**1. a) Consider a two-class classification problem with P(ω1) = 0.6, P(ω2) = 0.4, p(x|ω1) = and . Find the decision regions and boundaries for the minimum error rate Bayes classifier.**

**Sol:** The decision region R1 is the set of points x for which P(ω1|x) > P(ω2|x). The decision region R2 is the set of points x for which P(ω2|x) > P(ω1|x). The decision boundary is the set of points x for which P(ω2|x) = P(ω1|x). Begin with the decision boundary:

P(ω1|x) = P(ω2|x) => p(x|ω1)P(ω1) = p(x|ω2)P(ω2),

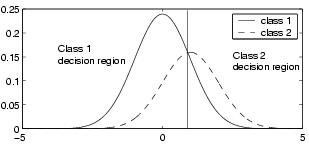
where we used the Bayes formula and multiplied with p(x). Continuing

p(x| ω1)P(ω1) = p(x|ω2)P(ω2) => ln[p(x|ω1)P(ω1)] = ln[p(x|ω2)P(ω2)]

=> −x2/2 + ln 0.6 = −(x − 1)2/2 + ln 0.4 => x2 − 2 ln 0.6 = x2 − 2x + 1 − 2 ln 0.4.

The decision boundary is x\* = **0.5 + ln0.6 − ln 0.4 ~ 0.91**.

R1 = {x : x < x\*}, R2 = {x : x > x\*}.



**1 b) Derive the optimum decision rule to minimize risk R using Bayes decision rule for two-category classification.**

**Sol:** Let us consider these results when applied to the special case of two-category classification problems. Here action *α*1 corresponds to deciding that the true state of nature is *ω*1, and action *α*2 corresponds to deciding that it is *ω*2. For notational simplicity, let *λij* = *λ*(*αi|ωj*) be the loss incurred for deciding *ωi* when the true state of nature is *ωj*.

*R*(*α*1*|***x**) = *λ*11*P*(*ω*1*|***x**) + *λ*12*P*(*ω*2*|***x**) and

*R*(*α*2*|***x**) = *λ*21*P*(*ω*1*|***x**) + *λ*22*P*(*ω*2*|***x**)*.*

There are a variety of ways of expressing the minimum-risk decision rule, each having its own minor advantages. The fundamental rule is to decide *ω*1 if *R*(*α*1*|***x**) *< R*(*α*2*|***x**). In terms of the posterior probabilities, we decide *ω*1 if

(*λ*21 *− λ*11)*P*(*ω*1*|***x**) *>* (*λ*12 *− λ*22)*P*(*ω*2*|***x**)*.*

Ordinarily, the loss incurred for making an error is greater than the loss incurred for being correct, and both of the factors *λ*21 *− λ*11 and *λ*12 *− λ*22 are positive. Thus in practice, our decision is generally determined by the more likely state of nature, although we must scale the posterior probabilities by the loss differences. By employing Bayes’ formula, we can replace the posterior probabilities by the prior probabilities and the conditional densities. This results in the equivalent rule, to decide *ω*1 if

(*λ*21 *− λ*11)*p*(**x***|ω*1)*P*(*ω*1) *>* (*λ*12 *− λ*22)*p*(**x***|ω*2)*P*(*ω*2)*,* and *ω*2 otherwise.

Another alternative, which follows at once under the reasonable assumption that *λ*21 *> λ*11, is to decide *ω*1 if



**2 a) Define the following terms with an example.**

1. **Supervised Learning:**

Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.

Y = f(X)

The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

It is called supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers, the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance. Supervised learning problems can be further grouped into regression and classification problems.

* Classification: A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”.
* Regression: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

1. **Unsupervised Learning**

Unsupervised learning is where you only have input data (X) and no corresponding output variables. The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data. These are called unsupervised learning because unlike supervised learning above there is no correct answers and there is no teacher. Algorithms are left to their own devises to discover and present the interesting structure in the data. Unsupervised learning problems can be further grouped into clustering and association problems.

* Clustering: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
* Association:  An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

1. **Reinforcement Learning**

Reinforcement learning (RL) is learning by interacting with an environment. An RL agent learns from the consequences of its actions, rather than from being explicitly taught and it selects it actions on basis of its past experiences (exploitation) and also by new choices (exploration), which is essentially trial and error learning. The reinforcement signal that the RL-agent receives is a numerical reward, which encodes the success of an action's outcome, and the agent seeks to learn to select actions that maximize the accumulated reward over time. (The use of the term reward is used here in a neutral fashion and does not imply any pleasure, hedonic impact or other psychological interpretations.)

**2 b) Consider an e-mail SPAM classification for every incoming e-mail is either a normal email (ω1) or junk mail (ω2). We have two actions α1 keep the mail and α2 put mail to /dev/null. We select a loss function λ(α1|ω1) = 0, λ(α1|ω2) = 1, λ(α2|ω1) = 3 and λ(α2|ω2) = 0. In addition, we know that P(ω1) = 0.7, P(ω2) = 0.3. Now, an e-mail has been received and its feature vector is x. Based on the feature vector, we have computed p(x|ω1) = 0.65, p(x|ω2) = 0.35. Find the posterior probabilities of P(ω1|x), P(ω2|x) and conditional risk or minimum risk classifier R(α1|x) and R(α2|x).**

**Sol:** A loss function

λ(α1|ω1) = 0, λ(α1|ω2) = 1,

λ(α2|ω1) = 3 and λ(α2|ω2) = 0

Prior probabilities P(ω1) = 0.7, P(ω2) = 0.3 and compound probabilitiesp(x|ω1) = 0.65, p(x|ω2) = 0.35.

The posterior probabilities are:

P(ω1|x) = (0.35 \* 0.7) / (0.35 \* 0.7 + 0.65 \* 0.3) = 0.556

P(ω2|x) = (0.65 \* 0.3) / (0.35 \* 0.7 + 0.65 \* 0.3) = 0.443

For the minimum risk classifier, we must still compute the conditional losses:

**R(α1|x) =** 0 \* 0.556 + 1 \* 0.556 = 0.556

**R(α2|x) =** 3 \* 0.443 + 0 \* 0.556 =1.329

The Bayes (minimum error) classifier classifies the e-mail as spam but the minimum risk classifier still keeps it in the inbox because of the smaller loss of that action.

**3 a) Discuss the univariate density function with the continuous normal or Gaussian density case.**

Sol: The structure of a Bayes classifier is determined by the conditional densities *p*(**x***|ωi*) as well as by the prior probabilities. Of the various density functions that have been investigated, none has received more attention than the multivariate normal or Gaussian density.

We begin with the continuous univariate normal or Gaussian density,



for which the *expected value* of *x* (an average, here taken over the feature space) is



and where the expected squared deviation or *variance* is



The univariate normal density is completely specified by two parameters: its mean *μ* and variance *σ*2. writing *p*(*x*) *∼ N*(*μ, σ*2) to say that *x* is distributed normally with mean *μ* and variance *σ*2.

**3 b) Assume the feature vector x cloud be any point in a d-dimensional Euclidean spar Rd. Discuss the Bayes Decision theorem for discrete features.**

Sol: we have assumed that the feature vector **x** could be any point in a *d-*dimensional Euclidean space, **R***d*. However, in many practical applications the components of **x** are binary-, ternary-, or higher integer valued, so that **x** can assume only one of *m* discrete values **v**1*, ...,* **v***m*. In such cases, the probability density function *p*(**x***|ωj*) becomes singular; integrals of the form



must then be replaced by corresponding sums, such as



where we understand that the summation is over all values of **x** in the discrete distribution.Bayes’ formula then involves probabilities, rather than probability densities:



where



The definition of the conditional risk *R*(*α|***x**) is unchanged, and the fundamental Bayes decision rule remains the same: To minimize the overall risk, select the action *αi* for which *R*(*αi|***x**) is minimum, or stated formally,

*α∗* = arg maxi *R*(*αi|***x**)*.*

The basic rule to minimize the error-rate by maximizing the posterior probability is also unchanged as are the discriminant functions.

**4 a) Let us assume a binary feature vector x = (x1, x2, .., xd)t, where the components xi are either 0 or 1 as independent binary features, discuss in detail.**

Sol: As an example of a classification involving discrete features, consider the two-category problem in which the components of the feature vector are binary-valued and conditionally independent. To be more specific we let **x** = (*x*1*, ..., xd*)*t*, where the components *xi* are either 0 or 1, with



This is a model of a classification problem in which each feature gives us a yes/no answer about the pattern. If pi > qi, we expect the ith feature to give a “yes” answer more frequently when the state of nature is ω1 than when when it is ω2. (As an example, consider two factories each making the same automobile, each of whose d components could be functional or defective. If it was known how the factories differed in their reliabilities for making each component, then this model could be used to judge which factory manufactured a given automobile based on the knowledge of which features are functional and which defective.) By assuming conditional independence we can write P(x|ωi) as the product of the probabilities for the components of x. Given this assumption, a particularly convenient way of writing the class-conditional probabilities is as follows:



Then the likelihood ratio is given by


We note especially that this discriminant function is linear in the *xi* and thus we can write



Let us examine these results to see what insight they can give. Recall first that we decide *ω*1 if *g*(**x**) *>* 0 and *ω*2 if *g*(**x**) *≤* 0. We have seen that *g*(**x**) is a weighted combination of the components of **x**.

**4 b) Let's assume a two dimensional binary feature vector X= (x1, x2) = (1, 0) that will attempt to classify either class1 or class2 and assume the prior probability of class1 is P(ω1) = 0.4 and P(ω2) = 0.6 with known probability or likelihood of each independent feature is given each class: p={0.7, 0.5} and q={0.2, 0.6}. Determine the discriminant function g(x) to create hyperplane.**

Sol:

Let us say that the prior probability for Class1 is P(ω1) = 0.4 and Class1 is P(ω2) = 0.6. Hence it is evident that there is a bias towards Class2.

Additionally, we know that likelihoods of each independent feature is given by p and q where:

pi = P(xi=1|ω1) and qi = P(xi=1|ω2)

meaning that we know the probability (or likelihood) of each independent feature given each class - these values are known and given:

p={0.7, 0.5} and q={0.2, 0.6}

therefore, the discriminant function is g(x) = g1(x) - g2(x) or by taking the log of both sides:



however, since the problem definition assumes that ***X***is independent, the discriminant function can be calculated by:



with







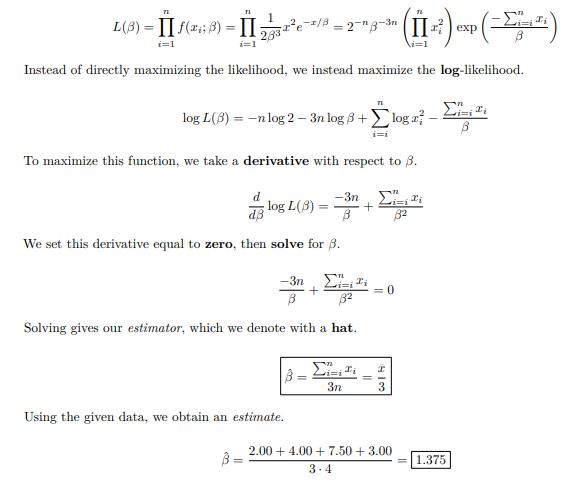


Input X = (x1, x2) = (1, 0) then g(X) = 2.23 – 1.163 = 1.037

After inputting the *xi* values into the discriminant function, the answer *g*(**x**) = 1.037. Therefore, this belongs to Class 1.  All points belong to class ω1 since if X = (1, 0), g(x) = 1.037 > 0.

**5 a) Let X1, X2,...,Xn be a random sample of size n from a distribution with probability density function , x > 0, β > 0. Find the maximum likelihood estimator of β, . Calculate the estimator when x1 = 5, x2 = 4, x3 = 5, x4 = 7.**

Sol: We first obtain the likelihood by multiplying the probability density function for each Xi. We then simplify this expression.

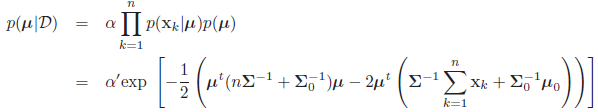


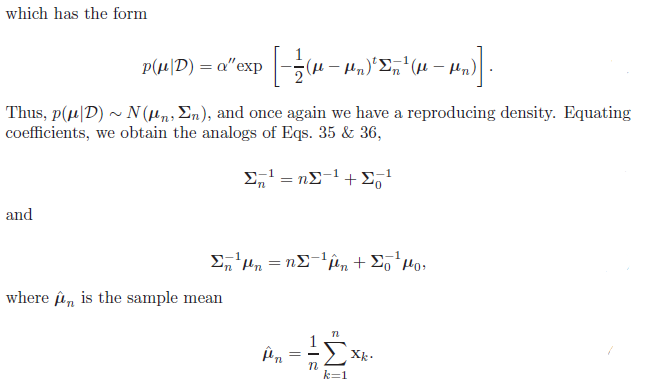
**5 b) Discuss the Bayesian Parameter Estimation of Multivariate Gaussian case.**

**Sol:** The treatment of the multivariate case in which **Σ** is known but ***μ*** is not, is a direct generalization of the univariate case. For this reason we shall only sketch the derivation. As before, we assume that

*p*(x*|μ*) *∼ N*(*μ,*Σ) and *p*(*μ*) *∼ N*(*μ*0*,*Σ0)

where Σ, Σ0, and μ0 are assumed to be known. After observing a set D of n independent samples x1,..., xn, we use Bayes’ formula to obtain



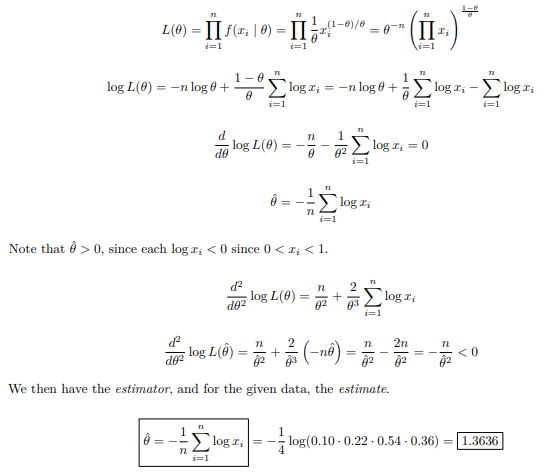


**6 a) Let X1, X2,...,Xn be a random sample from a population with probability density function (pdf).**

**, *0 < x < 1, 0 < θ < ∞.***

**Find the maximum likelihood estimator of θ, call it . Calculate an estimator using this estimator when x1 = 0.20, x2 = .44, x3 = 0.54, x4 = 0.22.**

**Sol:**



**6 b) Discuss the basic assumptions about Bayesian Parameter Estimation.**

**Sol:** We have just seen how the Bayesian approach can be used to obtain the desired density *p*(**x***|D*) in a special case — the multivariate Gaussian. This approach can be generalized to apply to any situation in which the unknown density can be parameterized. The basic assumptions are summarized as follows:

* The form of the density *p*(**x***|****θ***) is assumed to be known, but the value of the parameter vector ***θ*** is not known exactly.
* Our initial knowledge about ***θ*** is assumed to be contained in a known a priori density *p*(***θ***).
* The rest of our knowledge about ***θ*** is contained in a set *D* of *n* samples **x**1*, ...,* **x***n* drawn independently according to the unknown probability density *p*(**x**).

The basic problem is to compute the posterior density *p*(***θ****|D*),



**7 a) Suppose now that we are given a set D = {x1,...,xn} of n unlabeled samples drawn independently from the mixture density , where the full parameter vector θ is fixed but unknown. The likelihood of the observed samples is, by definition, the joint density Calculate the Maximum likelihood estimator .**

**Sol:** Suppose now that we are given a set *D* = *{***x**1*, . . . ,* **x***n}* of *n* unlabeled samples drawn independently from the mixture density

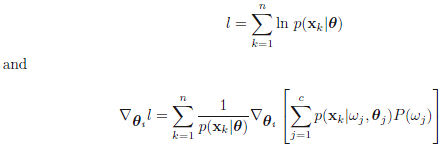
****

where the full parameter vector ***θ*** is fixed but unknown. The likelihood of the observed samples is, by definition, the joint density

****

The maximum-likelihood estimate ˆ***θ*** is that value of ***θ*** that maximizes *p*(*D|****θ***).

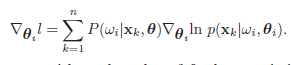
If we assume that *p*(*D|****θ***) is a differentiable function of ***θ***, then we can derive some interesting necessary conditions for ˆ***θ*** . Let *l* be the logarithm of the likelihood, and let *∇* ***θ****i l* be the gradient of *l* with respect to ***θ****i*. Then



If we assume that the elements of ***θ****i* and ***θ****j* are functionally independent if *i ≠* *j*, and if we introduce the posterior probability



we see that the gradient of the log-likelihood can be written in the interesting form



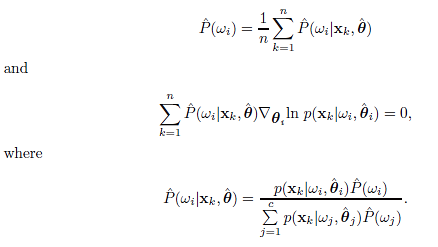
Since the gradient must vanish at the value of ***θ****i* that maximizes *l*, the maximum likelihood estimate ˆ***θ****i* must satisfy the conditions



Among the solutions to these equations for ˆ***θ****i* we may find the maximum-likelihood solution. It is not hard to generalize these results to include the prior probabilities *P*(*ωi*) among the unknown quantities. In this case the search for the maximum value of *p*(*D|****θ***) extends over ***θ*** and *P*(*ωi*), subject to the constraints

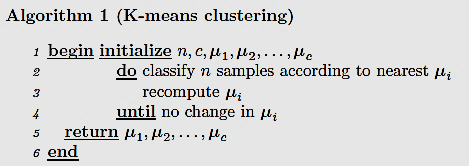
*P*(*ωi*) *≥* 0 *i* = 1*, . . . , c*





**7 b) Illustrate K-means Algorithm and give its computational complexity of the algorithm.**

**Sol:** Finding ˆμ1, . . . ,ˆμc.(Although the algorithm is historically referred to as k-means clustering, we retain the notation c, our symbol for the number of clusters.)



The computational complexity of the algorithm is O(ndcT) where d the number of features and T the number of iterations. In practice, the number of iterations is generally much less than the number of samples.

**8 a) Suppose that we have a set 'D' of 'n' samples x1, . . . , xn that we want to partition into exactly 'c' disjoint subsets D1, . . . ,Dc. Discuss the various scatter criteria functions for clustering.**

**Sol:**

**i. The scatter matrices:**

Another interesting class of criterion functions can be derived from the scatter matrices used in multiple discriminant analysis.

As before, it follows from these definitions that the total scatter matrix is the sum of the within-cluster scatter matrix and the between-cluster scatter matrix:

**S***T* = **S***W* + **S***B.*

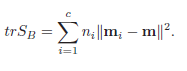
Note that the total scatter matrix does not depend on how the set of samples is partitioned into clusters; it depends only on the total set of samples. The within-cluster and between-cluster scatter matrices taken separately do depend on the partitioning, of course. Roughly speaking, there is an exchange between these two matrices, the between-cluster scatter going up as the within-cluster scatter goes down. This is fortunate, since by trying to minimize the within-cluster scatter we will also tend to maximize the between-cluster scatter. In the univariate case, these two measures are equivalent, and we can define an optimal partition as one that minimizes **S***W* or maximizes **S***B*.

**ii. The Trace Criterion**

Perhaps the simplest scalar measure of a scatter matrix is its trace — the sum of its diagonal elements. Roughly speaking, the trace measures the square of the scattering radius, since it is proportional to the sum of the variances in the coordinate directions. Thus, an obvious criterion function to minimize is the trace of **S***W*. In fact, this criterion is nothing more or less than the sum-of-squared-error criterion, since the definitions of scatter matrices yield



Since *tr***S***T* = *tr***S***W*+*tr***S***B* and *tr***S***T* is independent of how the samples are partitioned, we see that no new results are obtained by trying to maximize *tr***S***B*. However, it is comforting to know that in seeking to minimize the within-cluster criterion *Je* = *tr***S***W* we are also maximizing the between-cluster criterion



**iii. The Determinant Criterion**

we used the determinant of the scatter matrix to obtain a scalar measure of scatter. Roughly speaking, the determinant measures the square of the scattering volume, since it is proportional to the product of the variances in the directions of the principal axes. Since **S***B* will be singular if the number of clusters is less than or equal to the dimensionality, *|***S***B|* is obviously a poor choice for a criterion function. Furthermore, **S***B* may become singular, and will certainly be so if *n − c* is less than the dimensionality *d*. However, if we assume that **S***W* is nonsingular, we are led to consider the determinant criterion function



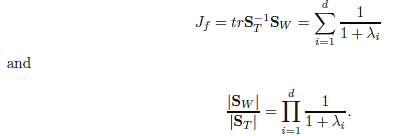
The partition that minimizes *Jd* is often similar to the one that minimizes *Je*, but the two need not be the same, as shown in Example 3. We observed before that the minimum-squared-error partition might change if the axes are scaled, though this does not happen with *Jd.*

**iv. Invariant Criteria**

It is not particularly hard to show that the eigenvalues *λ*1*, . . . , λd* of **Sw***−*1**S***B* are invariant under nonsingular linear transformations of the data. Indeed, these eigenvalues are the basic linear invariants of the scatter matrices. Their numerical values measure the ratio of between-cluster to within-cluster scatter in the direction of the eigenvectors, and partitions that yield large values are usually desirable. One can invent a great variety of invariant clustering criteria by composing appropriate functions of these eigenvalues. Some of these follow naturally from standard matrix operations. For example, since the trace of a matrix is the sum of its eigenvalues, one might elect to maximize the criterion function



By using the relation **S***T* = **S***W* +**S***B*, one can derive the following invariant relatives of [*tr***S***W* and *|***S***W|*].



**8 b) For a multivariate normal or Gaussian distribution p(x|ωi,θi)~N(μi,Σi). Determine the maximum likelihood estimation of μi when unknown mean vectors.**

**Sol:** If the only unknown quantities are the mean vectors μi, then of course θi consists of the components of μi.



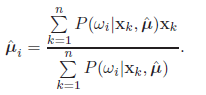
its derivative is



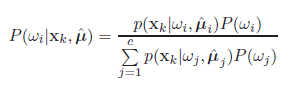
the maximum-likelihood estimate ˆμi must satisfy



After multiplying by Σi and rearranging terms, we obtain the solution:



This equation is intuitively very satisfying. It shows that the maximum-likelihood estimate for ***μ****i* is merely a weighted average of the samples; the weight for the *k*th sample is an estimate of how likely it is that **x***k* belongs to the *i*th class. If *P*(*ωi|***x***k,*ˆ***μ***) happened to be 1.0 for some of the samples and 0.0 for the rest, then ˆ***μ****i* would be the mean of those samples estimated to belong to the *i*th class. More generally, suppose that ˆ***μ****i* is sufficiently close to the true value of ***μ****i* that *P*(*ωi|***x***k,*ˆ***μ***) is essentially the true posterior probability for *ωi*. If we think of *P*(*ωi|***x***k,*ˆ***μ***) as the fraction of those samples having value **x***k* that come from the *i*th class, then we see that above equation essentially gives ˆ***μ****i* as the average of the samples coming from the *i*th class. Unfortunately, the above equation does not give ˆμi explicitly, and if we substitute



with p(x|ωi,ˆμi) ∼ N(ˆμi,Σi), we obtain a tangled snarl of coupled simultaneous nonlinear equations. These equations usually do not have a unique solution, and we must test the solutions we get to find the one that maximizes the likelihood.

**9 a) List and define the three main problems of Hidden Markov model computation.**

**Sol:** With these preliminaries behind us, we can now focus on the three central issues in hidden Markov models:

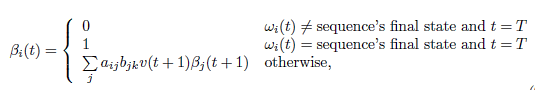
1. The Evaluation problem. Suppose we have an HMM, complete with transition probabilities aij and bjk. Determine the probability that a particular sequence of visible states VT was generated by that model.
2. The Decoding problem. Suppose we have an HMM as well as a set of observations VT . Determine the most likely sequence of hidden states ωT that led to those observations.
3. The Learning problem. Suppose we are given the coarse structure of a model (the number of states and the number of visible states) but not the probabilities aij and bjk. Given a set of training observations of visible symbols, determine these parameters.

**9 b) The main goal of HMM learning is to determine model parameters - the transition probabilities aij and bjk — from an ensemble of training samples. Describe the approximation algorithm to estimate the model parameters from the training dataset.**

**Sol:**

**The Forward-backward Algorithm**

The Forward-backward algorithm is an instance of a generalized Expectation-Maximization algorithm. The general approach will be to iteratively update the weights in order to better explain the observed training sequences. Above, we defined *αi*(*t*) as the probability that the model is in state *ωi*(*t*) and has generated the target sequence up to step *t*. We can analogously define *βi*(*t*) to be the probability that the model is in state *ωi*(*t*) and *will generate* the remainder of the given target sequence, i.e., from *t* + 1 *→ T*. We express *βi*(*t*) as:

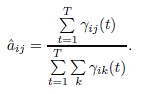


imagine we knew *αi*(*t*) up to step *T −*1, and we wanted to calculate the probability that the model would generate the remaining single visible symbol. This probability, *βi*(*T*), is just the probability we make a transition to state *ωi*(*T*) multiplied by the probability that this hidden state emitted the correct final visible

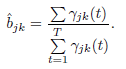
symbol. By the definition of *βi*(*T*) in Eq. 94, this will be either 0 (if *ωi*(*T*) is not the final hidden state) or 1 (if it is). Thus it is clear that *βi*(*T −*1) = ∑*j aij bij v*(*T*) *βi*(*T*). Now that we have determined *βi*(*T −* 1), we can repeat the process, to determine *βi*(*T −* 2), and so on, *backward* through the trellis. We can calculate an improved value by first defining *γij*(*t*) — the probability of transition between *ωi*(*t−*1) and *ωj*(*t*), given the model generated the entire training sequence **V***T* by *any* path. We do this by defining *γij*(*t*), as follows:

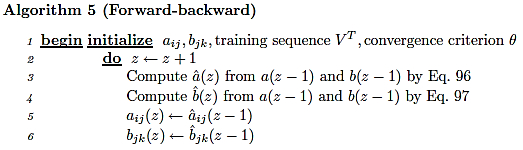


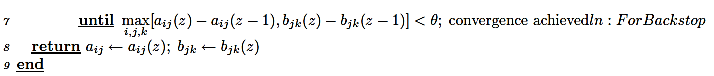
We can now calculate an improved estimate for *aij* . The expected number of transitions between state *ωi* (*t −*1) and *ωj*(*t*) at *any* time in the sequence



In the same way, we can obtain an improved estimate*ˆbij* by calculating the ratio between the frequency that any particular symbol *vk* is emitted and that for any symbol. Thus we have





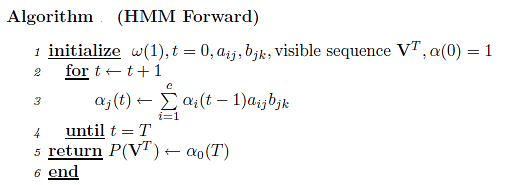


**10 a) Suppose we have an HMM with N sates and M visible state, and a particular sequence of visible states VT is observed, where T is the length of the sequence. Describe a smart algorithm to compute the probability that VT was generated by this model, i.e. using HMM forward and HMM backward algorithm. you need to give the definition of αj(t) or βj(t) and explain the meaning of it in words.**

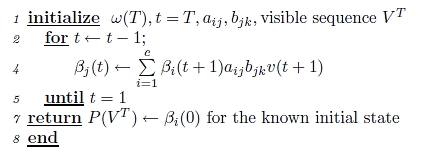
**Sol:** To compute P(VT) recursively, we use HMM Forward algorithm by defining



where aij and bjk are the transition and emission probabilities. Since αj (t) is the probability that the HMM is in hidden state ωi at step t having generated the first t elements of VT, therefore P(VT) can be represented by α0(T) at the final state (i.e. the absorber state with an unique null visible symbol v0).

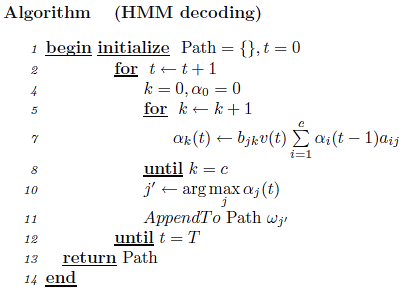






**10 b) Suppose we have an HMM with N sates and M visible state, and a particular sequence of visible states VT is observed, where T is the length of the sequence. Describe the decoding problem algorithm to find the most probable sequence of hidden states.**

**Sol:** Given a sequence of visible states VT, the decoding problem is to find the most probable sequence of hidden states. While we might consider enumerating every possible path and calculating the probability of the visible sequence observed, this is an O(cTT) calculation and prohibitive. Instead, we use perhaps the simplest decoding algorithm:



A closely related algorithm uses logarithms of the probabilities and calculates total probabilities by addition of such logarithms; this method has complexity O(c2T).

