n} f_x(x_i{:}\,\theta) = \theta^n \cdot \prod_{i=1}^{n} x_i^{-\theta-2}$$

The MLE is the value of θ that maximizes the likelihood function. Taking the natural log of the likelihood function, we can simplify the problem to finding the value of θ that maximizes the log-likelihood function:

$$\ln(L(\theta)) = n*\ln(\theta) + (-\theta - 2)\sum_{i=1}^{n} ln\,(x_i)$$

Taking the derivative of the log-likelihood function with respect to θ and setting it equal to zero, we can find the value of θ that maximizes the log-likelihood function:

$$0 = (1/\theta)*n - \sum_{i=1}^{n} \ln(x_i) \ldots. (I)$$

Solving for θ, we get:

$$\widehat{\boldsymbol{\theta}}_{\mathbf{MLE}} = \frac{\boldsymbol{n}}{\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \mathbf{ln}(\boldsymbol{x_i})}$$

Checking for maxima by taking a derivative w.r.t. θ again on (I):

$$\frac{\partial^2}{\partial\theta^2} ln\big(L(\theta)\big) = -n \,/\, \theta^2 \ldots\ldots. (II)$$

Above value in eqn. (II) is negative, therefore indicating $\hat{\theta}_{\mathrm{MLE}}$ is a maxima.

Thus, the maximum likelihood estimate of θ is the ratio of the number of observations n to the sum of 2 divided by each observation xi.

**Q2)** To find the maximum a posteriori (MAP) estimator of θ, we need to find the value of θ that maximizes the posterior distribution p(θ|x1,x2,...,xn). The posterior distribution is given by Bayes' theorem:

$$p(\theta|x_1,x_2,...,x_n) = (p(x_1,x_2,...,x_n|\theta) * p(\theta)) / p(x_1,x_2,...,x_n)$$

where $p(x_1,x_2,...,x_n|\theta)$ is the likelihood function and p(θ) is the prior distribution for θ.

The denominator $p(x_1,x_2,...,x_n)$ is a normalizing constant that ensures that the posterior distribution integrates to 1. Since we are only interested in the value of θ that maximizes the posterior distribution, we can ignore the normalizing constant and focus on the numerator of Bayes' theorem. The numerator is given by:

$$p(x_1,x_2,...,x_n|\theta) * p(\theta) = \theta^n \prod_{i=1}^{n} x_i^{-\theta-2} \lambda \theta^{-\lambda-2}$$

Taking the natural log of the numerator, we can simplify the problem to finding the value of θ that maximizes the log of the numerator:

$$\ln(p(x_1,x_2,...,x_n|\theta) * p(\theta)) = n*\ln(\theta) + \sum_{1=1}^{n}(-\theta-2)\,ln(x_i) + (-\lambda-2)\,ln(\theta)$$

Taking the derivative of the log of the numerator with respect to θ and setting it equal to zero, we can find the value of θ that maximizes the posterior distribution:

$$0 = (1/\theta)*n - \sum_{i=1}^{n} \ln(x_i) + (-\lambda-2) / \theta \text{ .... (I)}$$

Solving for θ, we get:

$$\widehat{\boldsymbol{\theta}}_{\mathbf{MAP}} = \frac{\boldsymbol{n} - \boldsymbol{\lambda} - \mathbf{2}}{\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{n}} \mathbf{ln}\,(\boldsymbol{x_i})}$$

Checking for maxima by taking a derivative w.r.t. θ again on (I):

$$\frac{\partial^2}{\partial\theta^2} ln\big(L(\theta)\big) = -(n - \lambda - 2) / \theta^2 \text{ ..... (II)}$$

Above value in eqn. (II) is negative, therefore indicating $\hat{\theta}_{\mathrm{MAP}}$ is a maxima.

Thus, the maximum aposteriori estimator of θ is the ratio of the difference between the number of observations n and the parameter governing the prior distribution λ to the sum of 2 divided by each observation xi.

**Q3)** The five main steps to calculate the maximum likelihood estimate (MLE) given a distribution are:

1. Define the likelihood function. This is a function that represents the probability of a set of observations, given a set of parameters. The likelihood function is typically denoted as L(θ|x), where θ is the set of parameters and x is the set of observations.

$$L(\theta) = \Pi f(x,\theta)$$

2. Define the log likelihood function. This is a function that represents the logarithm of the likelihood function. The log likelihood function is often easier to work with than the likelihood function because it is a monotonically increasing function, and it has the same maximum value as the likelihood function.

$$ln\big(L(\theta)\big) = ln\big(\Pi f(x,\theta)\big)$$

3. Specify the set of parameters that will be estimated. In most cases, the MLE will be a function of one or more unknown parameters, and these parameters will need to be estimated from the data. Choose an

optimization algorithm. There are many different optimization algorithms that can be used to find the maximum likelihood estimate, and the choice of algorithm will depend on the specific problem at hand. Some common optimization algorithms include gradient descent, Newton's method, and the Expectation-Maximization (EM) algorithm. In our case, we have differentiated to get the maxima of the function.

$$\frac{\partial}{\partial\theta} ln\big(L(\theta)\big)$$

4. Optimize the log likelihood function to find the maximum likelihood estimate. Once an optimization algorithm has been chosen, the log likelihood function can be optimized to find the set of parameters that maximize the likelihood of the observed data. This set of parameters is the maximum likelihood estimate.

$$step(3) = 0; find\ \hat{\theta}_{\mathrm{MLE}}$$

5. Check if $\frac{\partial^2}{\partial\theta^2} ln\big(L(\theta)\big) < 0\ [Maximized]$

Once the maximum likelihood estimate has been found, it can be used to make predictions about future observations, or to perform hypothesis testing.

### *3:Naive Bayes Classifier*

**Q1)** There are two types of independent parameters for a Naïve Bayes Classifier: Likelihood parameters & Prior parameters:

Likelihood parameters can be written as:

$$p(X = X_i|C = C_j)$$

where:

- $X_i$ is the i[th] feature, $i \in (1, 2, 3)$
- $C_j$ is the j[th] class, $j \in (1, 2)$

The total number of likelihood parameters will be (3 – 1)*2 = 4. (The third likelihood parameter can be computed by using the other two likelihood parameters)

Prior parameters are one per class. The total number of prior parameters will be 1 from 2 classes since if we know prior of 1 class, we can compute the second prior.

Therefore, the total number of independent parameters will be **4 + 1 = 5**.

**Q2)** To predict the MLE for a class, we have the below relation:

$$c = argmax_C \Big(p\big(X = X_i|C = C_j\big) \times p\big(C = C_j\big)\Big)$$

With the above relation, we will estimate the likelihood probabilities and the prior class probabilities below, using the data provided in the question:

For “Size” feature:

| Size | Class = Yes | Class = No |
|---|---|---|
| Large | 3 | 3 |
| Small | 1 | 3 |

| Size | Class = Yes | Class = No |
|---|---|---|
| Large | 0.75 | 0.5 |
| Small | 0.25 | 0.5 |

For “Color” feature:

| Color | Class = Yes | Class = No |
|---|---|---|
| Green | 0 | 5 |
| Red | 4 | 1 |

| Color | Class = Yes | Class = No |
|---|---|---|
| Green | 0 | 0.833 |
| Red | 1 | 0.167 |

For “Shape” feature:

| Shape | Class = Yes | Class = No |
|---|---|---|
| Irregular | 1 | 4 |
| Round | 3 | 2 |

| Shape | Class = Yes | Class = No |
|---|---|---|
| Irregular | 0.25 | 0.667 |
| Round | 0.75 | 0.333 |

For class priors:

| Class | Prior |
|---|---|
| Yes | 4 |
| No | 6 |

| Class | Prior |
|---|---|
| Yes | 0.4 |
| No | 0.6 |

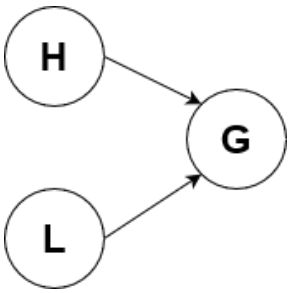
**Q3)**Given x = {Small, Red, Round},

$\mathrm{p}(\mathrm{y} = \mathrm{No}|\mathrm{x}) = \mathrm{p}(\mathrm{Size} = \mathrm{Small}|\mathrm{y} = \mathrm{No}) \times \mathrm{p}(\mathrm{Color} = \mathrm{Red}|\mathrm{y} = \mathrm{No}) \times \mathrm{p}(\mathrm{Shape} = \mathrm{Round}|\mathrm{y} = \mathrm{No}) \times \mathrm{p}(\mathrm{y} = \mathrm{No})$
$\mathrm{p}(\mathrm{y} = \mathrm{No}|\mathrm{x}) = 0.5 \times 0.167 \times 0.333 \times 0.6$
$\mathbf{p}(\mathbf{y} = \mathbf{No}|\mathbf{x}) = \mathbf{0.0166}$

$\mathrm{p}(\mathrm{y} = \mathrm{Yes}|\mathrm{x}) = \mathrm{p}(\mathrm{Size} = \mathrm{Small}|\mathrm{y} = \mathrm{Yes}) \times \mathrm{p}(\mathrm{Color} = \mathrm{Red}|\mathrm{y} = \mathrm{Yes}) \times \mathrm{p}(\mathrm{Shape} = \mathrm{Round}|\mathrm{y} = \mathrm{Yes}) \times \mathrm{p}(\mathrm{y} = \mathrm{Yes})$
$\mathrm{p}(\mathrm{y} = \mathrm{Yes}|\mathrm{x}) = 0.25 \times 1 \times 0.75 \times 0.4$
$\mathbf{p}(\mathbf{y} = \mathbf{Yes}|\mathbf{x}) = \mathbf{0.075}$
Since $\mathrm{p}(\mathrm{y} = \mathrm{Yes}|\mathrm{x}) > \mathrm{p}(\mathrm{y} = \mathrm{No}|\mathrm{x})$, our Naïve Bayes classifier will predict $\mathrm{y} = \mathrm{Yes}$.

## *4:Probabilistic Graphical Models*

**Q1)** In our case, the Bayesian Network is as follows:




The joint probability $p(G, H, L)$ is:

$$\boldsymbol{p(G, H, L) = p(H) \times p(L) \times p(G|H, L)}$$

**Q2)** From Q1. we know that:

$$p(G, H, L) = p(G|H, L) \cdot p(H) \cdot p(L)$$

On assuming no independence relations (given) we can deduce that:

$$p(H) = p(G, H, L) + p(\overline{G}, H, L) + p(G, H, \overline{L}) + p(\overline{G}, H, \overline{L})$$

Hence, $p(H) = p_1 + p_2 + p_4 + p_6$ (given) (1)

$$p(L) = p(G, H, L) + p(\overline{G}, H, L) + p(G, \overline{H}, L) + p(\overline{G}, \overline{H}, L)$$

Hence, $p(L) = p_1 + p_2 + p_3 + p_5$ (given) (2)

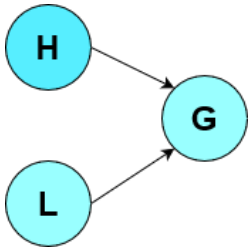
Also, $p(G|H, L) = p(G, H, L) = p_1$ (given) (3)

Substituting (1), (2) and (3) in our Joint Probability equation we get:

$$p(G, H, L) = p_1 \cdot (p_1 + p_2 + p_4 + p_6) \cdot (p_1 + p_2 + p_3 + p_5)$$

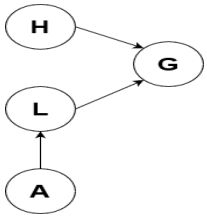
Hence, we can observe that in order to specify $p(G, H, L)$ we need 6 parameters $(p_1, p_2, p_3, p_4, p_5, p_6)$

**Q3)**



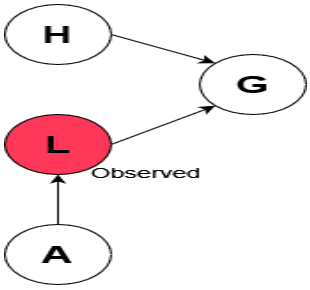

G is a **head-to-head** node and it is **observed**, therefore it is **unblocked**. Since there is an unblocked path between H and L, they are not independent.

**Q4)** Adding node for Lauren's assignments:



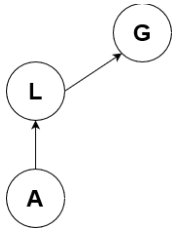

The joint probability $p(G, H, L, A)$ is

$$p(G, H, L, A) = p(H) \times p(A) \times p(L|A) \times p(G|H, L)$$




L is a **head-to-tail node** and it is **observed**, therefore it is **blocked**. A & G do not have any unblocked paths between them, and therefore are **independent**. Hence, if L is given, G and A are independent.

**Q5)** The new Bayesian Network after eliminating H is given as follows:




The joint probability $p(G, L, A)$ is:

$$\boldsymbol{p(G, L, A) = p(A) \times p(L|A) \times p(G|L)}$$

On assuming no independence relations (given) we can deduce that:

$\mathrm{p(A) = p(G, L, A) + p(\overline{G}, L, A) + p(G, \overline{L}, A) + p(\overline{G}, \overline{L}, A)}$

Hence, $\mathrm{p(A) = q_1 + q_2 + q_3 + q_5}$ (given) (1)

$\mathrm{p(L|A) = p(G = G, L, A) + p(G = \overline{G}, L, A)}$

Hence, $\mathrm{p(L|A) = q_1 + q_2}$ (given) (2)

$\mathrm{p(G|L) = p(G, L, A = A) + p(G, L, A = \overline{A})}$

Hence, $\mathrm{p(L|A) = q_1 + q_4}$ (given) (3)

Substituting (1), (2) and (3) in our Joint Probability equation we get:

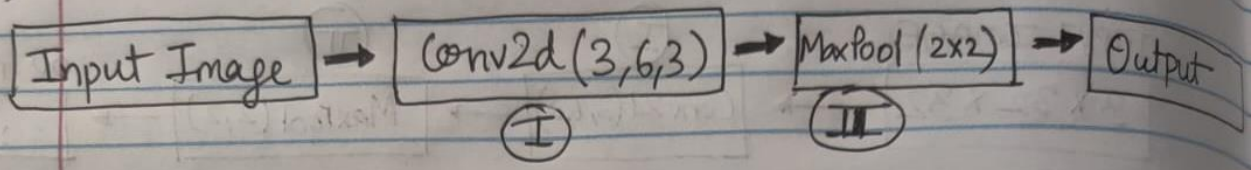
$\mathrm{p(G, L, A) = (q_1 + q_4) \cdot (q_1 + q_2) \cdot (q_1 + q_2 + q_3 + q_5)}$

Hence, we can observe that in order to specify $\mathrm{p(G, L, A)}$ we need 6 parameters: $(\mathrm{q_1, q_2, q_3, q_4, q_5})$

***5:Neural Networks/Convolutional Neural Networks***

**Q1)**

5.1) Given Image Shape : 3X32X32




(I) Size of kernel in Conv2d: 3×3 ⇒ K=3
Stride : 1×1 ⇒ S=1 (Default)
Dimensions of Input Image : $D_1 \times H_1 \times W_1$ : 3×32×32
Padding : 0 ⇒ P=0 (Default)
Dimensions of Output Image : $D_2 \times H_2 \times W_2$

$H_2 = (H_1 - K + 2P/S) + 1 = \frac{32-3}{1} + 1 = 30$
$W_2 = (W_1 - K + 2P/S) + 1 = \frac{32-3}{1} + 1 = 30$
$D_2$ = No. of filters = 6

∴ Dimensions of Output Image : (6×30×30) of (I)

(II) Size of Kernel in Max Pooling : 2×2 ⇒ K=2
Stride : 2×2 ⇒ S=2
Dimensions of Input Image : $D_2 \times H_2 \times W_2 = (6 \times 30 \times 30)$
Padding = 0 = P
Dimensions of final Output Image : $D_3 \times H_3 \times W_3$

$H_3 = (H_2 - K + 2P/S) + 1 = \frac{30-2}{2} + 1 = 15$
$W_3 = (W_3 - K + 2P)/S + 1 = \frac{30-2}{2} + 1 = 15$
$D_3 = 6$

∴ Final size of the image will become (6×15×15)

**Q2)** Given Feature Map:

| | | | |
|---|---|---|---|
| 1 | 1 | 2 | 4 |
| 5 | 6 | 7 | 8 |
| 3 | 2 | 1 | 0 |
| 1 | 2 | 3 | 4 |

After applying a (2, 2) max pooling layer to the given feature map, we will get the following new feature map:

| | |
|---|---|
| **6** | **8** |
| **3** | **4** |

This is because the max pooling operation will take the maximum value from each 2x2 region of the input feature map and produce a new feature map with these maximum values.

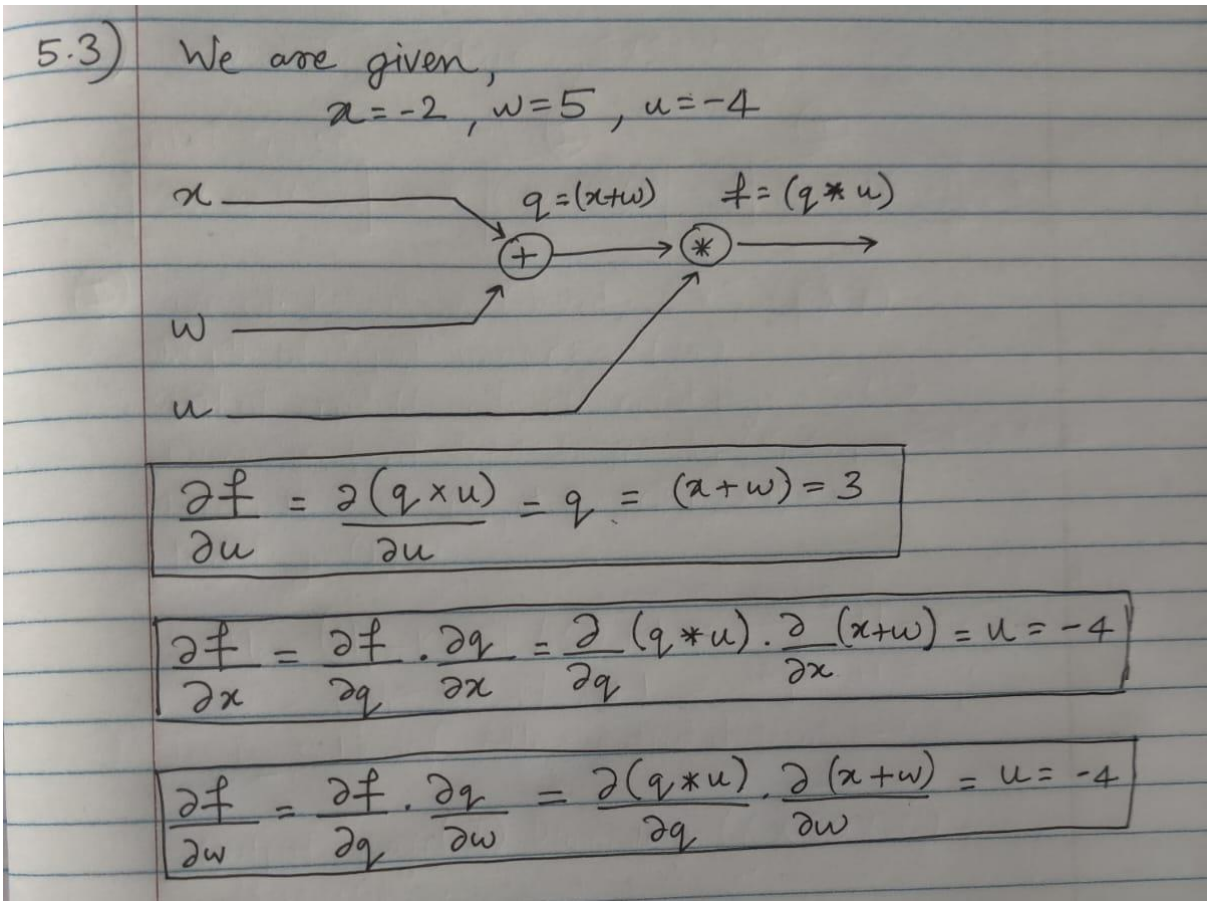
If we change the max pooling layer to an average pooling layer, the resulting feature map would be:

| | |
|---|---|
| **3.25** | **5.25** |
| **2** | **2** |

This is because the average pooling operation will take the average value from each 2x2 region of the input feature map and produce a new feature map with these average values. Calculations:

1. (1+1+5+6)/4 = 3.25
2. (2+4+7+8)/4 = 5.25
3. (3+2+1+2)/4 = 2
4. (1+0+3+4)/4 = 2

**Q3)**



***6:k-means***

**Q1)**

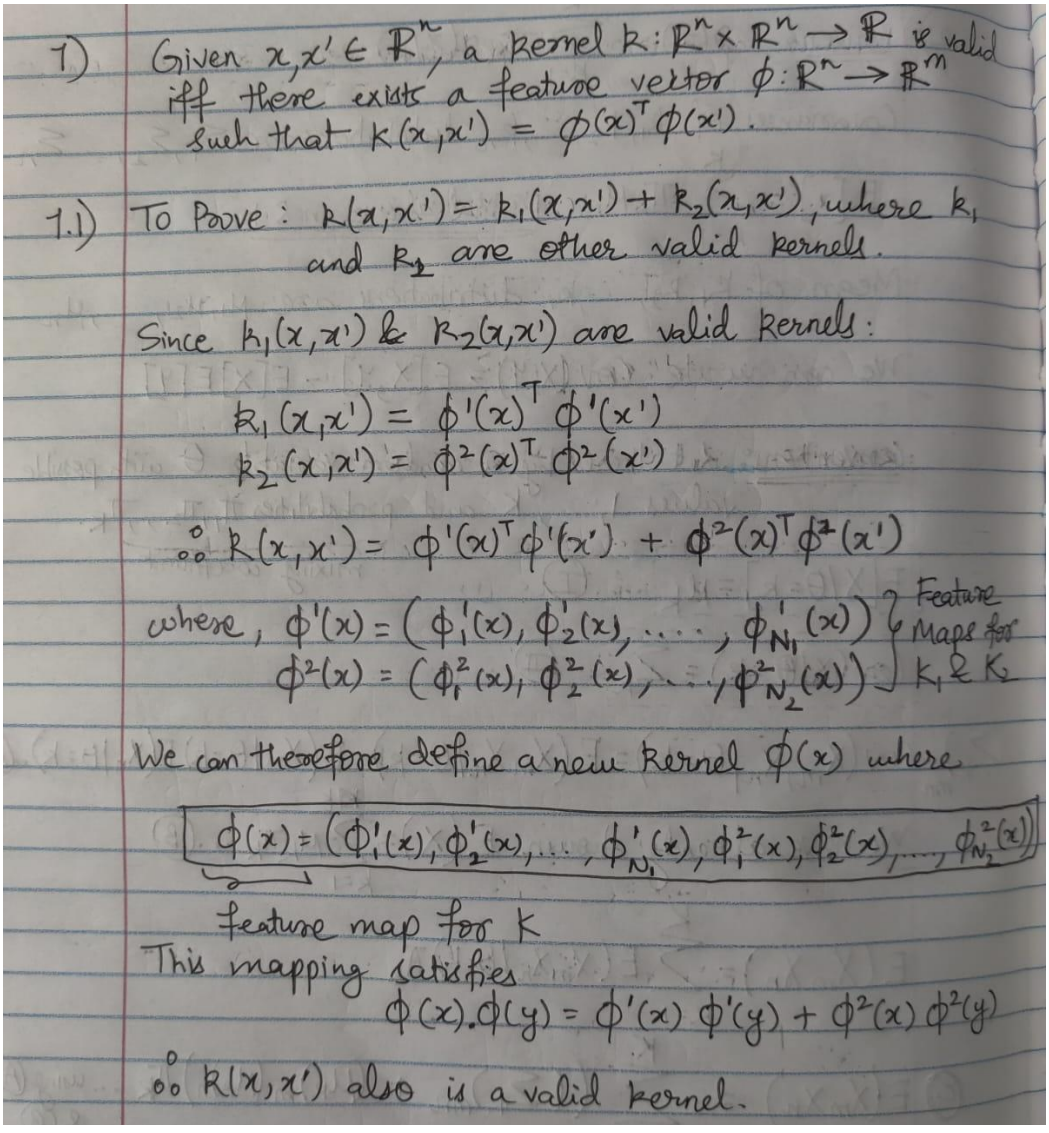
There are a total of 6 points in the given set, and we want to choose 3 of them to be the initial centroids for the k-means algorithm. This can be done in $C_6^3$= 20 ways. Therefore, there are 20 possible 3-starting configurations for the k-means algorithm.

**Q2)**

| **3-partition** | **Stable?** | **An example 3-starting configuration** |
|---|---|---|
| {a, b}, {c, f}, {d, e} | Yes | {b, e, f} |
| {a, b, e}, {c, d}, {f} | No | none |
| {a, d}, {b, c}, {e, f} | Yes | {a, b, e} |
| {a, b, d}, {e}, {c, f} | No | none |
| {a, d, e}, {b, c}, {f} | Yes | {c, d, f} |

***7:Kernel Methods***

**Q1)**



We can also prove it in Feature Vector Space Representation

$$\phi(x) = [\phi^1(x), \phi^2(x)]$$

$$k(x,x') = \langle \phi(x)^T, \phi(x') \rangle$$

$$= \langle [\phi^1(x)^T \; \phi^2(x)^T], [\phi^1(x') \; \phi^2(x')] \rangle$$

$$= \langle \phi^1(x)^T \phi^1(x') \rangle + \langle \phi^2(x)^T \phi^2(x') \rangle$$

$$\boxed{k(x,x') = k_1(x,x') + k_2(x,x')} \quad \therefore k(x,x') \text{ is a valid kernel.}$$

**Q2)**

7.2) To Prove: $k(x,x') = k_1(x,x')\, k_2(x,x')$ where $k_1$ & $k_2$ are valid kernels.

Since $k_1(x,x')$ & $k_2(x,x')$ are valid kernels:

$$k_1(x,x') = \phi^1(x)^T \phi^1(x'); \quad k_2(x,x') = \phi^2(x)^T \phi^2(x')$$

$$\therefore k(x,x') = \phi^1(x)^T \phi^2(x)^T \phi^1(x') \phi^2(x')$$

This can be written as:

$$k(x,x') = \begin{bmatrix} \phi_1^1(x)^T \phi_1^2(x)^T \phi_1^1(x') \phi_1^2(x') & \cdots & \phi_1^1(x)^T \phi_1^2(x)^T \phi_{N_2}^1(x') \phi_{N_2}^2(x') \\ \phi_2^1(x)^T \phi_2^2(x)^T \phi_1^1(x') \phi_1^2(x') & \cdots & \phi_2^1(x)^T \phi_2^2(x)^T \phi_{N_2}^1(x') \phi_{N_2}^2(x') \\ \vdots & & \vdots \\ \phi_{N_1}^1(x)^T \phi_{N_1}^2(x)^T \phi_1^1(x') \phi^2(x') & \cdots & \phi_{N_1}^1(x)^T \phi_{N_1}^2(x)^T \phi_{N_2}^1(x') \phi_{N_2}^2(x') \end{bmatrix}$$

↑ gram matrix representation

The above gram matrix representation is a Hadamard product (or element by element product) of $K_1$ & $K_2$.

If $K_1$ is represented by a covariance matrix of $(X_1, X_2, \ldots, X_n)$

If $K_2$ is represented by a covariance matrix of $(Y_1, Y_2, \ldots, Y_n)$

then

$K = K_1 \odot K_2$ is represented simply by covariance matrix of $(X_1Y_1, X_1Y_2, \ldots, X_nY_n)$, implying that it is symmetric & positive definite.

K is also the Schur product.

We can also show this in Feature Vector Space Representation

$$\phi(x)_{ij} = \phi^1(x)_i \; \phi^2(x)_j$$

where $i = 1, 2, \ldots, N_1$ & $j = 1, 2, \ldots, N_2$

$$K(x, x') = \langle \phi(x)^T \phi(x') \rangle$$

$$= \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \phi(x)^T_{ij} \; \phi(x')_{ij}$$

$$= \sum_{i=1}^{N_1} \phi^1(x)^T_i \; \phi^1(x')_i \sum_{j=1}^{N_2} \phi^2(x)^T_j \; \phi^2(x')_j$$

$$\boxed{K(x, x') = k_1(x, x') \; k_2(x, x')}$$

$\therefore K(x, x')$ is a valid kernel.

**Q3)**

1.3) To Prove: $k(x,x') = f(x)\, k_1(x,x')\, f(x')$, where $f: R^n \to R$

Since $k(x,x')$ is the product of $f(x)$, $k_1(x,x')$, and $f(x')$, we can write $k(x,x')$ as:

$$k(x,x') = f(x)\underbrace{(\phi'(x)^T \phi'(x'))}_{k_1 \text{ is a valid kernel}} f(x')$$

$$= \underbrace{f(x)\,\phi'(x)^T}_{\phi(x)^T} \quad \underbrace{\phi'(x')\,f(x')}_{\phi(x')} \quad \ldots (I)$$

Expression (I) is of the form $(\phi(x)^T * \phi(x'))$ where $\phi(x)$ is the product of the scalar function $f$ and the feature vector of $x$ corresponding to $k_1$.

∴ We have shown that there exists a feature vector $\phi(x)$ such that $k(x,x') = \phi(x)^T \phi(x')$. $\phi$ is just another kernel transformation in that domain.

$$\boxed{\phi(x) = (f(x)\,\phi_1'(x),\ f(x)\,\phi_2'(x),\ \ldots,\ f(x)\,\phi_N'(x))}$$

∴ $k(x,x')$ is a valid kernel.

**Q4)**

7.4) To Prove: $k(x,x') = \exp\{k_1(x,x')\}$, where $k_1$ is another valid kernel

We will first show ....(I)

$$k(x,x') = \alpha k_1(x,x') + \beta k_2(x,x')$$

where $k_1$ & $k_2$ are valid kernels

(I) Since $\alpha k_1(x,x') = \langle \sqrt{\alpha}\phi^1(x), \sqrt{\alpha}\phi^1(x') \rangle$ ...(A)

$$\beta k_2(x,x') = \langle \sqrt{\beta}\phi^2(x), \sqrt{\beta}\phi^2(x') \rangle$$ ..(B)

Then, $k(x,x') = \langle \sqrt{\alpha}\phi^1(x), \sqrt{\alpha}\phi^1(x') \rangle + \langle \sqrt{\beta}\phi^2(x), \sqrt{\beta}\phi^2(x') \rangle$ using (A) & (B)

$$k(x,x') = \langle [\sqrt{\alpha}\phi^1(x)\sqrt{\beta}\phi^2(x)], [\sqrt{\alpha}\phi^1(x')\sqrt{\beta}\phi^2(x')] \rangle$$

∴ $k$ can be expressed as an inner product and is a valid kernel.

(II) We also know from the proof in 7.2 that

$$\boxed{k(x,x') = k_1(x,x')\,k_2(x,x')} \quad \ldots (II)$$

(III) Now, we will show

$$k(x,x') = f(k_1(x,x'))$$ where $f$ is a polynomial with valid kernel +ve co-efficients

Since each polynomial term is a product of kernels with a +ve co-efficient using (I) & (II), we can deduce that

$k(x,x') = f(k_1(x,x'))$ is a valid kernel (III)

Now to prove that $k(x,x') = \exp\{k_1(x,x')\}$ we can write

$$\exp(x) = \lim_{i\to\infty}\left(1 + x + \ldots + \frac{x_i}{i!}\right)$$ (Taylor Series Expansion)

Now using (III), we can say

$$\boxed{k(x,x') = \lim_{i\to\infty} k_i(x,x')}$$

Since $k(x,x')$ is a sum of valid kernels ∴ it is a valid kernel.

**Q5)**

7.5) To Prove: $k(x,x') = \exp\left\{-\frac{\|x-x'\|^2}{2\sigma^2}\right\}$

$$\|x-x'\|^2 = x^Tx + (x')^Tx' - 2x^Tx'$$

$$\therefore k(x,x') = \exp\left(\frac{-x^Tx}{2\sigma^2}\right)\exp\left(\frac{x^Tx'}{\sigma^2}\right)\exp\left(\frac{-(x')^Tx'}{2\sigma^2}\right) \quad \text{....(A)}$$

We have proven that below kernels are valid:

(I) $K(x,x') = x^Tx' \Rightarrow$ Linear Kernel $\Rightarrow$ Rename it to $K_1(x,x')$

(II) $k(x,x') = \exp(k_1(x,x')) \Rightarrow$ From 7.4

(III)

$\therefore$ Eqn (A) $\Rightarrow$ $k(x,x') = f(x)\,k_1(x,x')\,f(x')$ is valid kernel

New Kernel $\Rightarrow$ $k(x,x') = \exp\left\{-\frac{1}{2\sigma^2}(k(x,x')+k(x',x')-2k(x,x'))\right\}$

***8:Mixture Models***

***Q1)***

8.1) Given the following mixture distribution of K probability densities:

$$p(x) = \sum_{k=1}^{K} \pi_k p(x/k) \quad \ldots (I)$$

where $\pi_k$ is the mixing co-efficient for the probability $p(x/k)$, and the elements of $x$ could be discrete or continuous or a combination of these.

Mean of $k_1, k_2, \ldots, k_n$ distributions are $\mu_1, \mu_2, \ldots \mu_K$.

We know that $E[x]$ for any random variable $x$ is given by:

$$E(x) = \int_x x\, p(x)\, dx$$

Using (I): $$E(x) = \int_x x \sum_{k=1}^{K} \pi_k p(x/k)\, dx$$

$$E(x) = \int_X x \left\{ \pi_1 p(x/k=1) + \pi_2 p(x/k=2) + \ldots + \pi_K p(x/k=K) \right\} dx$$

Breaking the above equation into smaller integrals:

$$E[x] = \pi_1 \int_X x\, p(x/k=1)\, dx + \pi_2 \int_X x\, p(x/k=2)\, dx + \ldots + \pi_K \int_X x\, p(x/k=K)\, dx$$

$$\mu_1 \qquad \mu_2 \qquad \mu_K$$

$$E[x] = \pi_1 \mu_1 + \pi_2 \mu_2 + \ldots + \pi_K \mu_K = \sum_{k=1}^{K} \pi_k \mu_k$$

**Q2)**

8.2) Given: $p(x) = \sum_{k=1}^{K} \pi_k p(x/k)$

Covariances of $k_1, k_2, \ldots, k_n$ distributions are $\Sigma_1, \Sigma_2, \ldots \Sigma_n$.

$E[x] = \sum_{k=1}^{K} \pi_k \mu_k$ proved above in 8.1

Mean of $k_1, k_2, \ldots, k_n$ distributions are $\mu_1, \mu_2, \ldots, \mu_n$

We can write: $Cov(X,Y) = E[X,Y] - E[X]E[Y]$

Conventions: Let there by a random variable $\theta$ with possible values $1, \ldots, K$ and probabilities $\pi_1, \pi_2, \ldots, \pi_K$.

mixing co-efficient

$$E[X/\theta=k] = \mu_k \quad \ldots (I)$$

$$Cov(X/\theta=k) = \Sigma_k \quad \ldots (II)$$

$$(\Sigma_k)_{mn} = Cov(X_m, X_n/\theta=k) = E(X_m X_n/\theta=k) - E(X_m/\theta=k)\,E(X_n/\theta=k) \quad \ldots (A)$$

We have already shown $E(X) = \sum_{k=1}^{K} \mu_k \pi_k \quad \ldots (B)$

$$E(X_m X_n) = \sum_{k=1}^{K} E(X_m X_n/\theta=k)\,\pi_k$$

$$(C)\ E(X_m X_n) = \sum_{k=1}^{K} \left( (\Sigma_k)_{mn} + (\mu_k)_m (\mu_k)_n \right) \pi_k \quad \ldots \text{using } (A) \text{ \& } (B)$$

Continuation:

Using (C) & (B)

$$Cov(X_m X_n) = E(X_m X_n) - E(X_m)E(X_n)$$

$$= \sum_{k=1}^{K} ((\Sigma_k)_{mn} + (\mu_k)_m (\mu_k)_n) - \sum_{k=1}^{K} (\mu_k)_m \pi_k \sum_{k=1}^{K} (\mu_k')_n \pi_k'$$

This can be expressed as

$$Cov(X_m X_n) = E[Cov(X_m X_n | \theta)] + Cov(E(X_m|\theta)\, E(X_n|\theta))$$

"Law of Total Covariance"

$$\therefore Cov[X] = \sum_{k=1}^{K} \Sigma_k \pi_k + \sum_{k=1}^{K} \mu_k^T \mu_k \pi_k - \mu^T \mu$$

$$\mu^T \mu \rightarrow E[X]E[X]^T$$

$$\boxed{Cov[X] = \sum_{k=1}^{K} \pi_k (\Sigma_k + \mu_k^T \mu_k) - E[X]E[X]^T}$$

Hence Proved

***9:Approximate Inference***

***Q1)***

9.1) We are given a fully Bayesian Model

$X = \{x_1, x_2, \ldots, x_N\}$

$Z = \{z_1, z_2, \ldots, z_N\}$ latent variables & parameters

Model specifies joint distribution $p(X, Z)$

Using Bayes' Theorem:

$$p(X) = \frac{p(X,Z)}{p(Z|X)} = \frac{\frac{p(X,Z)}{q(Z)}}{\frac{p(Z|X)}{q(Z)}}$$

where $q(Z)$ is the distribution of the latent variables

Taking ln on both sides:

$$\ln(p(X)) = \ln\left(\frac{p(X,Z)}{p(Z|X)}\right) = \ln\left(\frac{p(X,Z)}{q(Z)}\right) - \ln\left(\frac{p(Z|X)}{q(Z)}\right)$$

Multiplying by $q(Z)$ on both sides:

$$q(Z)\ln(p(X)) = q(Z)\ln\left(\frac{p(X,Z)}{q(Z)}\right) - q(Z)\ln\left(\frac{p(Z|X)}{q(Z)}\right)$$

Integrating w.r.t. $Z$ on both sides:

$$\int q(z)\ln(p(X))dz = \int q(z)\ln\left(\frac{p(X,Z)}{q(Z)}\right)dz - \int q(z)\ln\left(\frac{p(Z|X)}{q(Z)}\right)dz$$

$\ln(p(X))\int q(z)dz$ → $L(q)$ given; $KL(q\|p)$ given

Since $q(z)$ is a distribution, $\int q(z)dz = 1$

∴ we get,

$$\boxed{\ln(p(X)) = L(q) + KL(q\|p)}$$

***10:Hidden Markov Models***

**Q1)** To Prove: If π or A are initially set to zero, then they will remain zero in all subsequent updates of the EM algorithm.

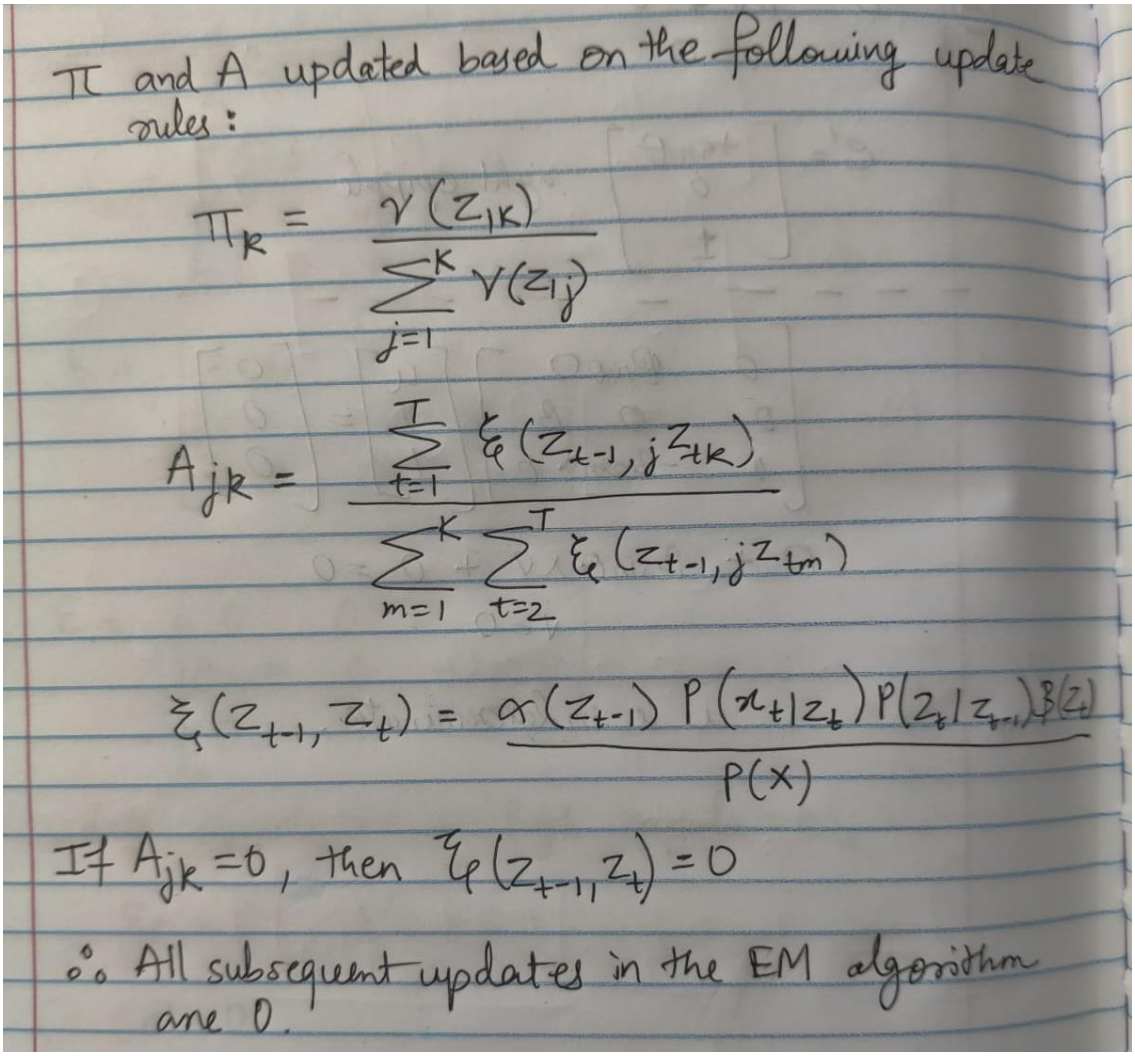
EM algorithm for a Hidden Markov Model:

E-step: Computation of the expectation of the complete-data log likelihood for model parameters

M-step: Updating model parameters to maximize expectation

For E-step: Since the initial values of the model parameters include elements that are set to zero, these zero elements will not contribute to the expected value of the complete-data log likelihood. Therefore, the updated model parameters in the M-step will not be affected by these zero elements.

For M-step: Since the initial values of the model parameters include elements that are set to zero, these zero elements will not affect the maximization of the expected value of the complete-data log likelihood. Therefore, the updated model parameters will not be affected by these zero elements, and they will remain zero in all subsequent updates of the EM algorithm.




Therefore, if any elements of the parameter π or A for a Hidden Markov Model are initially set to zero, then those elements will remain zero in all subsequent updates of the EM algorithm.

**Q2)** A Markov model is a mathematical model used to describe the behavior of a system where the future state of the system is dependent only on its current state, not on its past history. In this cake factory example, the state of the system is determined by the positions of the Chocolate and Caramel levers, and the future state of the system is determined by the actions of the controller staff in the control room. The possible states of the system are as follows:

- Chocolate lever on, Caramel lever on (Chocolate+Caramel cake) **(ChCa)**
- Chocolate lever on, Caramel lever off (Chocolate cake) **(Ch)**
- Chocolate lever off, Caramel lever on (Caramel cake) **(Ca)**
- Chocolate lever off, Caramel lever off (Plain cake) **(P)**

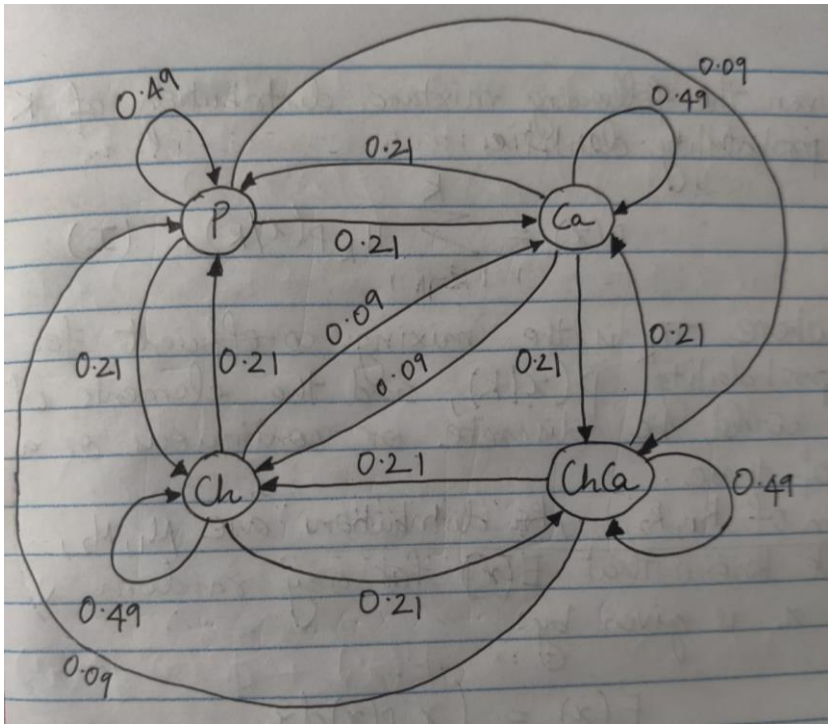
The prior state distribution is the initial probability distribution over the possible states of the system. In this case, since the Chocolate and Caramel levers are independently switched on or off at the beginning of the day with a 50% chance each, the prior state distribution is as follows:

- Chocolate lever on, Caramel lever on: 0.25
- Chocolate lever on, Caramel lever off: 0.25
- Chocolate lever off, Caramel lever on: 0.25
- Chocolate lever off, Caramel lever off: 0.25

This transition matrix indicates the probabilities of transitioning from one state to another when the controller staff in the control room switches the positions of the Chocolate and Caramel levers. The state transition probabilities are ((the probability of both levers being switched is 0.09, the probability of one being switched is 0.7*0.3=0.21, and the probability of neither being switched is 0.49).

| | **P** | **Ch** | **Ca** | **ChCa** |
|---|---|---|---|---|
| **P** | 0.49 | 0.21 | 0.21 | 0.09 |
| **Ch** | 0.21 | 0.49 | 0.09 | 0.21 |
| **Ca** | 0.21 | 0.09 | 0.49 | 0.21 |
| **ChCa** | 0.09 | 0.21 | 0.21 | 0.49 |

The Markov model for the cake factory can be represented as a directed graph, where the nodes represent the possible states of the system and the edges represent the transitions between states. The prior state distribution can be represented as the initial probabilities of being in each state, and the transition matrix can be represented as the probabilities of transitioning from one state to another along the edges of the graph. For example, the Markov model for the cake factory can be represented as follows:



**Q3)** The probability that the machine will produce the sequence {Plain, Chocolate, Chocolate, Chocolate+Caramel} in this order depends on the initial state of the system and the specific sequence of transitions between the states.

Assuming the initial state is randomly selected from the prior state distribution, the probability of producing the sequence {Plain, Chocolate, Chocolate, Chocolate+Caramel} in this order is as follows:

1. The initial state is Plain, with a probability of 0.25.
2. The next state is Chocolate, with a probability of 0.21 (transitioning from Plain to Chocolate with a probability of 0.21 according to the transition matrix).
3. The next state is Chocolate, with a probability of 0.49 (transitioning from Chocolate to Chocolate with a probability of 0.49 according to the transition matrix).
4. The next state is Chocolate+Caramel, with a probability of 0.21 (transitioning from Chocolate to Chocolate+Caramel with a probability of 0.21 according to the transition matrix).

The overall probability of producing the sequence {Plain, Chocolate, Chocolate, Chocolate+Caramel} is **P(P, Ch, Ch, ChCa) = P(P)*P(Ch|P)*P(Ch|Ch)*P(ChCa|Ch)** in this order is the product of the individual probabilities:

which is equal to **0.25 * 0.21 * 0.49 * 0.21 = 0.0054**.

Note that this is just one possible sequence of states that can result in the production of the sequence {Plain, Chocolate, Chocolate, Chocolate+Caramel} of cakes. There may be other sequences of states that result in the same sequence of cakes with different probabilities.