Machine Learning

Graphical Models

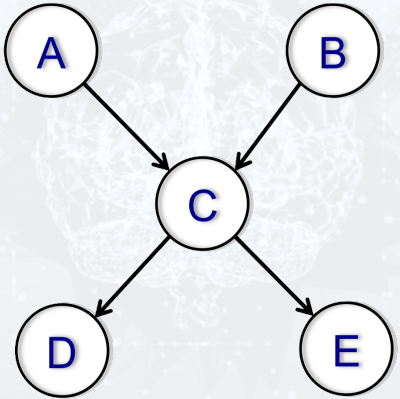
Lecturer: Duc Dung Nguyen, PhD.
Contact: nddung@hcmut.edu.vn

Faculty of Computer Science and Engineering
Hochiminh city University of Technology

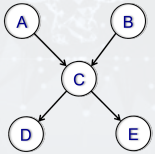
Contents

Bayesian Networks (revisited)

Bayesian Networks



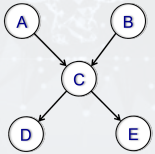
Advantages of graphical modeling:



Advantages of graphical modeling:

- Conditional independence:

$$p(D|C, E, A, B) = p(D|C)$$



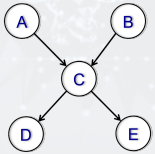
Advantages of graphical modeling:

- Conditional independence:

$$p(D|C, E, A, B) = p(D|C)$$

- Factorization:

$$p(A, B, C, D, E) = p(D|C)p(E|C)p(C|A, B)p(A)p(B)$$



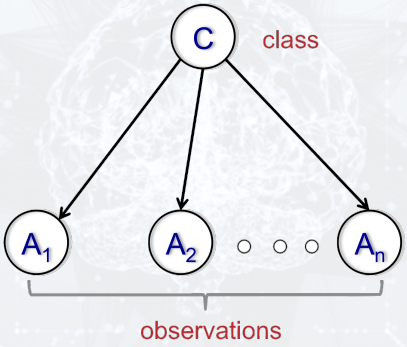
Naive Bayes Classifier (revisited)

Naive Bayes Classifier

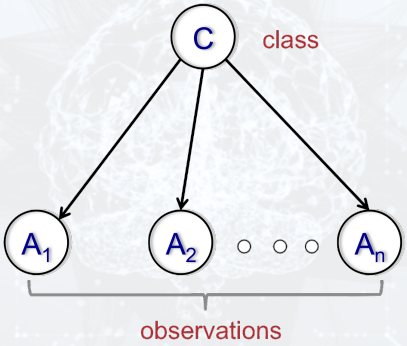
- Each instance x is described by a conjunction of attribute values $< a_1, a_2, ..., a_n >$
- It is to assign the most probable class c to an instance

$$C_{NB} = \arg\max_{c \in C}(a_1, a_2, ..., a_n|c)p(c)$$
$$= \arg\max_{c \in C} \prod_{i=1,n} p(a_i|c).p(c)$$

Naive Bayes Classifier



Naive Bayes Classifier



Joint distribution: $p(C, A_1, A_2, ..., A_n)$

Naive Bayes Classifier

Naive Bayes is a **generative model**:

Naive Bayes Classifier

Naive Bayes is a **generative model**:

- It models a joint distribution: $p(C, A)$

Naive Bayes Classifier

Naive Bayes is a **generative model**:

- It models a joint distribution: $p(C, A)$
- It can generate any distribution on C and A.

Naive Bayes Classifier

Naive Bayes is a **generative model**:

- It models a joint distribution: $p(C, A)$
- It can generate any distribution on C and A.

In contrast to a discriminative model (e.g., CRF)

Naive Bayes Classifier

Naive Bayes is a **generative model**:

- It models a joint distribution: $p(C, A)$
- It can generate any distribution on C and A.

In contrast to a discriminative model (e.g., CRF)

- Conditional distribution: $P(C|A)$

Naive Bayes Classifier

Naive Bayes is a **generative model**:

- It models a joint distribution: $p(C, A)$
- It can generate any distribution on C and A.

In contrast to a discriminative model (e.g., CRF)

- Conditional distribution: $P(C|A)$
- It discriminates C given A

Hidden Markov Models

Hidden Markov Models

- Introduction
- Example
- Independence assumptions
- Forward algorithm
- Viterbi algorithm
- Training
- Application to NER

Hidden Markov Models

- One of the most popular graphical models.
- Dynamic extension of Bayesian networks.
- Sequential extension of Naive Bayes classifier.

Example:

- Your possible looking prior to the exam: $(tired, hungover, scared, fine)$

Example:

- Your possible looking prior to the exam: $(tired, hungover, scared, fine)$
- Your possible activity last night: $(TV, pub, party, study)$

Example:

- Your possible looking prior to the exam: $(tired, hungover, scared, fine)$
- Your possible activity last night: $(TV, pub, party, study)$
- Given the sequence of observations of your looking, guess what you did in previous nights.

Example:

- Your possible looking prior to the exam: $(tired, hungover, scared, fine)$
- Your possible activity last night: $(TV, pub, party, study)$
- Given the sequence of observations of your looking, guess what you did in previous nights.

A model:

Example:

- Your possible looking prior to the exam: $(tired, hungover, scared, fine)$
- Your possible activity last night: $(TV, pub, party, study)$
- Given the sequence of observations of your looking, guess what you did in previous nights.

A model:

- Your looking depends on what you did in the night before.

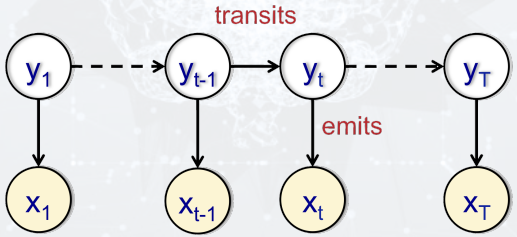
Example:

- Your possible looking prior to the exam: $(tired, hungover, scared, fine)$
- Your possible activity last night: $(TV, pub, party, study)$
- Given the sequence of observations of your looking, guess what you did in previous nights.

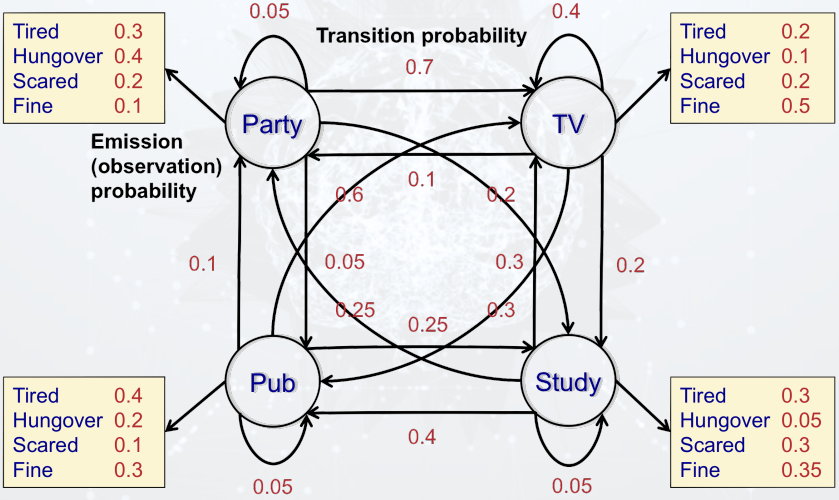
A model:

- Your looking depends on what you did in the night before.
- Your activity in a night depends on what you did in some previous nights.

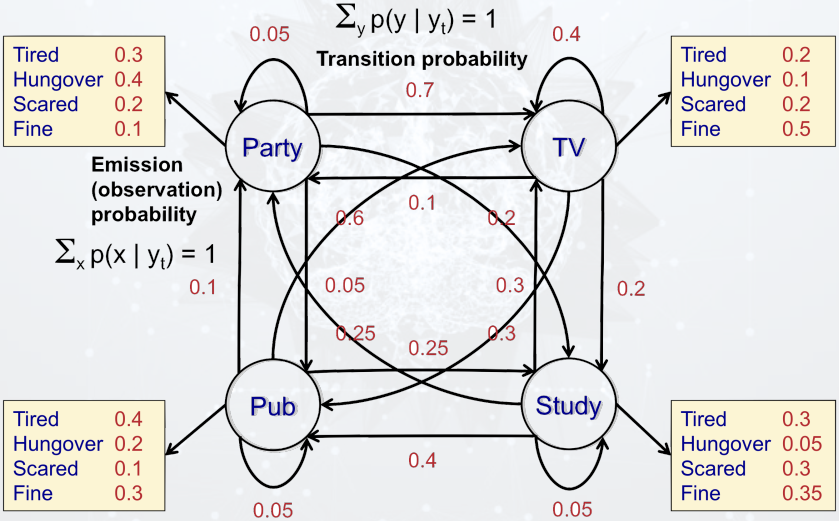
- A finite set of possible observations.
- A finite set of possible hidden states.
- To predict the most probable sequence of underlying stats $\{y_1, y_2, ..., y_T\}$ for a given sequence of observations $\{x_1, x_2, ..., x_T\}$



Hidden Markov Models



Hidden Markov Models



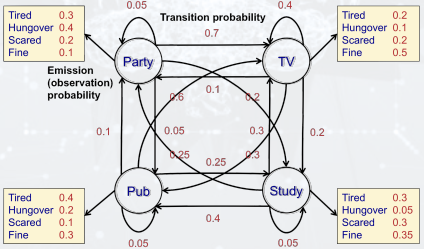
HMM conditional independence assumptions:

- State at time t depends only on state at time $t-1$.

$$p(y_t|y_{t-1}, Z) = p(y_t|y_{t-1})$$

- Observation at time t depends only on state at time t.

$$P(x_t|y_t, Z) = p(x_t|y_t)$$



HMM is a generative model:

- Joint distributions:

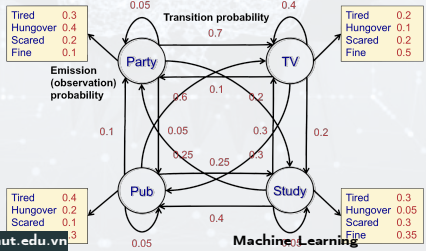
$$
\begin{aligned}
p(Y, X) &= p(y_1, y_2, ..., y_T, x_1, x_2, ..., x_T) \\
&= \prod_{t=1,T} p(y_t|y_{t-1}).p(x_t|y_t) \\
p(y_1|y_0) &= p(y_1)
\end{aligned}
$$

HMM is a generative model:

- Joint distributions:

$$p(Y, X) = p(y_1, y_2, ..., y_T, x_1, x_2, ..., x_T)$$
$$= \prod_{t=1,T} p(y_t|y_{t-1}).p(x_t|y_t)$$
$$p(y_1|y_0) = p(y_1)$$

- It can generate any distribution on Y and X



HMM is a generative model:

- Joint distributions:

$$p(Y, X) = p(y_1, y_2, ..., y_T, x_1, x_2, ..., x_T)$$
$$= \prod_{t=1,T} p(y_t|y_{t-1}).p(x_t|y_t)$$
$$p(y_1|y_0) = p(y_1)$$

- It can generate any distribution on Y and X

In contrast to a discriminative model (e.g., CRF):

- Conditional distributions: $p(Y|X)$
- It discriminates Y given X.

Forward algorithm:

- To compute the joint probability of the state at the time t being y_t and the sequence of observations in the first t steps being $\{x_1, x_2, ..., x_t\}$:

$$\alpha_t(y_t) = p(y_t, x_1, x_2, ..., x_t)$$

Forward algorithm:

- To compute the joint probability of the state at the time t being y_t and the sequence of observations in the first t steps being $\{x_1, x_2, ..., x_t\}$:

$$\alpha_t(y_t) = p(y_t, x_1, x_2, ..., x_t)$$

- Bayes' theorem gives:

$$\begin{aligned} p(y_t|x_1, x_2, ..., x_t) &= p(y_t, x_1, x_2, ..., x_t)/p(x_1, x_2, ..., x_t) \\ &= \alpha_t(y_t)/p(x_1, x_2, ..., x_t) \end{aligned}$$

Forward algorithm:

- To compute the joint probability of the state at the time t being y_t and the sequence of observations in the first t steps being $\{x_1, x_2, ..., x_t\}$:

$$\alpha_t(y_t) = p(y_t, x_1, x_2, ..., x_t)$$

- Bayes' theorem gives:

$$\begin{aligned} p(y_t|x_1, x_2, ..., x_t) &= p(y_t, x_1, x_2, ..., x_t)/p(x_1, x_2, ..., x_t) \\ &= \alpha_t(y_t)/p(x_1, x_2, ..., x_t) \end{aligned}$$

- The highest $\alpha_t(y_t)$ is the most likely y_t would be given the same $\{x_1, x_2, ..., x_t\}$.

Forward algorithm:

$$\begin{aligned}
\alpha_t(y_t) &= p(y_t, x_1, x_2, ..., x_t) = \sum_{y_{t-1}} p(y_t, y_{t-1}, x_1, x_2, ..., x_t) \\
&= \sum_{y_{t-1}} p(x_t|y_t, y_{t-1}, x_1, x_2, ..., x_{t-1}) p(y_t, y_{t-1}, x_1, x_2, ..., x_{t-1}) \\
&= \sum_{y_{t-1}} p(x_t|y_t) p(y_t|y_{t-1}, x_1, x_2, ..., x_{t-1}) p(y_{t-1}, x_1, x_2, ..., x_{t-1}) \\
&= \sum_{y_{t-1}} p(x_t|y_t) p(y_t|y_{t-1}) p(y_{t-1}, x_1, x_2, ..., x_{t-1}) \\
&= p(x_t|y_t) \sum_{y_{t-1}} p(y_t|y_{t-1}) \alpha_{t-1}(y_{t-1})
\end{aligned}$$

Forward algorithm:

$$\alpha_t(y_t) = p(y_t, x_1, x_2, ..., x_t) = \sum_{y_{t-1}} p(y_t, y_{t-1}, x_1, x_2, ..., x_t)$$

$$= \sum_{y_{t-1}} p(x_t|y_t, y_{t-1}, x_1, x_2, ..., x_{t-1}) p(y_t, y_{t-1}, x_1, x_2, ..., x_{t-1})$$

$$= \sum_{y_{t-1}} p(x_t|y_t) p(y_t|y_{t-1}, x_1, x_2, ..., x_{t-1}) p(y_{t-1}, x_1, x_2, ..., x_{t-1})$$

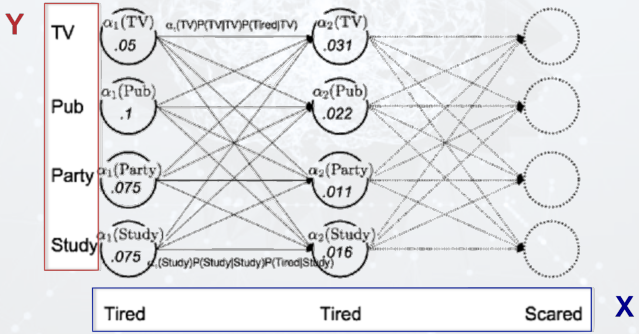
$$= \sum_{y_{t-1}} p(x_t|y_t) p(y_t|y_{t-1}) p(y_{t-1}, x_1, x_2, ..., x_{t-1})$$

$$= p(x_t|y_t) \sum_{y_{t-1}} p(y_t|y_{t-1}) \alpha_{t-1}(y_{t-1})$$

$$\alpha_1(y_1) = p(y_1, x_1) = p(x_1|y_1) p(y_1)$$

Forward algorithm:

$$\alpha_t(y_t) = p(x_t|y_t) \sum_{y_{t-1}} p(y_t|y_{t-1}).\alpha_{t-1}(y_{t-1})$$



Viterbi algorithm:

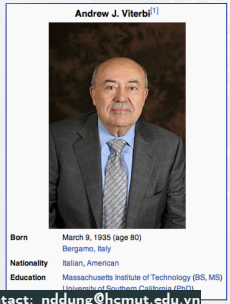
- To compute the most probable sequence of states $\{y_1, y_2, ..., y_T\}$ given a sequence of observations $\{x_1, x_2, ..., x_T\}$:

$$Y^* = \arg\max_Y p(Y|X) = \arg\max_Y p(Y, X)$$

Viterbi algorithm:

- To compute the most probable sequence of states $\{y_1, y_2, ..., y_T\}$ given a sequence of observations $\{x_1, x_2, ..., x_T\}$:

$$Y^* = \arg\max_Y p(Y|X) = \arg\max_Y p(Y, X)$$

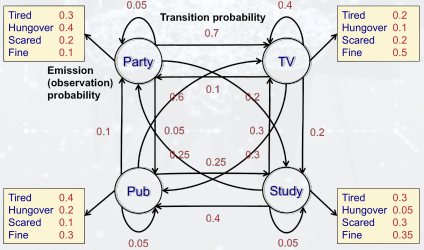


- **Viterbi algorithm**:

$$\max_{y_{1:T}} p(y_1, y_2, ..., y_T, x_1, x_2, ..., x_T) = \max_{y_T} \max_{y_{1:T-1}} p(y_1, y_2, ..., y_T, x_1, x_2, ..., x_T)$$

$$= \max_{y_T} \max_{y_{1:T-1}} \{p(x_T|y_T).p(y_T|y_{T-1})p(y_1, ..., y_{T-1}, x_1, x_2, ..., x_{T-1})\}$$

$$= \max_{y_T} \max_{y_{T-1}} \left\{ p(x_T|y_T).p(y_T|y_{T-1}) \max_{y_{1:T-2}} p(y_1, ..., y_{T-1}, x_2, ..., x_{T-1}) \right\}$$



- **Viterbi algorithm**:

$$\max_{y_{1:T}} p(y_1, y_2, ..., y_T, x_1, x_2, ..., x_T) = \max_{y_T} \max_{y_{1:T-1}} p(y_1, y_2, ..., y_T, x_1, x_2, ..., x_T)$$
$$= \max_{y_T} \max_{y_{1:T-1}} \{p(x_T|y_T).p(y_T|y_{T-1})p(y_1, ..., y_{T-1}, x_1, x_2, ..., x_{T-1})\}$$
$$= \max_{y_T} \max_{y_{T-1}} \left\{ p(x_T|y_T).p(y_T|y_{T-1}) \max_{y_{1:T-2}} p(y_1, ..., y_{T-1}, x_2, ..., x_{T-1}) \right\}$$

- **Dynamic programming**:
  - Compute

$$\arg\max_{y_1} p(y_1, x_1) = \arg\max_{y_1} p(x_1|y_1).p(y_1)$$

  - For each t from 2 to T, and for each state y_t, compute:

$$\arg\max_{y_{1:t-1}} p(y_1, y_2, ..., y_T, x_1, x_2, ..., x_t)$$

  - Select

$$\arg\max_{y_{1:T}} p(y_1, y_2, ..., y_T, x_1, x_2, ..., x_T)$$

- **Dynamic programming**:
  - Compute

$$\arg\max_{y_1} p(y_1, x_1) = \arg\max_{y_1} p(x_1|y_1).p(y_1)$$

- **Dynamic programming**:
  - Compute

$$\arg\max_{y_1} p(y_1, x_1) = \arg\max_{y_1} p(x_1|y_1).p(y_1)$$

  - For each t from 2 to T, and for each state y_t, compute:

$$\arg\max_{y_{1:t-1}} p(y_1, y_2, ..., y_T, x_1, x_2, ..., x_t)$$

- **Dynamic programming**:
  - Compute

$$\arg\max_{y_1} p(y_1, x_1) = \arg\max_{y_1} p(x_1|y_1).p(y_1)$$

  - For each t from 2 to T, and for each state y_t, compute:

$$\arg\max_{y_{1:t-1}} p(y_1, y_2, ..., y_T, x_1, x_2, ..., x_t)$$

  - Select

$$\arg\max_{y_{1:T}} p(y_1, y_2, ..., y_T, x_1, x_2, ..., x_T)$$

- **Dynamic programming**:
  - Compute

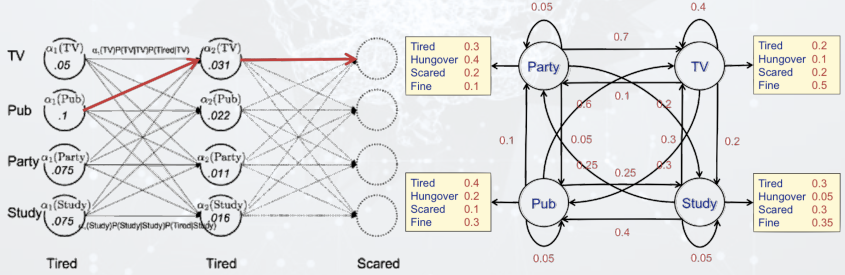
$$\arg\max_{y_1} p(y_1, x_1) = \arg\max_{y_1} p(x_1|y_1).p(y_1)$$

  - For each t from 2 to T, and for each state y_t, compute:

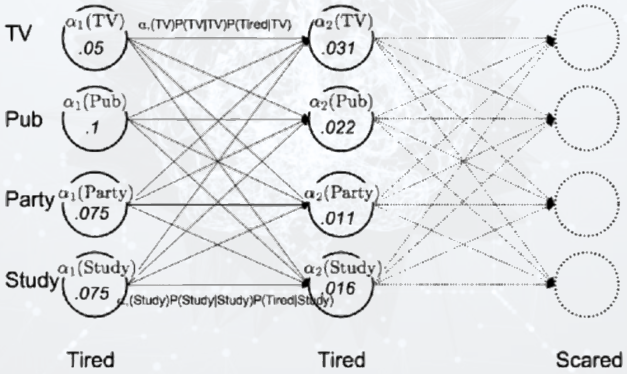
$$\arg\max_{y_{1:t-1}} p(y_1, y_2, ..., y_T, x_1, x_2, ..., x_t)$$

  - Select

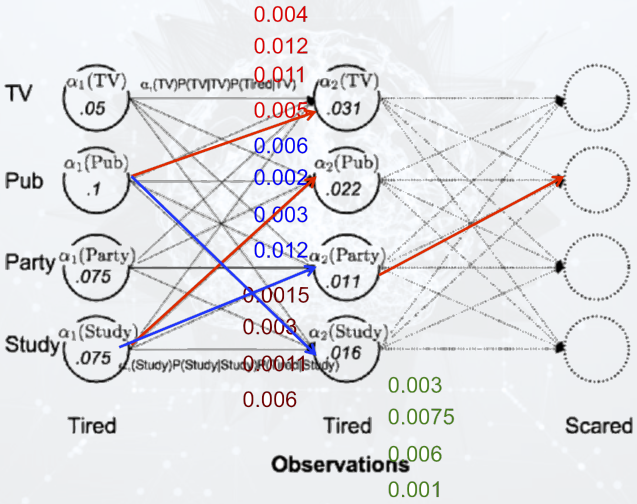
$$\arg\max_{y_{1:T}} p(y_1, y_2, ..., y_T, x_1, x_2, ..., x_T)$$



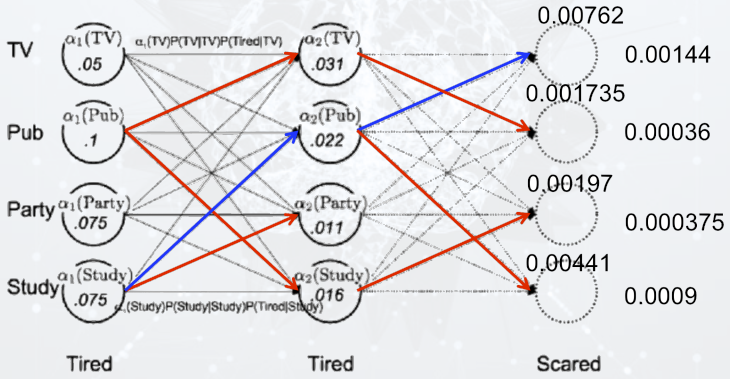
- Could the results from the forward algorithm be used for Viterbi algorithm?



Hidden Markov Models



Could the results from the forward algorithm be used for Viterbi algorithm?



Training HMMs:

- Topology is designed beforehand.
- Parameters to be learned: **emission** and **transition** probabilities.
- Supervised or unsupervised training.

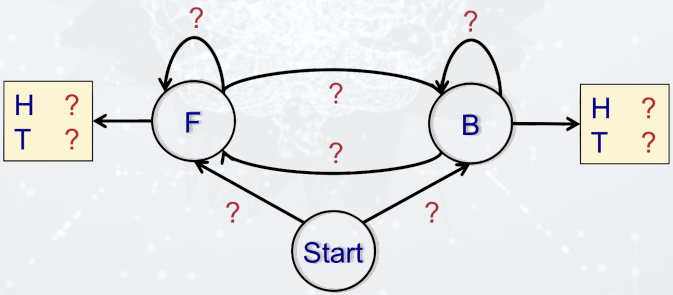
Supervised learning:

- Training data: paired sequences of states and observations $(y_1, y_2, ..., y_T, x_1, x_2, ..., x_t)$
- $p(y_i)$ = number of sequences starting with y_i/number of all sequences.
- $p(y_j|y_i)$ = number of (y_i, y_j)'s / number of all (y_i, y)'s
- $p(x_j|y_i)$ = number of (y_i, x_j)'s / number of all (y_i, x)'s

Supervised learning example:

FFFBFF BFFBFF FFBFFF FFFFBF
HHTHTH THTHTH THHTTH THTTTH

BFFFBF FFFBBF BFFFFF BFBFFF
THHTHT HHTHHT HHTTHT HTTTHH



Unsupervised learning:

- Only observation sequences are available

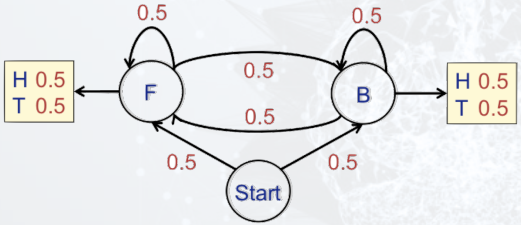
Unsupervised learning:

- Only observation sequences are available

- Iterative improvement of model parameters.
- How?

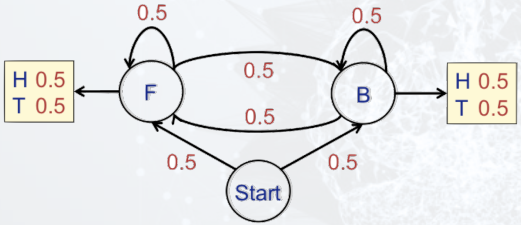
Unsupervised learning:

- Initialize estimated parameters



Unsupervised learning:

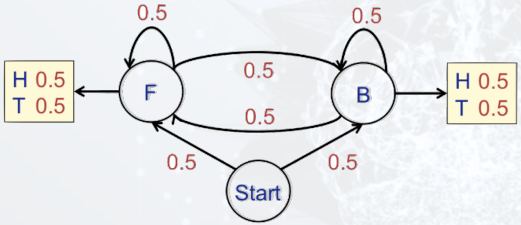
- Initialize estimated parameters



- For each observation sequence, compute the most probable state sequence, using Viterbi algorithm.

Unsupervised learning:

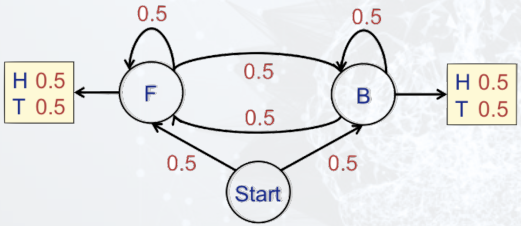
- Initialize estimated parameters



- For each observation sequence, compute the most probable state sequence, using Viterbi algorithm.
- Update the parameters using supervised learning on obtained paired state-observation sequences.

Unsupervised learning:

- Initialize estimated parameters



- For each observation sequence, compute the most probable state sequence, using Viterbi algorithm.
- Update the parameters using supervised learning on obtained paired state-observation sequences.
- Repeat it until convergence.

Application to NER:

- Example: "Facebook CEO Zuckerberg visited Vietnam".

ORG = "Facebook"
PER = "Zuckerberg"
LOC = "Vietnam"
NIL = "CEO", "visited"

Application to NER:

- Example: "Facebook CEO Zuckerberg visited Vietnam".

  ORG = "Facebook"
  PER = "Zuckerberg"
  LOC = "Vietnam"
  NIL = "CEO", "visited"

- States = Class labels

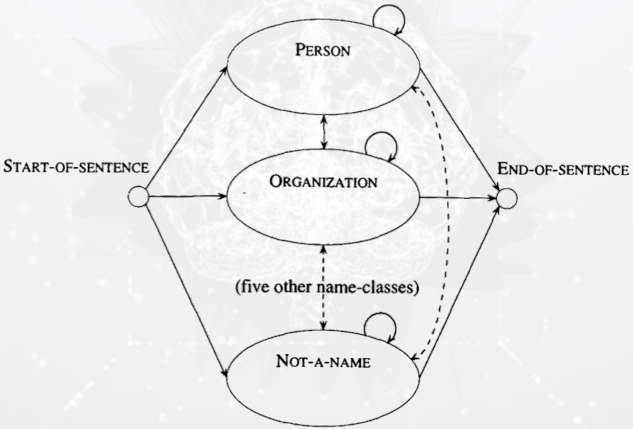
Application to NER:

- Example: "Facebook CEO Zuckerberg visited Vietnam".

  ORG = "Facebook"
  PER = "Zuckerberg"
  LOC = "Vietnam"
  NIL = "CEO", "visited"

- States = Class labels
- Observations = Words + Features

Application to NER:



Bikel, D.M., (1997) A high-performance learning name-finder

Application to NER:

- What if a name is a multi-word phrase?
- Example: "...John von Neumann is ..."

$$\text{B-PER} = \text{"John"}$$
$$\text{I-PER} = \text{"von","Neumann"}$$
$$\text{O} = \text{"is"}$$

- BIO notation: {B-PER, I-PER, B-ORG, I-ORG, B-LOC, I-LOC, B-MISC, I-MISC, O}

Homework

- Readings
  - Marsland, S. (2009) Machine learning:An algorithmic perspective. Chapter 15 (graphical models).
  - Bikel, D. M. etal. (1997) Nymble: a high performance learning name-finder.
- HW
  - Apply Viterbi algorithm to find the most probable 3-state sequence in the looking-activity example in the lecture.
  - Write a program to carry out the unsupervised learning example for HMM in the lecture. Discuss on the result, in particular the convergence of the process.