Unit 4

Syllabus: Sequential Data - Markov Models, HMM - Maximum likelihood for the HMM, The forward & Backward algorithm, the sum product algorithm, scaling factors, Viterbi algo-, linear dynamical system.

Sequential data - Sequential data machine learning is a type of ML that uses models to process data that is ordered into sequences, & where the order of data point is important.

Example of Sequential Data - Textual data, Time series data, video streams, audio signals etc.

Tasks that can be modelled using sequential data -

- Text classification
- Language Translation
- Sentiment classification
- Time series forecasting ie stock price prediction.

* RNNs are well-known method in sequence models.

Markov Model

Markov models are also called Markov chains or markov processes.

These are probabilistic models that capture the dynamics & dependencies of a sequence of events.

Use of Markov model -

To generate sequence of data such as text, music or time series

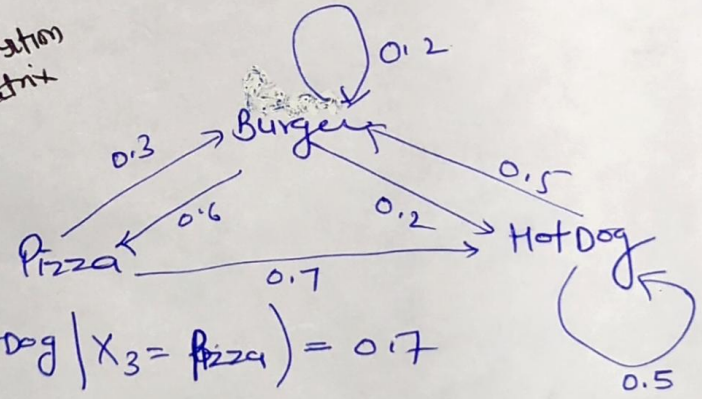
Markov Properties- ① future state depends on current state only not on the previous states

$$P(X_{n+1} = x \mid x_n = x_n)$$

② Sum of the weights of outgoing edge from any state is equal to 1

	B	P	H
B	0.2	0.6	0.2
P	0.3	0	0.7
H	0.5	0	0.5

$= A$ Transition Matrix



$$P(X_4 = \text{HotDog} \mid X_3 = \text{Pizza}) = 0.7$$

Let 1st day one person ate Pizza

$$\pi_0 = [0 \quad 1 \quad 0]$$

$$\pi_0 A = [0 \quad 1 \quad 0] \begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.3 & 0 & 0.7 \\ 0.5 & 0 & 0.5 \end{bmatrix} = [0.3 \quad 0 \quad 0.7]$$

future probability of the pizza state

P to $B = 0.3$ | P to $P = 0$ | P to $H = 0.7$

$$\pi_1 A = [0.3 \quad 0 \quad 0.7] \begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.3 & 0 & 0.7 \\ 0.5 & 0 & 0.5 \end{bmatrix} = [0.41 \quad 0.18 \quad 0.41]$$

Perform same process until we get equilibrium or stationary state ie output row vector = input row vector

ie $\boxed{\pi A = \pi}$ ← Eign vector eqn

$$Av = \lambda v$$

$$\pi = [0.35211 \quad 0.21127 \quad 0.43662]$$

Note:

- In markov model, a sequence of events are represented by a series of states, and the transition b/w states are governed by probabilities
- Markov model is represented as directed graph.

Order of markov model- No of previous states considered when predicting the next state.

Ist order markov model is also called markov chains.

Training markov model- Estimating the transition probabilities from a given dataset.

Given a sequence of events, the model calculates the probabilities of transitioning from one state to another based on the observed frequency in the data.

Forward Algorithm for HMM

Hidden Markov Model

A HMM is a statistical model that can be used to describe the evolution of observable events that depends on internal factors, which are not directly observable.

An HMM consists of two stochastic processes, an invisible process of hidden state & a visible process of hiddden observable symbols.

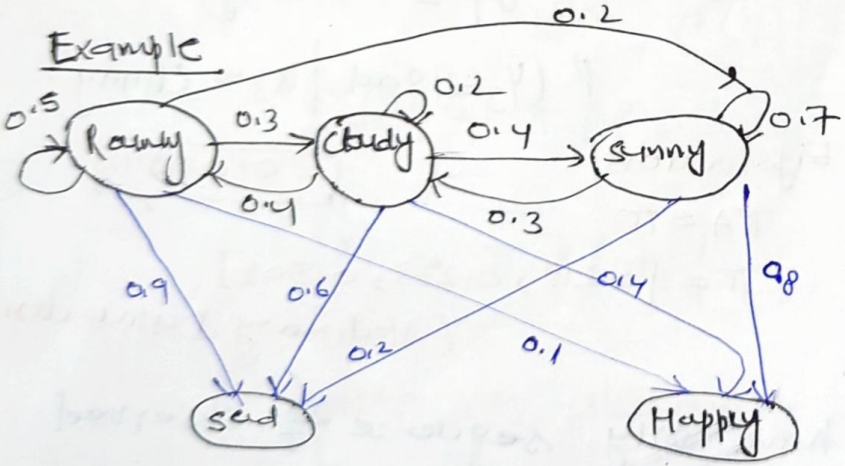
HMM assumes Markov property

Application of HMM

Reinforcement learning

Temporal Pattern Recognition ie Speech handwriting

Example



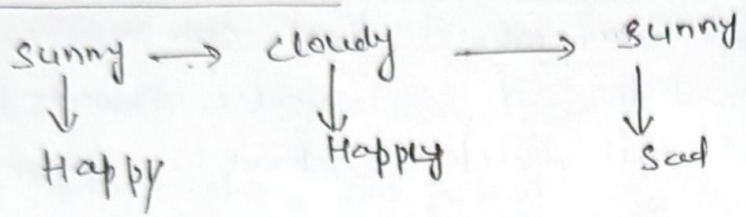
	R	C	S
R	0.5	0.3	0.2
C	0.4	0.2	0.4
S	0.0	0.3	0.7

Transition matrix

	Sad	Happy
R	0.9	0.1
C	0.6	0.4
S	0.2	0.8

Emmm Matrix

Let us consider



WHAT is the probability of occurrence of above scenario?

or

$P(Y = \text{Happy Happy Sad}, X = \text{Sunny cloudy, sunny})$?

Calculation can be performed as follows

$P(X_1 = \text{sunny})$ 0.509 $P(Y_1 = \text{Happy} \mid X_1 = \text{sunny})$ 0.8

$P(X_2 = \text{Cloudy} \mid X_1 = \text{sunny})$ 0.3 $P(Y_2 = \text{Happy} \mid X_2 = \text{cloudy})$ 0.4

$P(X_3 = \text{sunny} \mid X_2 = \text{cloudy})$ 0.4

$P(Y_3 = \text{sad} \mid X_3 = \text{sunny})$ 0.2

$= 0.00391$

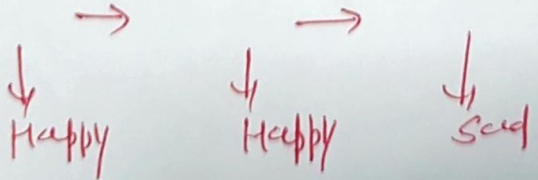
Using left Eigen vectors

$\pi A = \pi$

$\pi = [0.218, 0.273, 0.509]$

(Stationary Distribution)

But if we have only sequence of observed variables then what is the most likely weather sequence for the observed mood sequence given below?



Solution To find most likely sequence we have to find/calculate the probability for each of the possible weather sequences to find the sequence with MAX probability.

$$P(Y = \text{Happy Happy sad}, X = \text{sunny sunny cloudy})$$

$$= 0.04105$$

Y = Observed variable

X = Markov chain

$$\underset{X = x_1 x_2 \cdots x_n}{\arg\max} P(X = x_1, x_2, x_3, \cdots x_n \mid Y = y_1, y_2 \cdots y_n)$$

using Bayes theorem

$$\underset{X = x_1, x_2 \cdots x_n}{\arg\max} \frac{P(Y/X)\, P(X)}{P(Y)} \quad \text{---(1)}$$

$P(Y)$ — ignore this

$$P(Y/X) = \pi P(Y_i \mid X_i)$$

$$P(X) = \pi P(X_i \mid X_{i-1})$$

for first term use stationary distribution vector

∴ Eqn 1 can be written as —

$$\underset{X = x_1, x_2 \cdots x_n}{\arg\max} \prod P(Y_i \mid X_i^o)\, P(X_i^o \mid X_{i-1}^o)$$

Forward Algorithm for HMM

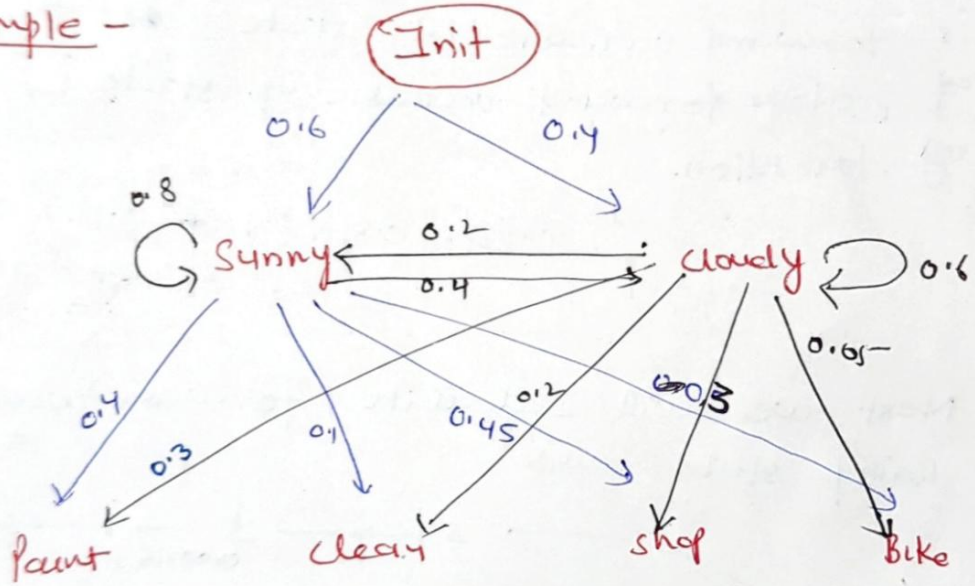
Forward Algorithm comprises of three steps –

1. Initialization $\alpha_1(i) = \pi_i \cdot b_i(O_i)$
2. Recursion
3. Termination

Recursion $\alpha_{t+1}(j) = \sum_{i=1}^{N} \alpha_t(i) \cdot a_{ij} b_j(O_{t+1})$

Termination $P(O/\lambda) = \sum_{i=1}^{N} \alpha_T(i)$

Example –



Hidden states – Sunny & Rainy
Observable states – Paint, clean, Shop & Bike.
Initial probability = [0.6, 0.4]

$$A = \begin{bmatrix} a_{11} & a_{12} \\ 0.8 & 0.2 \\ 0.4 & 0.6 \\ a_{21} & a_{22} \end{bmatrix}$$

Transition Probability

Emission Probability

	P	C	S	B
S	0.4	0.1	0.2	0.3
C	0.3	0.45	0.2	0.05

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \end{bmatrix}$$

Question – what is the likelihood that observation sequence $O = \{O_{paint}, O_{clean}, O_{shop}, O_{bike}\}$ can be derive from HMM λ? $P(O/\lambda) = ???$

Forward Algorithm for HMM

Initialization

Initial probability of sunny = 0.6

" " of sunny to paint = 0.4

∴ Initial forward variable for sunny = 0.24

" " Rainy state = 0.4 × 0.3 = 0.12

Recursion

$$\alpha_{t+1}(j) = \sum_{i=1}^{N} \alpha_t(i) \cdot a_{ij} \cdot b_j(O_{t+1})$$

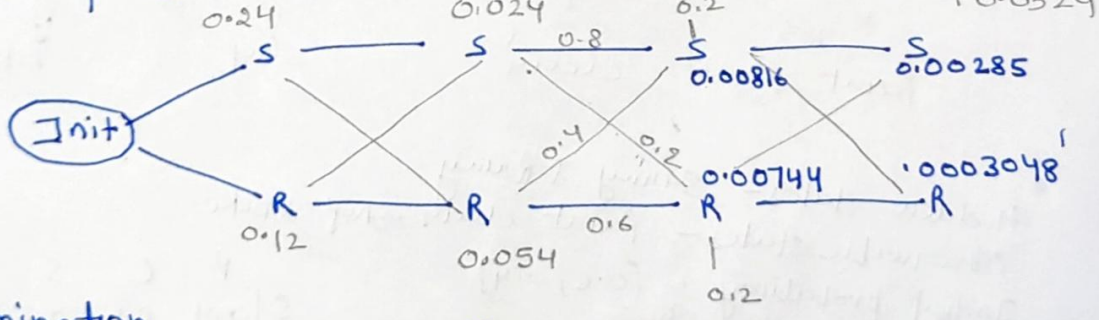
ie forward variable of state j is the product of previous forward variable of state i, multiplied by transition.

$0.24 \times 0.8 \times 0.1 + 0.12 \times 0.4 \times 0.1$

$= 0.192 + 0.0048$

$= 0.024$

Next we will calculate forward variable for Rainy state. $0.24 \times 0.2 \times 0.45 + 0.12 \times 0.6 \times 0.45 = 0.0216 + 0.0324$



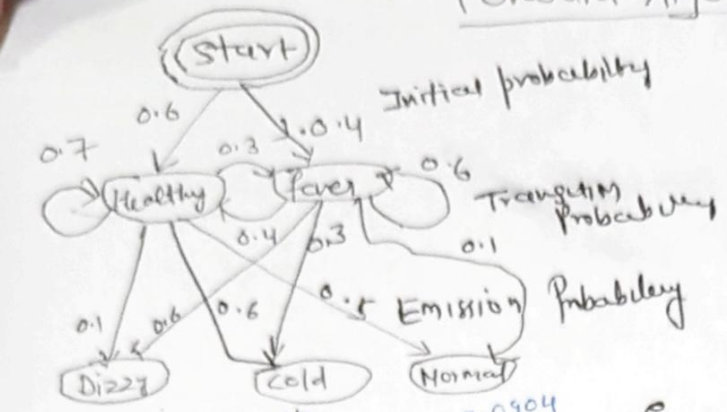
Termination

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

$= 0.00028512 + 0.0003048$

$= 0.003156$

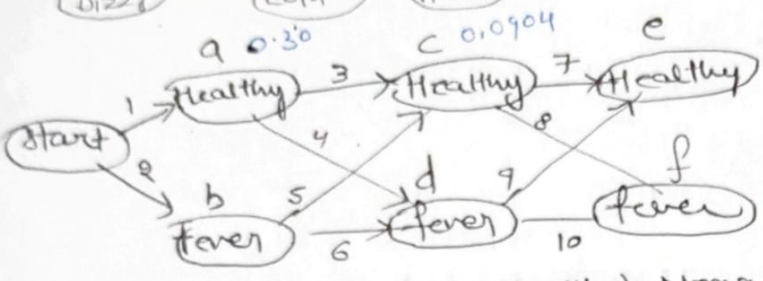
Forward Algorithm for HMM



	D	C	N
H	.1	.4	.5
F	.6	.3	.1

	H	F
H	.7	.3
F	.4	.6

Observed sequence = Normal Cold Dizzy



① ⇒ 0.6 × 0.5 = .3 (Start → Healthy → Normal) (Node 1)

② ⇒ 0.4 × 0.1 = .04 (Start → Fever → Normal (Node B)

NODE C

.3 × 0.7 × .4 = 0.084 (Start → Healthy Healthy → Cold)

Node .04 × .4 × .4 = 0.0064 + = 0.0904

Node D

.3 × .3 × .3 = 0.027 + = 0.03342

.04 × .6 × 0.3 = 0.0072

Node E

0.904 × 0.7 × 0.1 = 0.006328

0.0342 × 0.4 × 0.1 = 0.001368 + = 0.007696

Node F

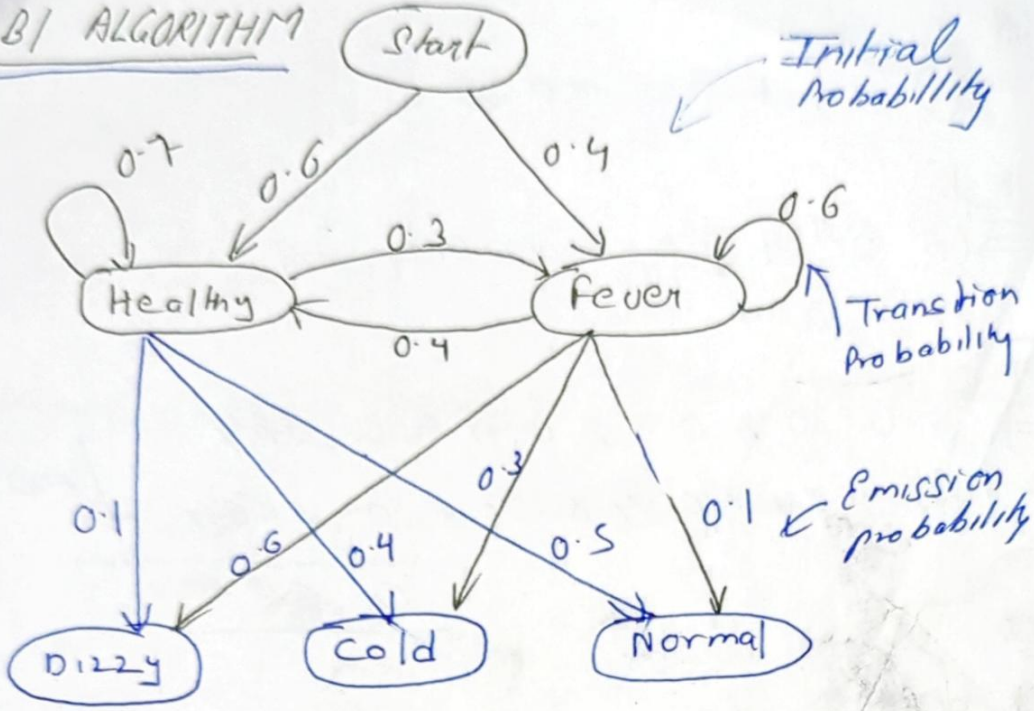
0.0904 × 0.3 × 0.6 = 0.016272

0.0342 × 0.6 × 0.6 = 0.012312 + ⇒ 0.028584

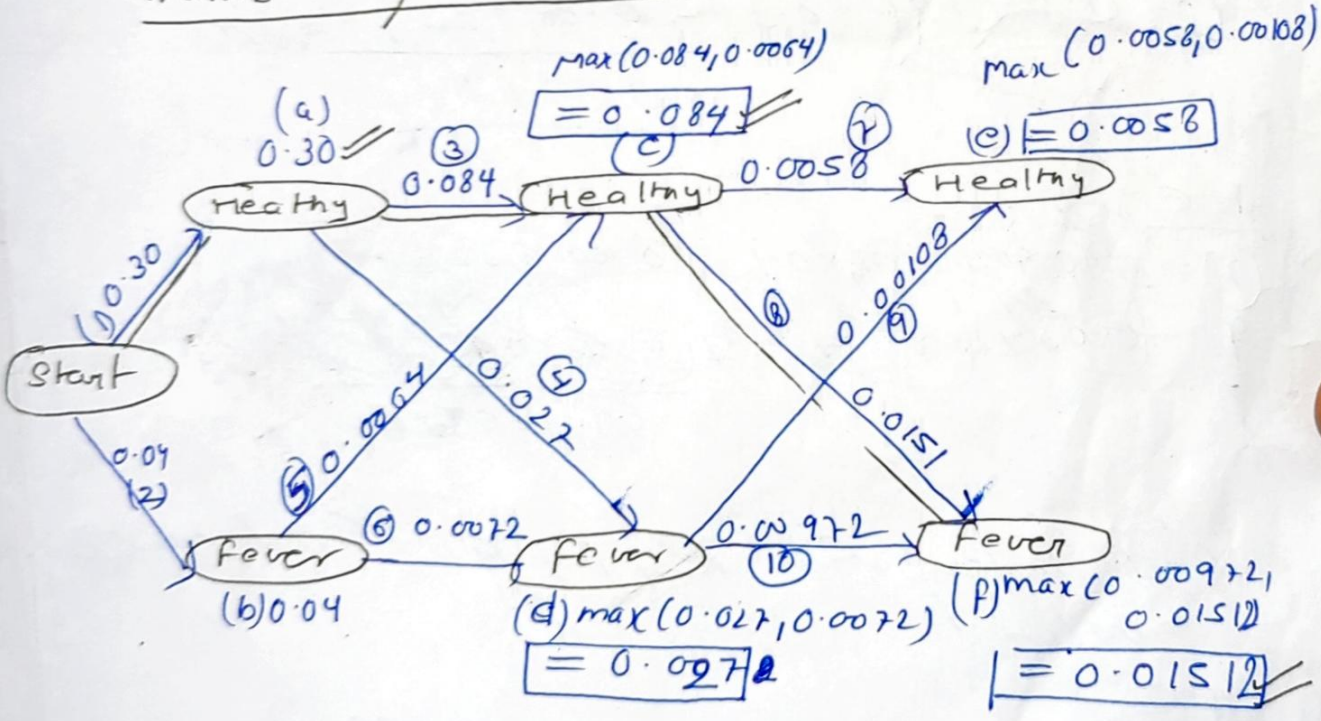
∴ for sequence Healthy Healthy Fever the observation Normal Cold ~~Dizzy~~ will be most suitable Observation.

∴ for Given observation H H F are most likely transition.

VITERBI ALGORITHM



Trellis Diagram (Time Based Digram



Normal → Cold → Dizzy

Healthy → Healthy → Fever

* a

→ (1) $0.6 \times 0.5 = 0.30$

* b

→ (2) $0.4 \times 0.1 = 0.04$

* c

→ (3) $0.30 \times 0.7 \times 0.4 = 0.084$

→ (5) $0.04 \times 0.4 \times 0.4 = 0.0064$] max

0.084

* d

→ (4) $0.30 \times 0.3 \times 0.3 = 0.027$]

→ (6) $0.04 \times 0.6 \times 0.3 = 0.0072$] max

0.027

* e

(7) $= 0.30 \times 0.7 \times 0.1 = 0.00 6328$] max

9 $= 0.027 \times 0.4 \times 0.1 = 0.00 1368$

$0.00 6328$

And similary for e, f, g find max value

* e

→ (7) $0.084 \times 0.7 \times 0.1 = 0.00588$ } max

→ (9) $0.027 \times 0.4 \times 0.1 = 0.00108$

0.00588

* f

→ (8) $0.84 \times 0.3 \times 0.6 = 0.1512$] max = 0.1512

→ (10) $0.027 \times 0.6 \times 0.6 = 0.0097$

$$A = \begin{array}{c} \\ B \\ P \\ H \end{array} \begin{array}{c} \begin{array}{ccc} B & P & H \end{array} \\ \begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.3 & 0 & 0.7 \\ 0.5 & 0 & 0.5 \end{bmatrix} \end{array}$$

(Transition Matrix)

π → denotes probability distribution of the states

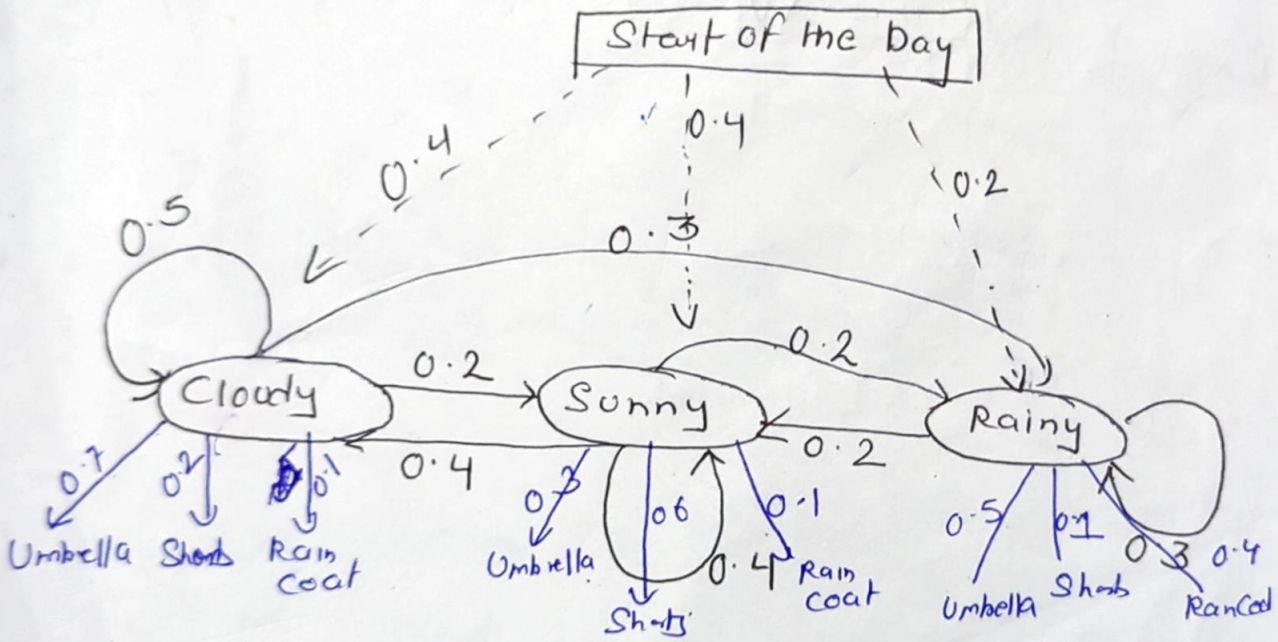
Suppose on beginning we are on Pizza day $\pi_0 = [0\ 1\ 0]$

$$\pi_0 A = [0\ 1\ 0] \begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.3 & 0 & 0.7 \\ 0.5 & 0 & 0.5 \end{bmatrix} = [0.34\ 0.25\ 0.41]$$

$$\pi A = \pi$$

$\boxed{Av = \lambda v}$ Eigen Vector Equation..

HMM



	Cloudy	Sonny	Rainy	
Cloudy	0.5	0.2	0.3	=1
Sonny	0.4	0.4	0.2	=1
Rainy	0.3	0.2	0.5	=1

(Transition Probability)

Backward HMM Algorithm

Question for Observation sequence

Paint (P) Clean (C) Shop (S) Bike (B)

Calculate Backward probability after applying Backward HMM algorithm.

$$A = \begin{array}{c|cc} & S & R \\ S & 0.8 & 0.2 \\ R & 0.4 & 0.6 \end{array}$$

Transition Matrix

$$B = \begin{array}{c|cccc} & P & C & S & B \\ S & 0.4 & 0.1 & 0.2 & 0.3 \\ R & 0.3 & 0.45 & 0.2 & 0.05 \end{array}$$

Emission Matrix

Initial Probability

P(Sunny) = 0.6

P(Rainy) = 0.4

β_1 β_2 β_3 β_4

S S S S

R R R R

Paint Clean Shop Bike

$\beta_3(S) = T(S \to S)\, B(\text{Bike} \mid \text{Sunny}) \times \beta_4(\text{Sunny}) + T(S \to R)\, B(\text{Bike} \mid \text{Rainy}) \times \beta_4(\text{Rainy})$

$\beta_2(S) = T(S \to S)\, B(S \mid S) \times \beta_3(\text{Sunny}) + T(S \to R) \times B(S \mid R) \times \beta_3(\text{Rainy})$

To understand

$\beta_3(\text{Sunny}) = A(S \to S) \times B(\text{Bike} \mid S) \times \beta_4(S) + A(S \to R) \times B(\text{Bike} \mid R) \times \beta_4(R)$

$\beta_3(\text{Rainy}) = A(R \to S)\, B(\text{Bike} \mid S) \times \beta_4(S) + A(R \to R)\, B(\text{Bike} \mid R) \times \beta_4(R)$

Backward Algorithm

Observation = { Opaint, Oclean, Oshop, Obike }

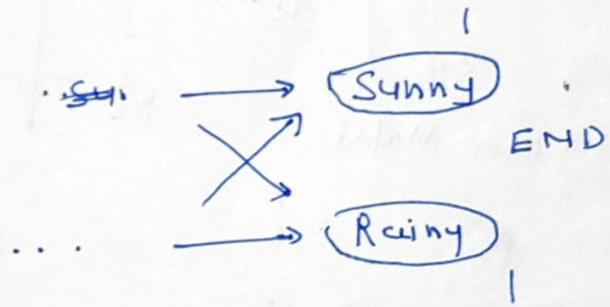
Backward algorithm runs backward in time

It also has 3 steps

1. Initialization
2. Recursion
3. Termination

Initialization

$\beta_T(i) = 1$



Calculate the backward variable of previous state sunny.

= transition probability from S to S times emission probability of bike from sunny state

$0.8 \times 0.3 \times 1 + 0.2 \times 0.05 \times 1 = 0.01$

$= 0.25$

Calculate backward variable fo previous state Rainy

$= 0.6 \times 0.05 \times 1 + 0.2 \times 0.05 \times 1 = 0.25$

kward Algorithm

It goes backward in time

Steps ① Initialization
② Recursion
③ Termination

Initialization

$\beta_T(i) = 1$ At time T (at the end of the observation sequence) the backward variable of every state is equal to 1

$\beta_4(\text{sunny}) = 1$

$\beta_4(\text{Rainy}) = 1$

Recursion

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} \cdot b_j(O_{t+1}) \cdot \beta_{t+1}(j)$$

a_{ij}: Transition probability

$b_j(O_{t+1})$: Emission probability

$\beta_{t+1}(j)$: Backward variable

For sunny at t = 3 :

$$\beta_3(\text{sunny}) = A(\text{sunny} \to \text{sunny}) \times B(\text{Bike}|\text{sunny}) \times \beta_4(\text{sunny}) + A(\text{sunny} \to \text{Rainy}) \times B(\text{Bike}|\text{Rainy}) \times \beta_4(\text{Rainy})$$

$$\beta_3(\text{sunny}) = 0.8 \times 0.3 \times 1 + 0.2 \times 0.05 \times 1 = 0.25$$

$$\beta_3(\text{Rainy}) = A(\text{Rainy} \to \text{sunny}) \times B(\text{Bike}|\text{sunny}) \times \beta_4(\text{sunny}) + A(\text{Rainy} \to \text{Rainy}) \times B(\text{Bike}|\text{Rainy}) \times \beta_4(\text{Rainy})$$

$$= 0.4 \times 0.3 \times 0.05 \times 1$$

$$0.4 \times 0.3 \times 1 + 0.6 \times 0.05 \times 1 = 0.15$$

$\beta_3(\text{Rainy}) = 0.15$

For sunny at $t=2$

$\beta_2(\text{sunny}) = A(\text{sunny} \to \text{sunny}) \times B(\text{shop}|\text{sunny}) \times \beta_3(\text{sunny})$

$A(\text{sunny} \to \text{Rainy}) \times B(\text{shop}|\text{Rainy}) \times \beta_3(\text{Rainy})$

$= 0.8 \times 0.2 \times 0.25 + 0.2 \times 0.2 \times 1.5 = 0.046$

for Rainy at $t=2$

$\beta_2(\text{Rainy}) = A(\text{Rainy}|\text{sunny}) \times B(\text{Shop}|\text{sunny}) \times \beta_3(\text{sunny})$

$A(\text{Rainy}|\text{Rainy}) \times B(\text{shop}|\text{Rainy}) \times \beta_3(\text{Rainy})$

$= 0.4 \times 0.2 \times 0.25 + 0.6 \times 0.2 \times 0.15$

$= 0.038$

for sunny at $t=1$

$\beta_1(\text{sunny}) = A(\text{sunny} \to \text{sunny}) \times B(\text{clean}|\text{sunny}) \times \beta_2(\text{sunny})$

$+ A(\text{sunny} \to \text{Rainy}) \times B(\text{clean}|\text{Rainy}) \times \beta_2(\text{Rainy})$

$= 0.8 \times 0.1 \times 0.046 + 0.2 \times 0.45 \times 0.038$

$= .0071$

for Rainy at $t=1$

$\beta_1(\text{Rainy}) = A(\text{Rainy} \to \text{sunny}) \times B(\text{clean}|\text{sunny}) \times \beta_2(\text{sunny})$

$+ A(\text{Rainy} \to \text{Rainy}) \times B(\text{clean}|\text{Rainy}) \times \beta_2(\text{Rainy})$

$= 0.4 \times 0.1 \times 0.046 + 0.6 \times 0.45 \times 0.038$

$= 0.0121$

Termination

$$P(O/\lambda) = \sum_{i=1}^{N} \pi_i \cdot b_i(O_1) \cdot \beta_1(i)$$

π_i = "initial probabilities of being in state (sunny or Rainy)

$b_i(O_1)$ is the emission probability of first observation O_1

β_1 is the backward probability at time $t=1$

$\pi(\text{sunny}) = 0.6$ $\pi(\text{Rainy}) = 0.4$

Emission probabilities for Paint

$B(\text{Paint}|\text{sunny}) = 0.4$ $B(\text{Paint}|\text{Rainy}) = 0.3$

$\beta_1(\text{sunny}) = 0.0071$ $\beta_1(\text{Rainy}) = 0.0121$

Considering sunny state :

$$P(O) = \pi(\text{sunny}) \times B(\text{Paint} \mid \text{sunny}) \times \beta_1(\text{sunny})$$

$$= 0.6 \times 0.4 \times 0.0071 = 0.001704$$

Considering Rainy state

$$P(O) = \pi(\text{Rainy}) \times B(\text{Paint} \mid \text{Rainy}) \times \beta_1(\text{Rainy})$$

$$= 0.4 \times 0.3 \times 0.0121 = 0.001452$$

Final value $= P(O \mid \lambda) = 0.001704 + 0.001452$

$$= 0.003156$$

1.SCALING

What is scaling

In Hidden Markov Models (HMMs), scaling is a technique used to prevent numerical underflow during the computation of probabilities in algorithms like the forward and backward algorithms.

Why Scaling is needed

When dealing with long sequences of observations, the probabilities calculated in HMMs can become extremely small, leading to numerical underflow (i.e., values so small that they can't be represented accurately by the computer). This happens because the probabilities are products of many small values, and multiplying many such small values together results in a very small number.

How Does Scaling Work?

To avoid this problem, scaling factors are introduced at each time step to normalize the probabilities. The scaling procedure typically involves the following steps:

1. Calculate the scaling factor: For each time step, compute a scaling factor that normalizes the forward or backward probabilities. This factor is usually the sum of the probabilities at that time step.
2. Scale the probabilities: Divide the probabilities at each time step by the scaling factor. This normalization step keeps the values within a manageable numerical range.
3. Store the scaling factors: Keep track of the scaling factors for each time step so that the original probabilities can be recovered if needed (e.g., for calculating the likelihood of the observed sequence).

4. Benefits of Scaling

1.Prevents numerical underflow: Keeps the values of probabilities from becoming too small to represent.

2.Improves numerical stability: Ensures that the calculations remain accurate even for long sequences.

Methods for scaling in data

Scaling in data preprocessing is essential in machine learning and statistical modelling to ensure that different features contribute equally to the learning process. Different methods of scaling are used based on the data type and algorithm. Here are some common scaling techniques:

1. Min-Max Scaling (Normalization)

- This technique transforms the data to a fixed range, typically [0, 1] or [-1, 1].
- The formula is: $X' = X - Xmin / Xmax - XminX'$

  Where X is original value and, Xmin and X max are minimum and maimum value of features.

- **Use Case:** Recommended when the data distribution does not follow a Gaussian distribution and when you need to bound the data values to a certain range (e.g., image pixel values).

2. Standardization (Z-score normalization)

- This method centers the data around the mean with a unit standard deviation.
- The formula is: $X' = X - \mu$

- Where X is the original value, μ is the mean of the feature, and σ is the standard deviation.
- **Use Case:** Commonly used in algorithms like Support Vector Machines, k-means clustering, and Principal Component Analysis (PCA) where the data distribution should be normalized to a mean of 0 and standard deviation of 1.

3. Max Abs Scaling

- This technique scales each feature by the maximum absolute value of the feature. The resulting values range between -1 and 1.
- The formula is: X′=X/max(| X |)
- **Use Case:** Useful when the data has a sparse representation and you want to preserve the sign of the original data.

4. Robust Scaling

- This method uses the median and the interquartile range (IQR) for scaling.
- The formula is: X′=X−median/IQR
- Where IQR is the difference between the 75th and 25th percentiles.
- **Use Case:** Suitable when the data contains outliers, as it is robust to the presence of extreme values.

5. Log Transformation

- It applies a logarithmic function to the data to reduce skewness.
- The formula is: X′=log (X+1)X' = \log(X + 1)X′=log(X+1)
- **Use Case:** Used when the data is highly skewed, as it helps to stabilize variance and make the data more normally distributed.

6.. Power Transformation (Box-Cox or Yeo-Johnson)

- A more advanced technique used to stabilize variance and make data more normal distribution-like.
- The transformation can be done using a power function (Box-Cox) or an optimized method (Yeo-Johnson).
- **Use Case:** Applied when the data needs to be transformed to approximate a normal distribution.

7.. Unit Vector Scaling (Normalization to unit norm)

- This scales the feature vector to have a unit norm (e.g., 1).
- The formula is: X′=X // X // X' = \frac{X}{\|X\|}X′= // X // X
- **Use Case:** Commonly used in text classification or clustering where you want to normalize the feature vectors' length.

Why is scaling important when training machine learning models?

Scaling is an important preprocessing step when training machine learning models because it ensures that features contribute equally to the learning process. Here are the key reasons why scaling is essential:

1. Prevents Features from Dominating Others

- When features have different scales, those with larger ranges may dominate the learning process and skew the results.

- For example, if one feature ranges from 0 to 1 (e.g., "age") and another ranges from 1,000 to 100,000 (e.g., "income"), the model might prioritize the feature with the larger range, even if it's not more important.
- Scaling puts all features on the same scale, allowing the model to treat them with equal importance.

2. Improves Convergence Speed in Gradient Descent

- Algorithms like Gradient Descent, which are used in many optimization-based machine learning algorithms (e.g., linear regression, neural networks), are sensitive to the scale of the features.
- When features are on different scales, the optimization process may take longer to converge because it has to navigate uneven gradients.
- Scaling helps to make the gradient smoother, thus speeding up the convergence.

3. Ensures Fair Distance Calculations

- Distance-based algorithms such as k-Nearest Neighbors (k-NN), Support Vector Machines (SVM), and clustering algorithms like k-means are highly sensitive to the scale of data.
- These algorithms rely on distance metrics like Euclidean distance, which will be affected if one feature has a much larger range than others.
- Scaling ensures that each feature contributes equally to the distance calculation, leading to better performance of these models.

4. Helps with Regularization Techniques

- Regularization techniques like Lasso (L1) and Ridge (L2) regression add a penalty term to the loss function based on the magnitude of feature coefficients.
- If features are on different scales, the penalty may disproportionately affect some coefficients, leading to suboptimal regularization.
- Scaling helps to ensure that the regularization term affects all features equally.

5. Avoids Numerical Instability

- Some algorithms are sensitive to the numerical ranges of the data, which can cause numerical instability during computation.
- For instance, neural networks may suffer from exploding or vanishing gradients when the input features have very large or very small values.
- Scaling features to a smaller range (e.g., [0, 1] or [-1, 1]) helps avoid these issues.

6. Improves Model Interpretability

- In some cases, having features on a common scale makes it easier to interpret the results.
- For instance, when visualizing decision boundaries in SVMs or clusters in k-means, scaled data provides a more meaningful representation.

2. Linear Dynamical System (LDS)

3. What is LDS

Linear Dynamical Systems (LDS) in machine learning are models used to represent systems that evolve over time in a linear manner. LDS is used to model time series data, where observations are made over time and there is an underlying latent state that evolves linearly with some noise. LDS is particularly useful in tasks like tracking, filtering, and predicting future states in a sequence.

Components of Linear Dynamical Systems

1. State Vector (xt):
   - Represents the hidden or latent state of the system at time ttt.
   - This vector captures the underlying factors that drive the evolution of the system over time.
2. Observation Vector (yt):
   - Represents the observed data at time ttt, which is a noisy measurement of the state.
   - In many cases, it is a linear transformation of the state vector with added noise.
3. State Transition Matrix (A):
   - Defines the linear relationship between the state vector at time ttt and the state vector at time t+1.
   - It determines how the system evolves from one state to the next.
4. Observation Matrix (C):
   - Maps the latent state vector to the observed data.
   - This matrix defines the relationship between the hidden states and the observations.
5. Process Noise (wt)
   - Represents random fluctuations in the state evolution.
   - It accounts for the uncertainty in the state transition.
6. Observation Noise (vt):
   - Represents the noise or error in the observations.
   - It accounts for the measurement uncertainty in the observed data.
7. Initial State Distribution (x0):
   - Specifies the distribution of the initial state of the system.
   - Usually assumed to be a Gaussian distribution with a mean and covariance.

Example of Linear Dynamical Systems

An example of LDS is tracking the position and velocity of an object moving in one dimension. In this case:

- The state vector (xt) could represent the position and velocity of the object.
- The observation vector (yt) could represent the observed position of the object, possibly with some measurement noise.
- The state transition matrix (A) would model how the position and velocity change over time, while the observation matrix (C) could map the state (position and velocity) to the observed position.

How can the state transition process in an HMM be modeled as a linear dynamical system?

The state transition process in a Hidden Markov Model (HMM) can be extended to be modeled as a Linear Dynamical System (LDS) by using continuous-valued hidden states and linear relationships for transitions and observations. Here's how this process works:

Key Differences and Modifications

1. **State Representation:**

- In an HMM, states are discrete, where each state represents a specific condition or class.
- In an LDS, the states are continuous-valued vectors, allowing the system to represent a range of values rather than discrete labels.

2. **State Transition Process:**

    - In an HMM, the state transition is governed by a transition probability matrix, where each element represents the probability of transitioning from one state to another.
    - In an LDS, the state transition is modeled as a linear equation