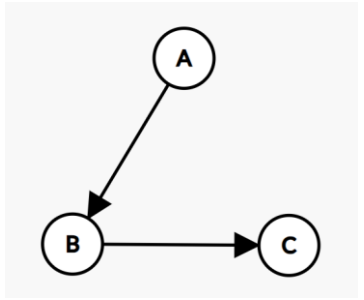
Probabilistic generative models

Probabilistic Graphical Models (Bayesian and Markovian)

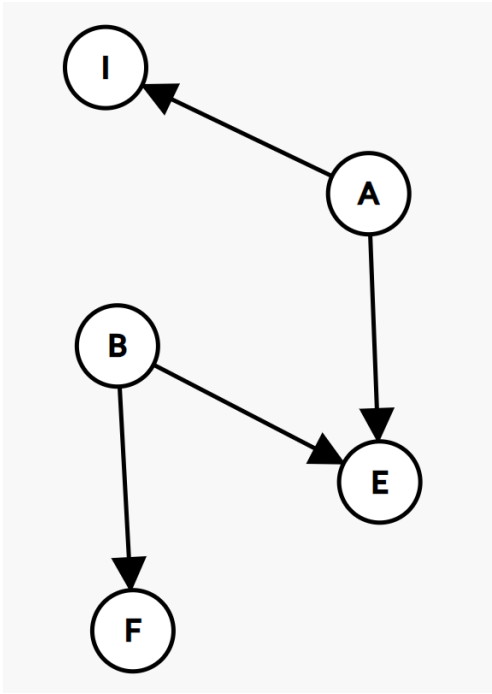
https://tinyurl.com/2s3jrhm2

Notions of Directed and Undirected Graphs – Directed Graphs

- A **directed graph** (or DAG: Directed Acyclic Graph) consists of nodes and directed edges between these nodes.
- Each **node** represents a random variable and each **directed edge** represents a conditional dependency between the variables.
- **Representation:**
  - Directional relationships are indicated by arrows.
  - Simple example: A influences B and B influences C.

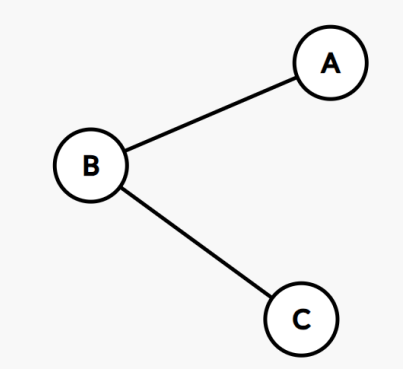


- **Example: Bayesian Networks:**
  - A Bayesian network is a directed graph where each node is annotated with conditional distributions that model dependencies between variables.
  - The parents of a node are the variables on which the corresponding variable of the current node directly depends.
  - Example: Here, (B) and (A) influence (E), (B) also influences (F), and (A) influences (I).



Notions of Directed and Undirected Graphs – Undirected Graphs

- An **undirected graph** consists of nodes and undirected edges.
- Each **node** represents a variable and each **edge** represents a direct dependency between the variables.
- **Representation:**
  - Relationships are indicated by lines without arrows.
  - Simple example: A is connected to B, and B is connected to C.



- **Example: Markov Random Fields:**
  - A Markov random field (or Markov Graph) is an undirected graph that models the conditional dependencies between variables.
  - The nodes represent the variables, and an edge between two nodes means they are conditionally dependent.

Notions of Directed and Undirected Graphs – Concept of Independence

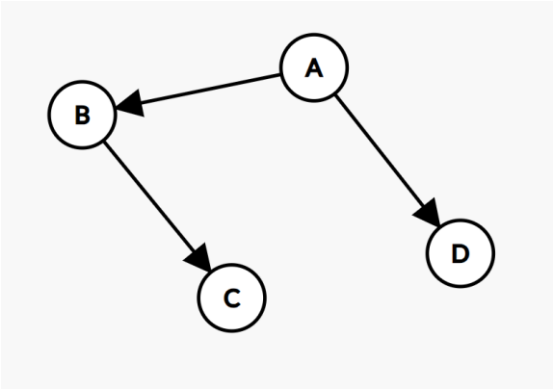
- **Independence:**
  - Two variables *X* and *Y* are independent if the probability of occurrence of *X* is not affected by *Y* and vice versa.

$$P(X, Y) = P(X)P(Y)$$

- **Conditional Independence:**
  - *X* is independent of *Y* given *Z* if, once *Z* is known, *X* and *Y* are independent.
  - Notation: $X \perp Y|Z$
  - Formula: $P(X, Y|Z) = P(X|Z)P(Y|Z)$
- **d-Separation:**
  - Two nodes *X* and *Y* are d-separated by a set of nodes *Z* if every path between *X* and *Y* is "blocked" by *Z*.
  - A path is blocked if at least one of the following conditions is met along the path:
    - There exists a chain $X \rightarrow Z \rightarrow Y$ or $X \leftarrow Z \leftarrow Y$ and *Z* is observed.
    - There exists a chain $X \leftarrow Z \rightarrow Y$ and *Z* is observed.
    - There exists a chain $X \rightarrow Z \leftarrow Y$ and neither *Z* nor any of its descendants are observed.

Bayesian Networks – Structure

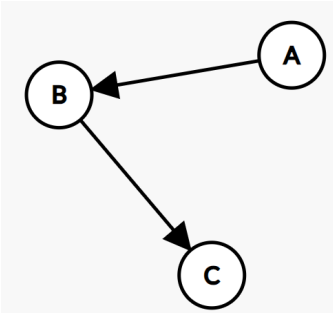
- **Nodes and Directed Edges, Directed Acyclic Graphs (DAGs):**
  - A **Bayesian network** is a graphical representation that uses **directed acyclic graphs** to model probabilistic dependencies among a set of random variables.
  - Each **node** represents a random variable, and each **directed edge** represents a conditional dependency between the variables.
  - **Acyclic** means there are no cycles in the graph, so one cannot return to the same node by following the edges.
- **Graphical Representation and Interpretation:**
  - Arrows indicate the direction of the conditional dependency.
  - **Parent nodes** are those from which arrows point to a given node, and **child nodes** are those which receive arrows from a given node.
  - Convention:$\mathrm{Parents}(X_i)$denotes the set of nodes having directed edges to *Xi*.
- **Example:**



- **Interpretation:**
  - Variable *A* influences *B* and *D*.
  - Variable *B* influences *C*.

Bayesian Networks – Factorization

- **Decomposition of the Joint Probability:**
  - The joint distribution of variables in a Bayesian network can be decomposed into a product of conditional distributions:
$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|\text{Parents}(X_i))$$
  - This simplification facilitates probability calculations using the conditional relationships represented in the graph.



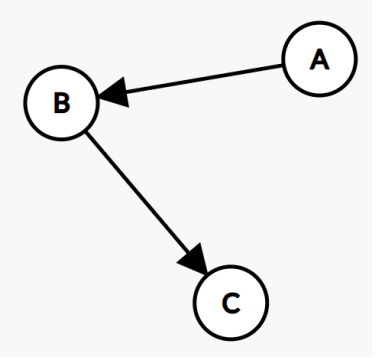
- **Practical Example of Factorization with a Simple Bayesian Model:**
  - Consider a simple model with three variables:
  - According to the conditional dependencies, the joint distribution decomposes as follows:
$$P(A, B, C) = P(A) \cdot P(B|A) \cdot P(C|B)$$
  - If we know the conditional distributions, calculating the joint distribution becomes more manageable.

Bayesian Networks – Exact and Approximate Inference

Exact Inference:

- **Variable Elimination Algorithms** is a method for computing exact marginals by successively eliminating variables that are outside the target set.
  - Example: To calculate *P(X)* from *P(X, Y, Z)*, eliminate *Y* and *Z*: $P(X) = \sum_{Y} \sum_{Z} P(X, Y, Z)$

- **Belief Propagation Algorithm** is an inference technique in graphs. It works by passing messages (beliefs) between nodes until convergence. It is effective for tree-structured graphs or acyclic networks.
  - Example: To find *P(A|C)*, beliefs are propagated from *C* to *A*.



Bayesian Networks – Exact and Approximate Inference

Approximate Inference:

- **Markov Chain Monte Carlo (MCMC)** methods are used to sample from complex distributions where exact inference is challenging.
- **Gibbs Sampling** is an MCMC technique where variables are sampled successively from their conditional distributions.
- **Gibbs Sampling Example:**

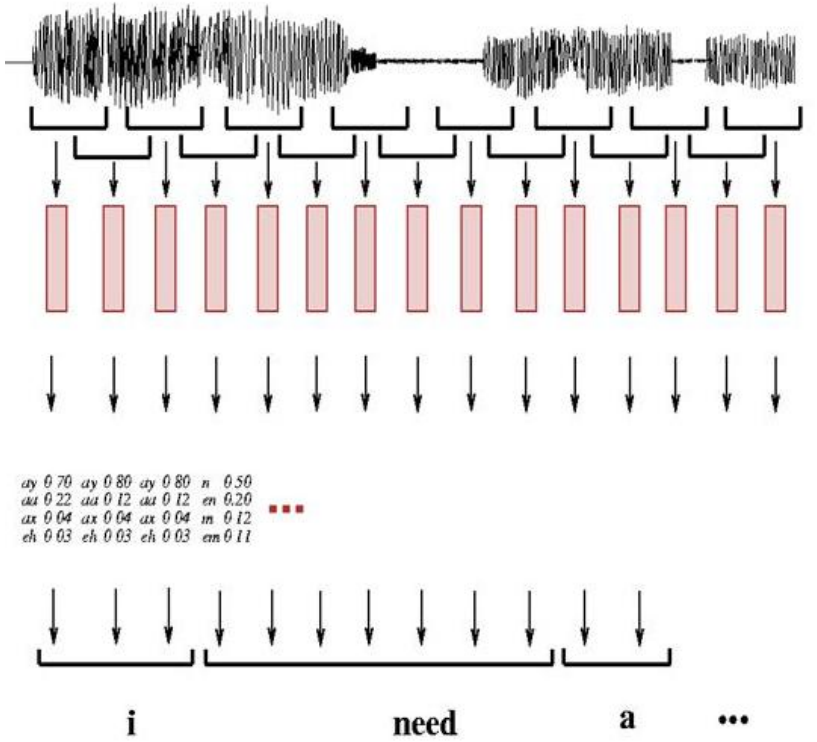
  Initialize *X* and *Y*. Sample $X \sim P(X|Y)$. Sample $Y \sim P(Y|X)$ Repeat until convergence.
- **Approximation Techniques for Complex Bayesian Networks:**
  - Use Variational Methods to approximate posteriors.
  - Importance Sampling to estimate expectations by sampling from an easier-to-handle distribution.
- **Practical Example:**
  - Approximate *P(X|Y)* in a complex network using samples generated by Gibbs Sampling.

Hidden Markov Models (HMMs) – Structure

Hidden Markov Models (HMMs) are a class of statistical models in which the system is assumed to be a Markov process with unobservable (hidden) states.

- **Hidden State S_t:** Variable representing the hidden state at time *t*. The states are finite, and their number is generally denoted by *N*.
- **Finite State Space:** The set of possible states that S_t can take.

  For example, $S_t \in S_1, S_2, \ldots, S_N$



Hidden Markov Models (HMMs) – Structure

- **Transitions between States:** Transitions between states are modeled by transition probabilities

$$A = [a_{ij}], where : a_{ij} = P(S_{t+1} = S_j | S_t = S_i)$$

- **Transition Matrix ((A)):** $A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix}$

- **Observation Probabilities:** The observations O_t are generated by the current hidden state according to an emission distribution characterized by matrix *B*.

- **Emission Matrix ((B)):**

$$B = \begin{bmatrix} b_1(O_1) & b_1(O_2) & \cdots & b_1(O_T) \\ b_2(O_1) & b_2(O_2) & \cdots & b_2(O_T) \\ \vdots & \vdots & \ddots & \vdots \\ b_N(O_1) & b_N(O_2) & \cdots & b_N(O_T) \end{bmatrix}$$

- $b_i(O_t) = P(O_t | S_t = S_i)$: Probability of observing O_t when the hidden state is S_i.

Hidden Markov Models (HMMs) – Assumptions

- **Markov Assumption:**

  The future state S_{t+1} depends only on the current state S_t and is independent of previous states $(S_1, S_2, \ldots, S_{t-1})$.

$$P(S_{t+1}|S_t, S_{t-1}, \ldots, S_1) = P(S_{t+1}|S_t)$$

- **Emission Assumption:**

  The observation O_{t+1} depends only on the current hidden state S_t and is conditionally independent of other observations.

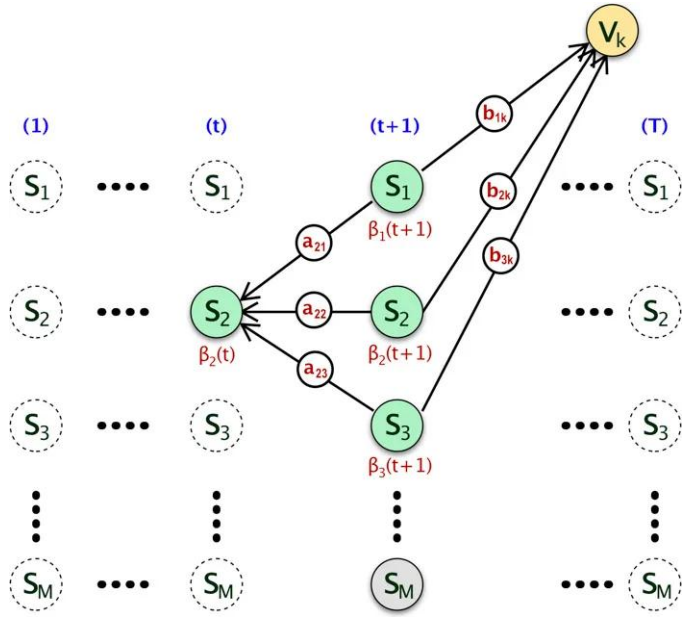
$$P(O_t|S_t, O_{t-1}, \ldots, O_1) = P(O_t|S_t)$$

Hidden Markov Models (HMMs) – Inference Algorithms

Forward-Backward Algorithm:

- **Objective:** Calculate the probability that a certain sequence of hidden states generated a given sequence of observations.
- **Forward Step:**
  - Calculates the probability of the observation sequence up to a certain point, considering all possible states at that point.
  - Definition of forward probability α_t: Probability that the observations up to t are generated and that the system is in state S_i at time t.
  - **Initialization:** $\alpha_1(i) = \pi_i \cdot b_i(O_1)$
  - **Recurrence:**

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

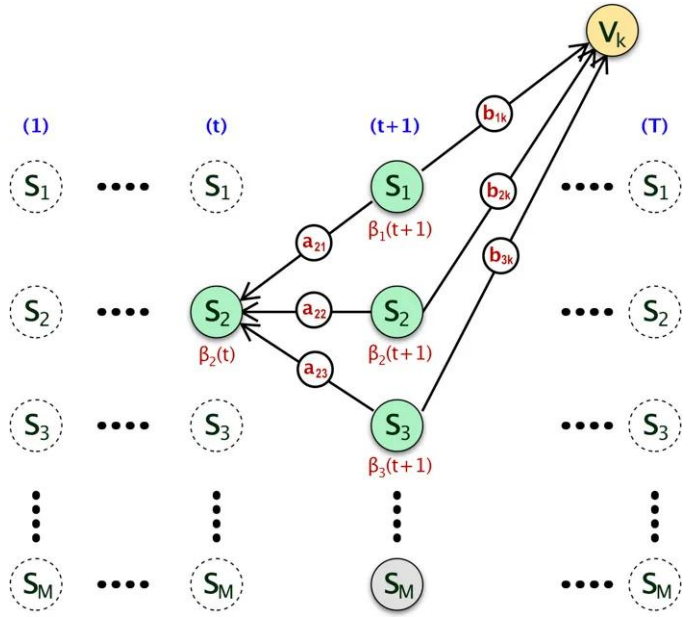


Hidden Markov Models (HMMs) – Inference Algorithms

Forward-Backward Algorithm:

- **Backward Step:**
  - Calculates the probability of future observations given a specific state at a certain point.
  - Definition of backward probability $\beta_t(i)$: Probability of observations O_{t+1} to O_T, given that the system is in state S_i at time t.
  - **Initialization:** $\beta_T(i) = 1$
  - **Recurrence:**

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



Hidden Markov Models (HMMs) – Inference Algorithms

Viterbi Algorithm:

- **Objective:** Find the most probable sequence of hidden states given a sequence of observations.
- **Steps of the Algorithm:**
  - **Initialization:** $\delta_1(i) = \pi_i \cdot b_i(O_1)$ Initial Tracking: $\psi_1(i) = 0$
  - **Recurrence:** $\delta_t(j) = \max_i[\delta_{t-1}(i) \cdot a_{ij}] \cdot b_j(O_t)$ $\psi_t(j) = \arg\max_i[\delta_{t-1}(i) \cdot a_{ij}]$
  - **Termination:** $P^* = \max_i[\delta_T(i)]$ $q_T^* = \arg\max_i[\delta_T(i)]$
  - **Optimal Path:** Backtrack from q_T^* using ψ_t.

- **Example Comparison with the Forward-Backward Algorithm:**
  - **Forward-Backward Algorithm:** Finds the marginal probabilities of each state at each time point.
  - **Viterbi Algorithm:** Identifies a single most probable sequence of states in its entirety.

Hidden Markov Models (HMMs) – Conclusion

HMMs are powerful tools for modeling time sequences with hidden states and performing inferences on hidden states from observations. Understanding their assumptions, structures, and inference algorithms is crucial for their successful application in various fields such as speech recognition, bioinformatics, and financial modeling

Modeling Time Series with HMMs (Practical Applications) – Application Scenarios

- **Speech Recognition:**
  - HMMs are used to model the temporal sequence of phonemes in speech.
  - Each hidden state can represent a phoneme or a combination of phonemes.
  - Examples: Automatic transcription systems, virtual assistants (Siri, Google Assistant).

- **Human Activity Detection:**
  - Analyzing time series from sensors to determine human activities such as walking, running, or remaining stationary.
  - Hidden states: Different activities.
  - Examples: Fitness tracking, smart home systems for the elderly.

- **Bioinformatics:**
  - Analyzing DNA or protein sequences.
  - Hidden states: Functional biological motifs or secondary structures.
  - Examples: Predicting functional sites in proteins, modeling gene expression.

Modeling Time Series with HMMs (Practical Applications) – Pre-processing Techniques for HMMs

- **Normalization and Scaling:**
  - Time series data may require scaling to fit within a specific range.
  - Example: Normalizing audio signal amplitudes.

- **Feature Extraction:**
  - Convert raw data into more informative representations.
  - Speech recognition: extracting parameters like Mel-Frequency Cepstral Coefficients (MFCCs).
  - Activity detection: extracting features like entropy, variance, windowed statistics.
  - Bioinformatics: encoding amino acids with biochemical traits (hydrophobicity, charge, etc.).

- **Segmentation:**
  - Fragment data into time windows where each window corresponds to an observation.
  - Example: Segmenting an audio recording into 20ms frames.

Modeling Time Series with HMMs (Practical Applications) – Modeling and learning

Defining Hidden States and Transition Probabilities

- **Hidden States:**
  - Identify natural hidden states for the problem at hand (phonemes for speech, activities for sensors, genetic motifs).
  - Example: In speech recognition, hidden states may represent phonemes.
- **Transition Probabilities:**
  - Determine transition probabilities between hidden states.
  - Use prior knowledge or estimate these probabilities from data.

Model Training with Sample Data

- **Model Initialization:**
  - Initialize HMM parameters (transition and emission probabilities, initial state probabilities).
  - Use techniques like uniform initialization or expert knowledge insertion.
- **Baum-Welch Algorithm:**
  - Use the Baum-Welch algorithm (a form of Expectation-Maximization) to adjust model parameters based on training data.
  - E-step and M-step for parameter optimization.

Modeling Time Series with HMMs (Practical Applications) – Python Code Example for HMM Training

- **Installation of Required Libraries**

```
pip install numpy hmmlearn
```

- **Data Preparation**

```
import numpy as np
from hmmlearn import hmm

# Simulate example data
X = np.concatenate([np.random.normal(loc=i, scale=0.1, size=100) for i in range(3)])
X = X.reshape(-1, 1) # HMM requires 2D observations even if only one dimension

# Sequence labels for illustration
lengths = [100] * 3 # Each segment has a length of 100
```

- **Initialize and Train the HMM Model**

```
# Initialize the Gaussian HMM model
model = hmm.GaussianHMM(n_components=3, covariance_type="full", n_iter=100)

# Train the model with example data
model.fit(X, lengths)

# Display learned parameters
print("Transition probabilities:\n", model.transmat_)
print("State means:\n", model.means_)
print("State covariances:\n", model.covars_)
```

Modeling Time Series with HMMs (Practical Applications) – Analyze Results

- **Use the Model for Predictions**

```
# Predict the sequence of states for a new observation
logprob, hidden_states = model.decode(X, algorithm="map"  or "viterbi" )

print("Predicted hidden states:\n", hidden_states)
```

- **Visualize Hidden States**

```
import matplotlib.pyplot as plt

# Visualization of hidden states on simulated data
plt.plot(X, label='Observed Data’)
plt.plot(hidden_states, label='Predicted Hidden States', linestyle='--’)
plt.legend()
plt.show()
```

- **Model Evaluation:**
  - Analyze the states predicted by the HMM and verify the concordance with the observed events.
  - Compare results with other methods when applicable to evaluate performance (e.g., k-means, supervised models).

These practical applications underscore the flexibility and power of HMMs in various fields, leveraging their capability to model and infer hidden states from observable sequences effectively. Understanding and implementing HMMs can profoundly impact tasks requiring temporal sequence modeling.

Modeling Time Series with HMMs (Practical Applications) – Analyze Results