CS217: Artificial Intelligence and Machine Learning
(associated lab: CS240)

Pushpak Bhattacharyya
CSE Dept.,
IIT Bombay

Week9 of 10mar25, sklearn, P-R-F score, HMM, fwd, bkwd, em

Main points covered: week8 of 3mar25

Prolog

Make and Break

```
Compute_length ([],0).
Compute_length ([Head|Tail], Length):-
    Compute_length (Tail,Tail_length),
    Length is Tail_length+1.
```

<u>High level explanation:</u>

The length of a list is 1 plus the length of the tail of the list, obtained by removing the first element of the list.

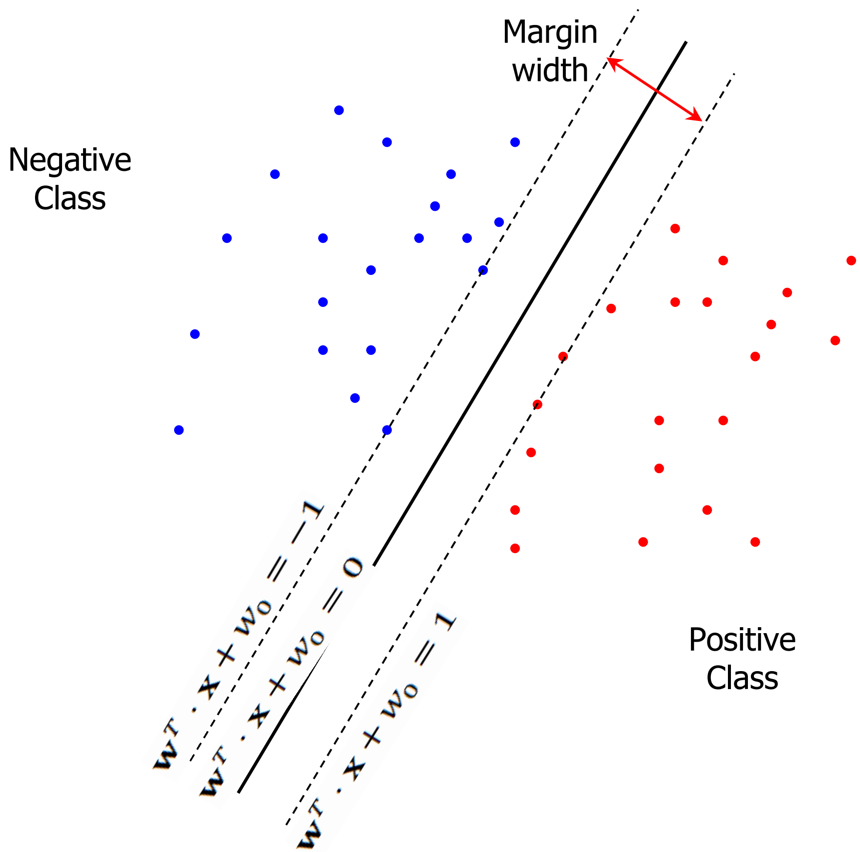
This is a declarative description of the computation.

Syntax

- **<head> :- <body>**
- Read **':-' as 'if'.**
- E.G.
  - *likes(john,X) :- likes(X,cricket).*
  - *"John likes X if X likes cricket".*
  - *i.e., "John likes anyone who likes cricket".*
- Rules always end with '.'.

Support Vector Machine (SVM)

SVM: Optimization Problem



- Objective:
  - Maximize the margin

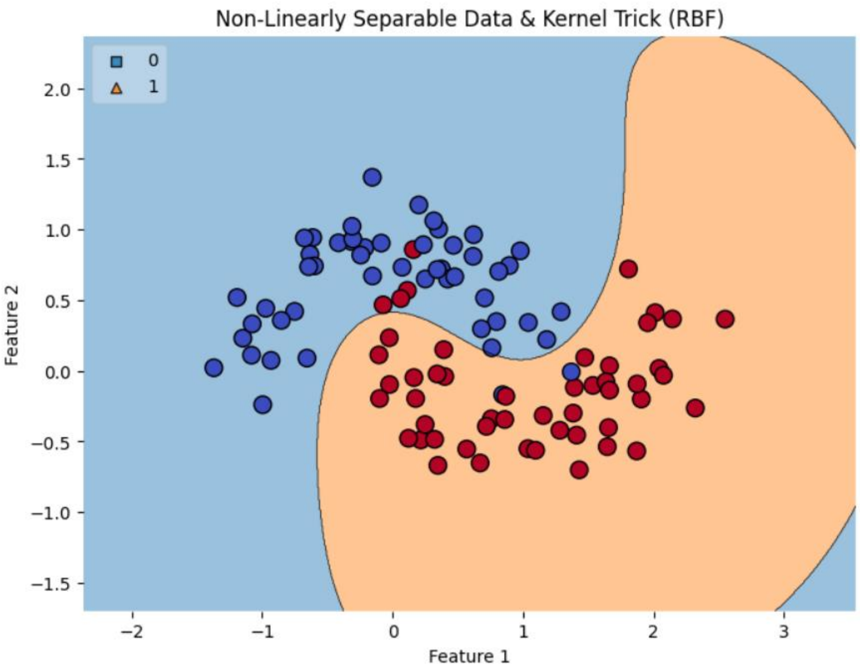
$$\min_{w,b} \left(\frac{1}{2} w^T w \right)$$

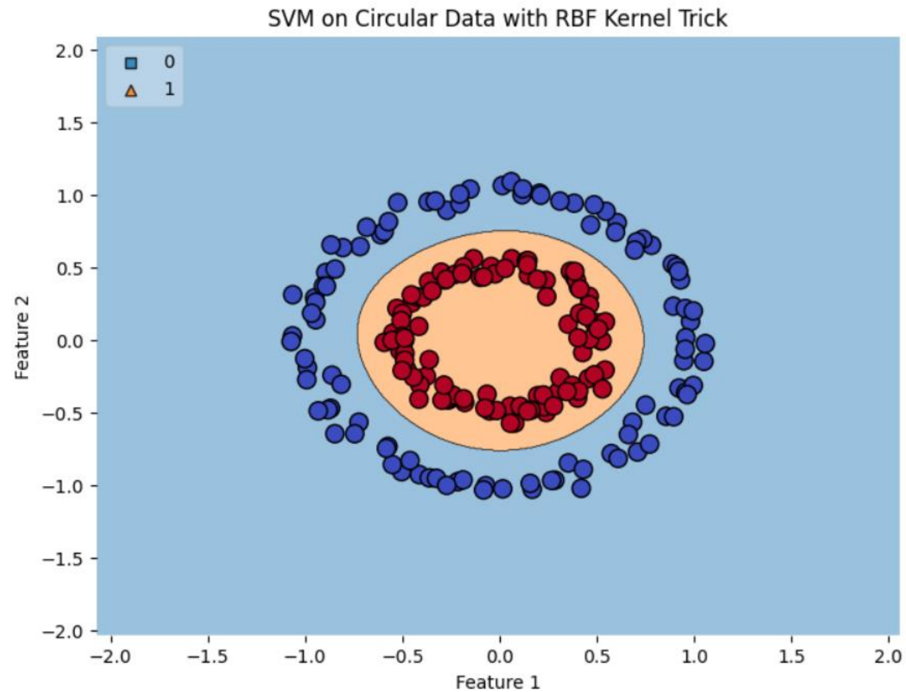
- Subject to the following constraints:
  - Every training instance should lie on the appropriate (positive / negative) side of the linear separator

$$y^i (w^T x^i + b) \geq 1, \quad \forall 1 \leq i \leq N$$

$$-y^i (w^T x^i + b) + 1 \leq 0, \quad \forall 1 \leq i \leq N$$

Kernel Trick for SVM





- SVM works well with linearly separable data.
- But, many real-world data are non-linearly separable in nature
- The kernel function implicitly maps data into a higher-dimensional space where it becomes linearly separable.
- Instead of explicitly transforming data into a higher-dimensional space, we use a kernel function to compute the dot product in that space efficiently.

End main points

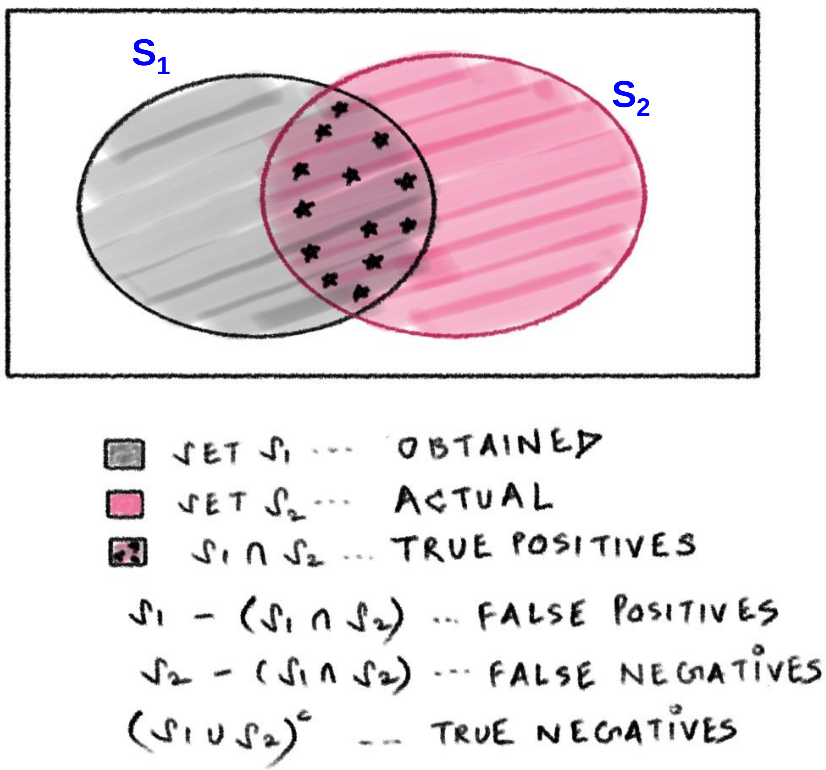
SVM Classifier using sklearn

Slide credit: Dr. Sachin Pawar, ex-CFILT, now in TCS Research

Blood Transfusion Service Center Dataset

- Dataset from the UCI Machine Learning Repository
  - https://archive.ics.uci.edu/ml/datasets.php
- Number of instances = 748
- Number of attributes = 4
- **Attributes / Features**:
  - Recency - months since last donation
  - Frequency - total number of donation
  - Monetary - total blood donated in c.c.
  - Time - months since first donation
- **Class label**:
  - A binary variable representing whether he/she donated blood in March 2007
  - 1 stands for donating blood; 0 stands for not donating blood

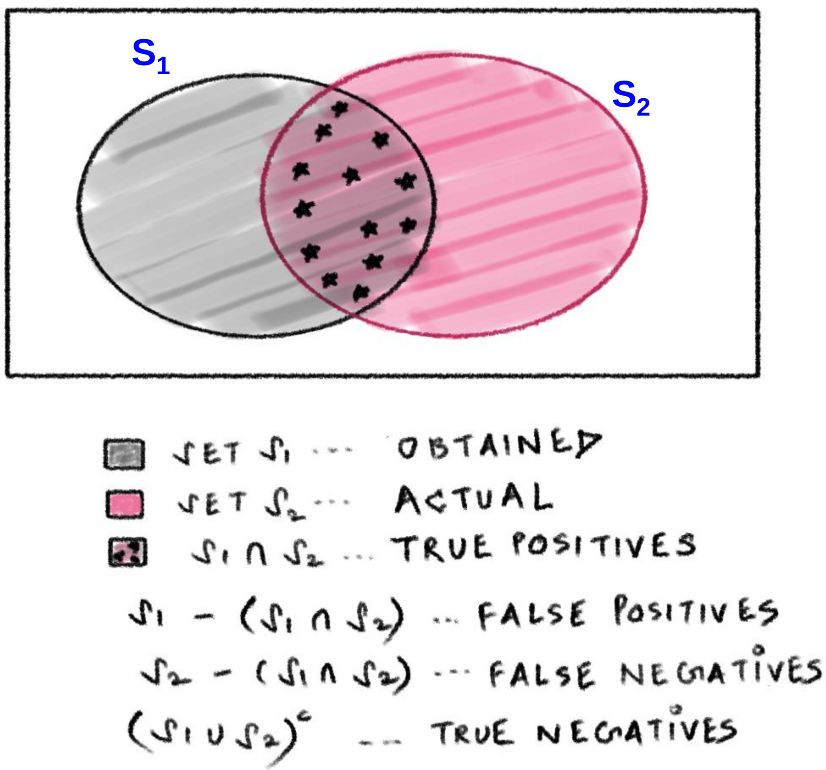
False Positives, False Negatives, Precision, Recall, F-score



$$Precision = \frac{|S_1 \bigcap S_2|}{|S_1|}$$

$$Recall = \frac{|S_1 \bigcap S_2|}{|S_2|}$$

False Positives, False Negatives, Precision, Recall, F-score

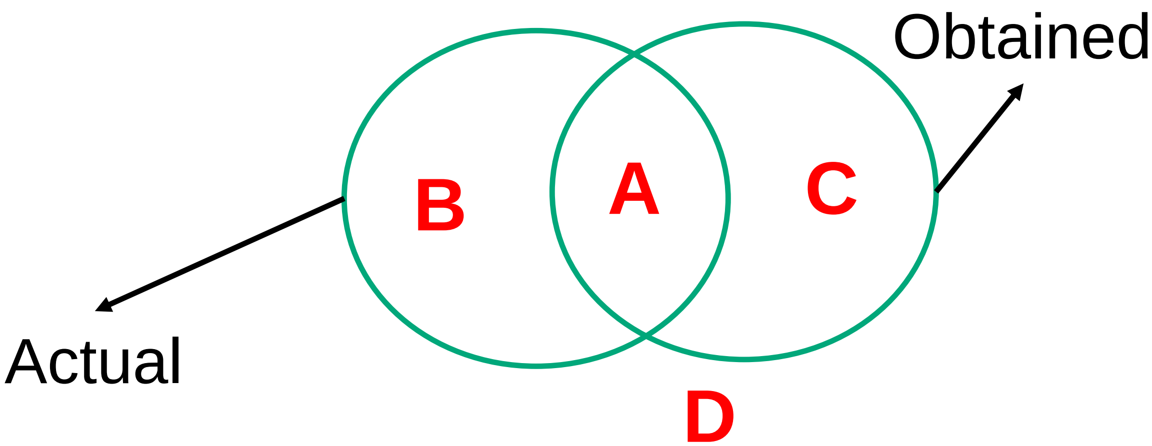


$$Precision = \frac{|S_1 \bigcap S_2|}{|S_1|}$$

$$Recall = \frac{|S_1 \bigcap S_2|}{|S_2|}$$

In terms of contingency table

			Obtained	
		Y	**N**	**Row-Total**
	Y	**A**	**B**	**A+B**
Actual	**N**	**C**	**D**	**C+D**
	Col-Total	**A+C**	**B+D**	**A+B+C+D**



A: TP (true +ve)
B: FN (false -ve)
C: FP
D: TN

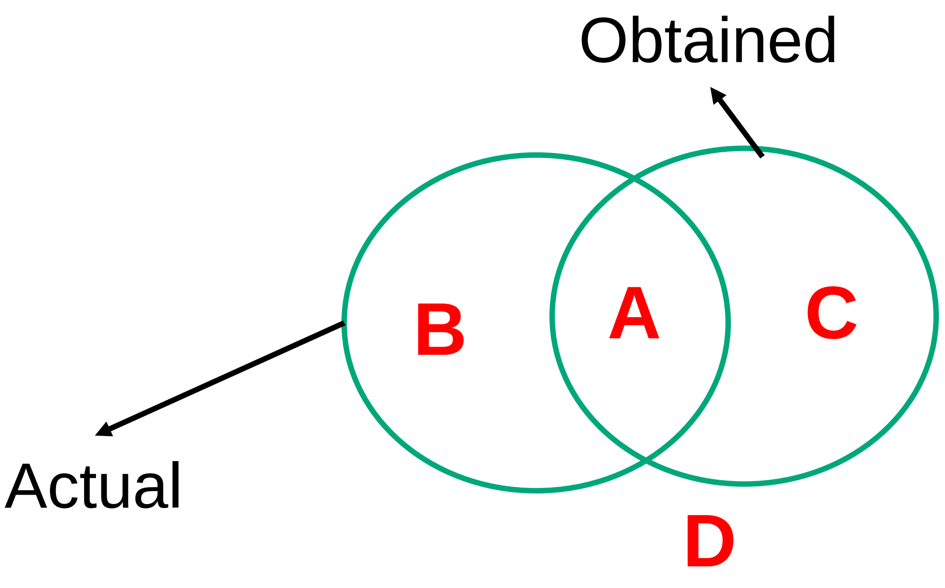
In terms of TP, TN, FP, FN

			Obtained	
		Y	N	Row-Total
	Y	A	B	A+B
Actual	N	C	D	C+D
	Col-Total	A+C	B+D	A+B+C+D

A: TP (true +ve)
B: FN (false -ve)
C: FP
D: TN

$$P = \frac{Actual \cap Obtained}{Obtained}$$

$$= \frac{A}{A+C} = \frac{TP}{TP+FP}$$



$$R = \frac{Actual \cap Obtained}{Actual}$$

$$= \frac{A}{A+B} = \frac{TP}{TP+FN}$$

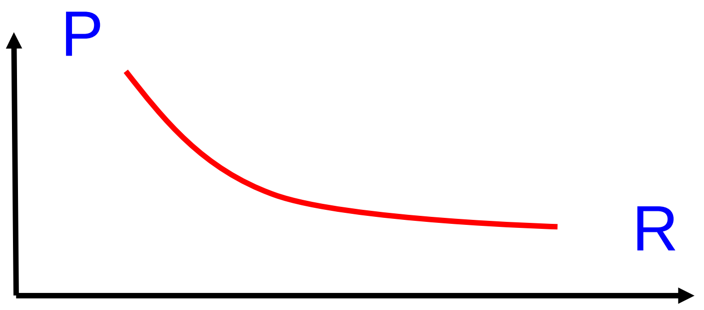
Generalized F-score

$$F_{\beta} = \frac{(1+\beta^2)PR}{\beta^2 P + R} = \frac{1}{\dfrac{\beta^2}{(1+\beta^2)R} + \dfrac{1}{(1+\beta^2)P}}$$

As $\beta \rightarrow 0$, $F_{\beta} \rightarrow P$ and as $\beta \rightarrow \infty$, $F_{\beta} \rightarrow R$

Why F-score?

- P and R need balancing act
- P vs. R is a falling curve
- Harmonic mean gives importance to the smallest of the entities
- We cannot afford to be very low on either P or R
- Hence F-score



```python
import pandas as pd
data_frame = pd.read_csv('../input/blood-transfusion-dataset/transfusion.csv')
print(data_frame.columns)
X = data_frame[['Recency (months)','Frequency (times)','Monetary (c.c. blood)','Time (months)']].to_numpy()
y = data_frame['whether he/she donated blood in March 2007']
print(X.shape)
print(y.shape)
```

```
Index(['Recency (months)', 'Frequency (times)', 'Monetary (c.c. blood)',
       'Time (months)', 'whether he/she donated blood in March 2007'],
      dtype='object')
(748, 4)
(748,)
```

```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=248, random_state=10)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(500, 4)
(500,)
(248, 4)
(248,)
```

```python
from sklearn.preprocessing import StandardScaler
scaling_model = StandardScaler()
scaling_model.fit(X_train)
print(X_train[0:2])
X_train = scaling_model.transform(X_train)
print(X_train[0:2])
X_test = scaling_model.transform(X_test)
```

```
[[   2    2  500   10]
 [   2    6 1500   28]]
[[-0.91873929 -0.57755664 -0.57755664 -0.97689596]
 [-0.91873929  0.12081898  0.12081898 -0.23843941]]
```

```
from sklearn.svm import SVC
SVM_classifier = SVC(C=1.0,kernel='linear',class_weight='balanced')
SVM_classifier.fit(X_train, y_train)
print(len(SVM_classifier.support_vectors_))
```

```
359
```

```python
y_predicted = SVM_classifier.predict(X_test)
print(y_predicted)

from sklearn.metrics import classification_report
report = classification_report(y_test, y_predicted)
print(report)
```

```
[0 1 1 0 1 1 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 0 1
 1 1 0 0 0 1 0 0 1 1 1 0 1 0 0 1 0 0 1 0 1 1 0 1 0 0 0 0 0 0 0 1 1 1 0 0 1
 1 0 0 0 1 1 1 1 0 1 0 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 1 1 0 0 1 0 1 1 1 0 1
 0 0 0 0 1 0 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1 1 0 1 0 1 1 1 0 1 1 1 1 1 1 0 1
 0 1 1 0 1 0 0 1 0 0 1 1 1 1 1 1 1 1 0 1 1 0 0 0 1 1 0 1 0 0 1 1 1 1 1 0 1
 0 1 1 1 1 1 0 1 1 1 0 0 0 1 1 0 1 0 0 0 1 0 1 1 0 1 0 1 0 0 1 0 0 0 1 1 0
 0 1 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0 1 0 0 1 0 0 0 1 0]
              precision    recall  f1-score   support

           0       0.90      0.57      0.70       190
           1       0.36      0.79      0.49        58

    accuracy                           0.62       248
   macro avg       0.63      0.68      0.60       248
weighted avg       0.77      0.62      0.65       248
```

```python
SVM_classifier = SVC(C=1.0,kernel='rbf',class_weight='balanced')
SVM_classifier.fit(X_train, y_train)
print(len(SVM_classifier.support_vectors_))
```

```
361
```

```python
y_predicted = SVM_classifier.predict(X_test)
print(y_predicted)

from sklearn.metrics import classification_report
report = classification_report(y_test, y_predicted)
print(report)
```

```
[0 1 1 0 1 1 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1
 0 1 0 0 0 1 0 0 0 1 1 0 0 0 0 1 0 0 1 0 1 1 0 1 0 0 0 0 0 0 0 1 1 1 0 0 1
 0 0 0 0 0 0 1 1 0 1 0 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 1 1 1 0 1
 0 0 0 0 1 0 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1 1 0 1 0 1 1 0 0 1 1 1 1 1 1 0 0
 0 1 1 0 1 0 0 1 0 0 1 1 0 1 1 1 1 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 1 1 0
 0 1 1 1 1 1 0 1 1 1 0 0 0 1 1 0 1 0 0 0 1 0 1 1 0 1 0 1 0 0 1 0 0 0 1 1 0
 0 1 1 1 0 0 1 1 0 1 1 1 0 1 1 1 0 1 0 0 1 0 0 0 1 0]
              precision    recall  f1-score   support

           0       0.90      0.64      0.74       190
           1       0.39      0.76      0.51        58

    accuracy                           0.67       248
   macro avg       0.64      0.70      0.63       248
weighted avg       0.78      0.67      0.69       248
```

Hidden Markov Model

Noisy Channel Model

$(w_n, w_{n-1}, \ldots, w_1)$ $(t_m, t_{m-1}, \ldots, t_1)$

Sequence *W* is transformed into sequence *T*

$$T^* = \operatorname*{argmax}_{T}(P(T|W))$$

$$W^* = \operatorname*{argmax}_{W}(P(W|T))$$

Bayes Theorem

- $P(B|A)=[P(B).P(A|B)]/P(A)$
- $P(B|A)$: Posterior Probability
- $P(B)$: Prior
- $P(A/B)$: Likelihood

argmax computation

$$T^* = \operatorname{argmax}_T [P(T|W)]$$
$$= \operatorname{argmax}_T = [\{P(T).P(W|T)\}/P(W)]$$
$$= \operatorname{argmax}_T [\{P(T).P(W|T)\}]$$
$$= \operatorname{argmax}_T [P(T,W)]$$

- Choose that T (called T^*) which has the highest probability given W
- Computation with $P(T|W)$ is called **Discriminative Modelling**
- Computation with $P(T,W)$ is called **Generative Modelling**

Application of Bayes Theorem: HMM Based POS Tagging

$$T^* = \arg\max_T (P(T \mid W))$$

$$= \arg\max_T \left[\frac{P(T).P(W \mid T)}{W} \right]$$

$$= \arg\max_T P(T).P(W \mid T)$$

Chain Rule of Probability: Bayes Theorem Generalized

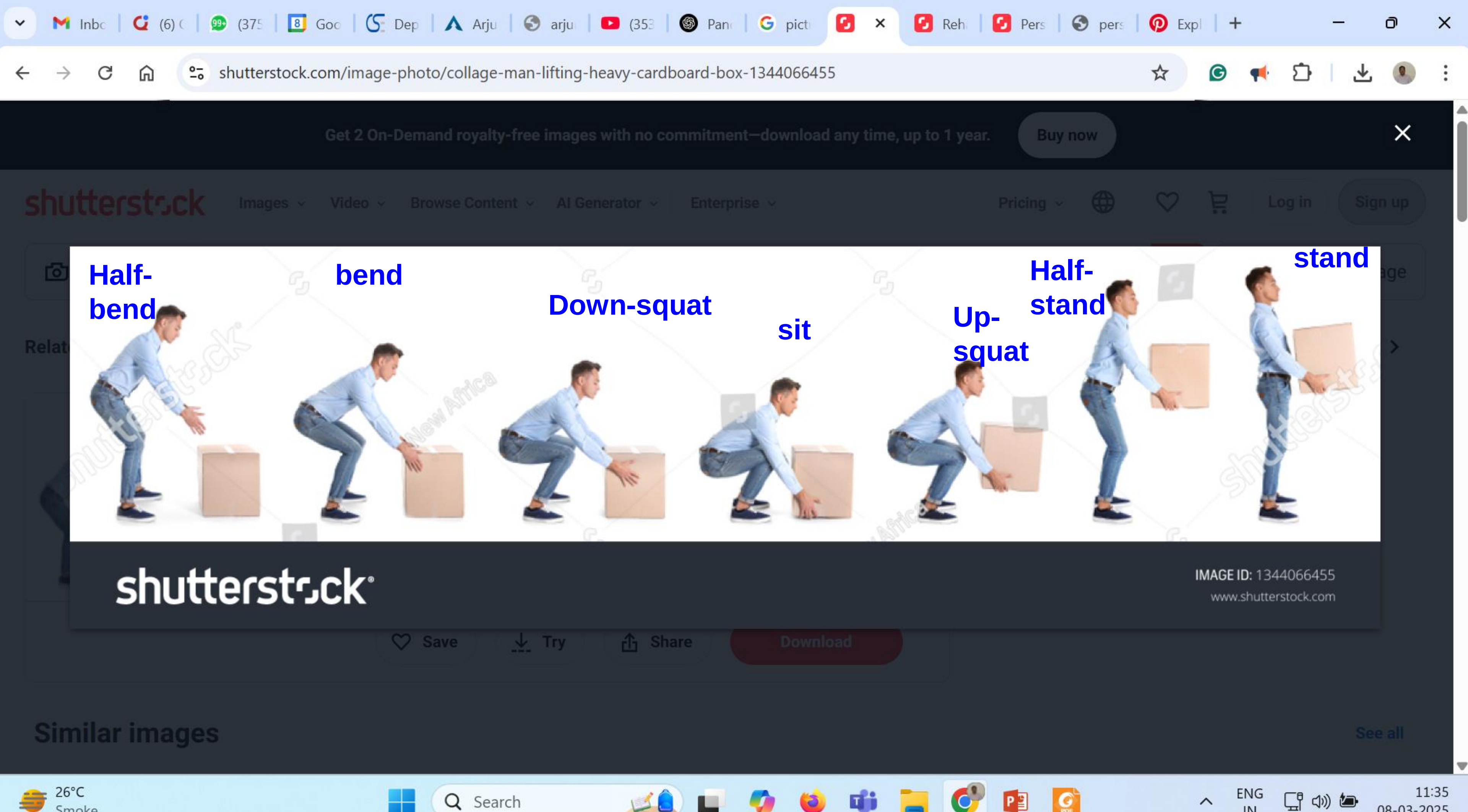
$$P(T) = P(t_0{=}\wedge\ t_1 t_2 \ldots t_{n+1}{=}.)$$

$$= P(t_0)P(t_1|t_0)P(t_2|t_1t_0)P(t_3|t_2t_1t_0) \ldots \ldots P(t_n|t_{n-1}t_{n-2}\ldots t_0)P(t_{n+1}|t_nt_{n-1}\ldots t_0)$$

Considering the current word depends only on the only previous word and not on all the previous words.

$$= P(t_0)P(t_1|t_0)P(t_2|t_1) \ldots P(t_n|t_{n-1})P(t_{n+1}|t_n)$$

Sequence labeling in Computer Vision

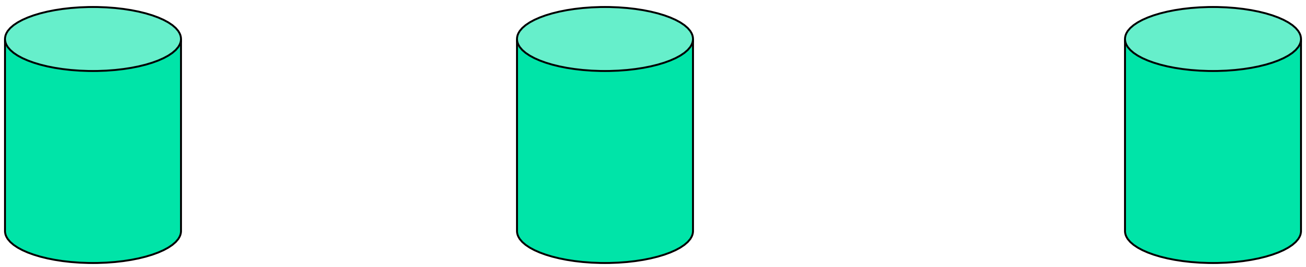


Sequence labelling in NLP, POS Tagging- "To bank, I bank on the bank on the river bank"

- To (IN - Preposition)
- bank (VB - Verb, base form)
- , (PUNCT - Punctuation)
- I (PRP - Pronoun)
- bank (VBP - Verb, non-3rd person singular present)
- on (IN - Preposition)
- the (DT - Determiner)
- bank (NN - Noun, singular)
- on (IN - Preposition)
- the (DT - Determiner)
- river (NN - Noun, singular)
- bank (NN - Noun, singular)

A Motivating Example

Colored Ball choosing



Urn 1
# of Red = 30
# of Green = 50
# of Blue = 20

Urn 2
# of Red = 10
# of Green = 40
# of Blue = 50

Urn 3
# of Red =60
# of Green =10
# of Blue = 30

Probability of transition to another Urn after picking a ball:

	U_1	U_2	U_3
U_1	0.1	0.4	0.5
U_2	0.6	0.2	0.2
U_3	0.3	0.4	0.3

Example (contd.)

Given :

	U_1	U_2	U_3
U_1	0.1	0.4	0.5
U_2	0.6	0.2	0.2
U_3	0.3	0.4	0.3

and

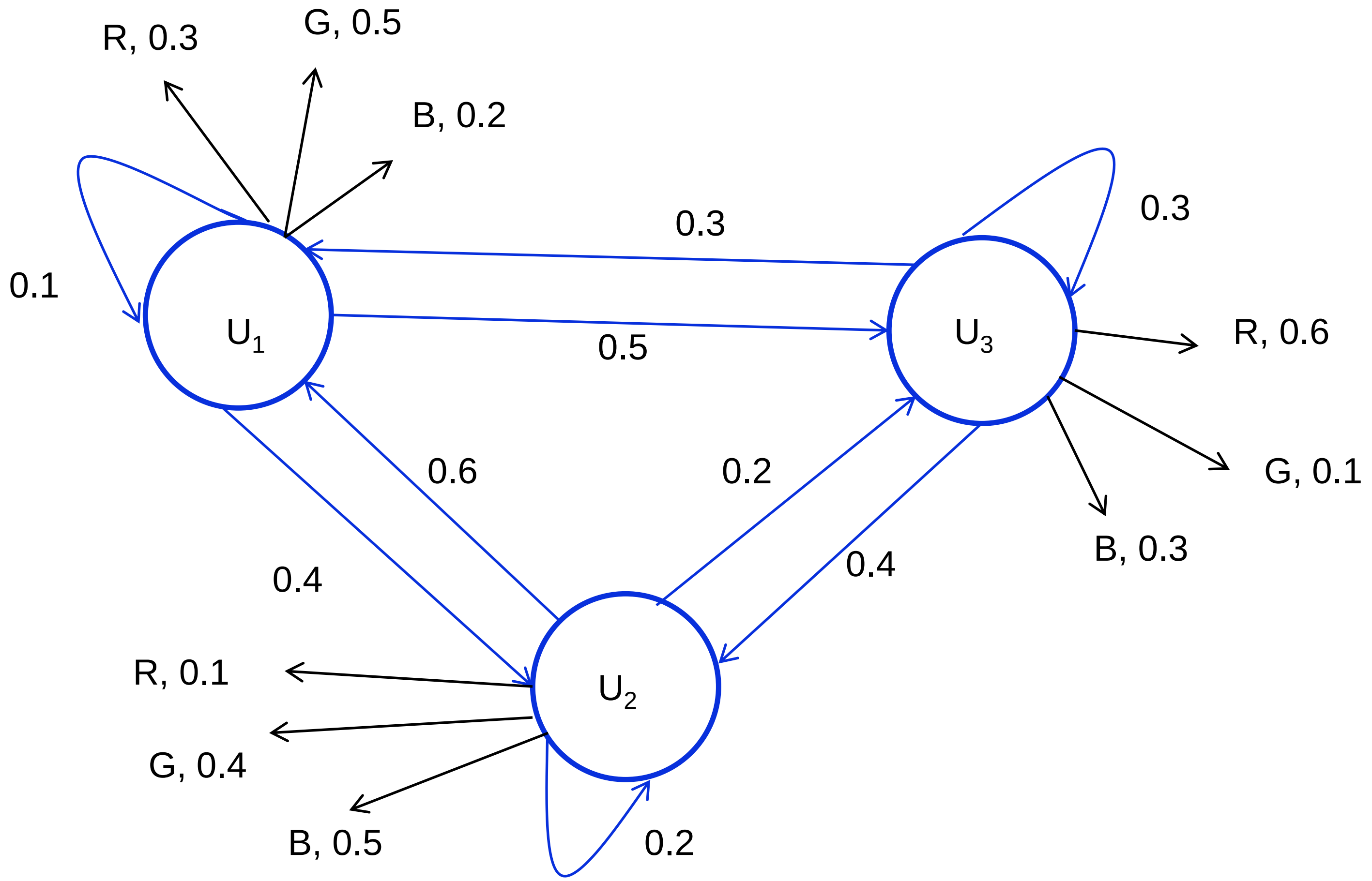
	R	G	B
U_1	0.3	0.5	0.2
U_2	0.1	0.4	0.5
U_3	0.6	0.1	0.3

Observation : RRGGBRGR

State Sequence : ??

Not so Easily Computable.

Diagrammatic representation



Classic problems with respect to HMM

1. Given the observation sequence, find the possible state sequences- **Viterbi**

2. Given the observation sequence, find its probability- **forward/backward** algorithm

3. Given the observation sequence find the HMM prameters.- **Baum-Welch** algorithm

Formal definition of HMM (1/2)

- N= #states
- M= #distinct observation symbols
- State transition probability distribution: $A=\{a_{ij}\}$
- The observation symbol probability distribution in state j, $B=b_j(k)$
- The initial state distribution, $\pi=\{\pi_i\}$

Formal definition of HMM (2/2)

- $A=\{a_{ij}\}$
  - $a_{ij}= P(q_{t+1}=S_j \mid q_t=S_i)$, $1<= i,j <=N$
- $B=b_j(k)$
  - $b_j(k)= P(V_k \text{ at } t \mid q_t= S_j)$, $1<= j <=N$; $1<= k <= M$
- $\pi=\{\pi_i\}$
  - $\pi_i = P(q_1=S_i)$, $1<= i<=N$

Observations and states

	O1	O2	O3	O4	O5	O6	O7	O8
OBS:	R	R	G	G	B	R	G	R
State: S1		S2	S3	S4	S5	S6	S7	S8

$S_i = U_1/U_2/U_3$; A particular state

S: State sequence

O: Observation sequence

S* = "best" possible state (urn) sequence

Goal: Maximize P(S|O) by choosing "best" S

Goal

- Maximize P(S|O) where S is the State Sequence and O is the Observation Sequence

$$S^* = \arg\max_S (P(S \mid O))$$

Baye's Theorem

$$P(A \mid B) = P(A).P(B \mid A) / P(B)$$

P(A) -: Prior

P(B|A) -: Likelihood

$$\arg\max_S P(S \mid O) = \arg\max_S P(S).P(O \mid S)$$

State Transitions Probability

$$P(S) = P(S_{1-8})$$

$$P(S) = P(S_1).P(S_2 \mid S_1).P(S_3 \mid S_{1-2}).P(S_4 \mid S_{1-3})\ldots P(S_8 \mid S_{1-7})$$

By Markov Assumption (k=1)

$$P(S) = P(S_1).P(S_2 \mid S_1).P(S_3 \mid S_2).P(S_4 \mid S_3)\ldots P(S_8 \mid S_7)$$

Observation Sequence probability

$$P(O \mid S) = P(O_1 \mid S_{1-8}).P(O_2 \mid O_1, S_{1-8}).P(O_3 \mid O_{1-2}, S_{1-8})\ldots P(O_8 \mid O_{1-7}, S_{1-8})$$

Assumption that ball drawn depends only on the Urn chosen

$$P(O \mid S) = P(O_1 \mid S_1).P(O_2 \mid S_2).P(O_3 \mid S_3)\ldots P(O_8 \mid S_8)$$

$$P(S \mid O) = P(S).P(O \mid S)$$

$$P(S \mid O) = P(S_1).P(S_2 \mid S_1).P(S_3 \mid S_2).P(S_4 \mid S_3)\ldots P(S_8 \mid S_7).$$

$$P(O_1 \mid S_1).P(O_2 \mid S_2).P(O_3 \mid S_3)\ldots P(O_8 \mid S_8)$$

Grouping terms

	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	
Obs:	ε	R	R	G	G	B	R	G	R	
State:	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9

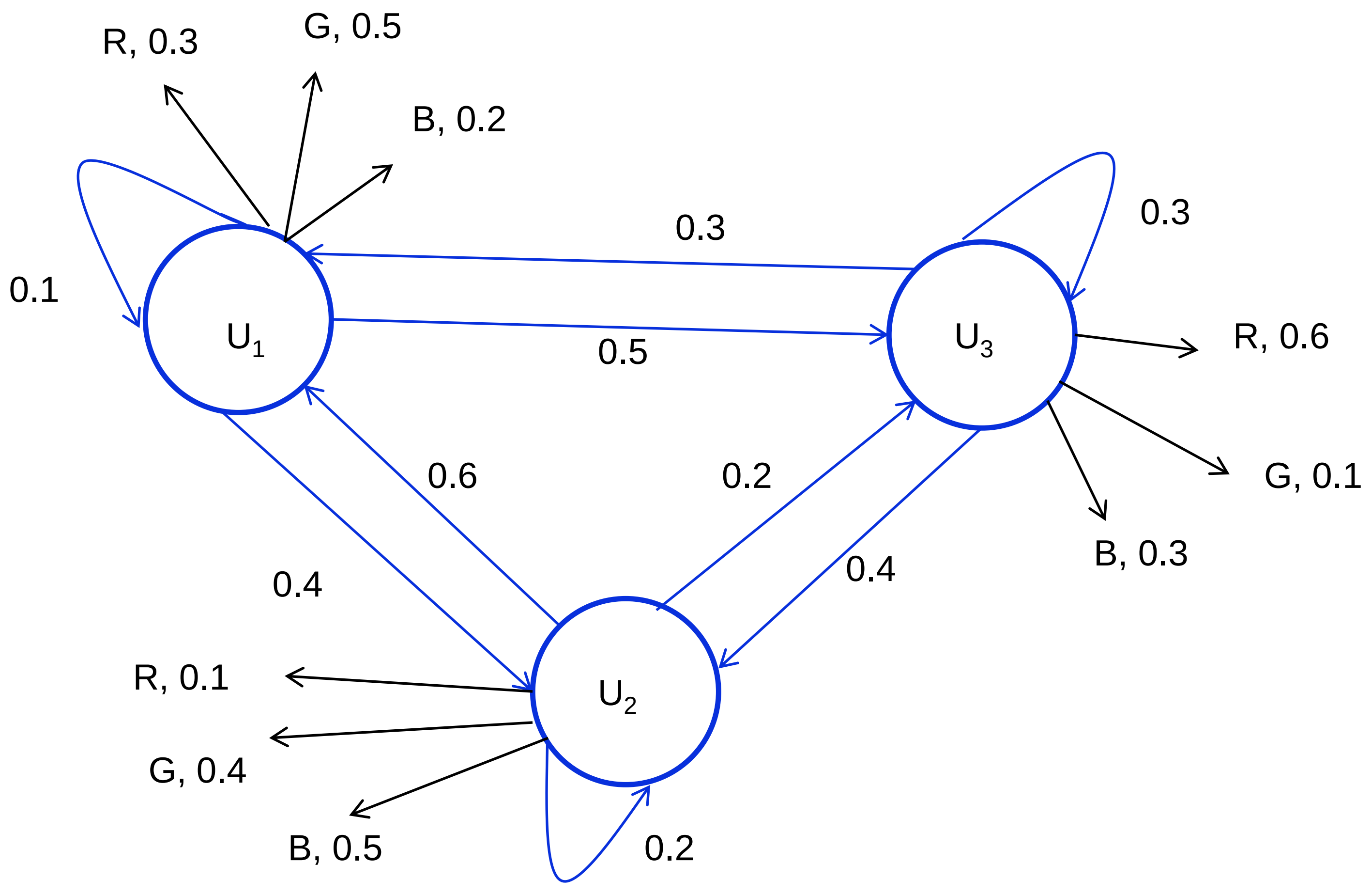
$$
\begin{aligned}
&P(S).P(O|S) \\
&= [P(O_0|S_0).P(S_1|S_0)]. \\
&\quad [P(O_1|S_1).\quad P(S_2|S_1)]. \\
&\quad [P(O_2|S_2).\quad P(S_3|S_2)]. \\
&\quad [P(O_3|S_3).P(S_4|S_3)]. \\
&\quad [P(O_4|S_4).P(S_5|S_4)]. \\
&\quad [P(O_5|S_5).P(S_6|S_5)]. \\
&\quad [P(O_6|S_6).P(S_7|S_6)]. \\
&\quad [P(O_7|S_7).P(S_8|S_7)]. \\
&\quad [P(O_8|S_8).P(S_9|S_8)].
\end{aligned}
$$

We introduce the states S_0 and S_9 as initial and final states respectively.

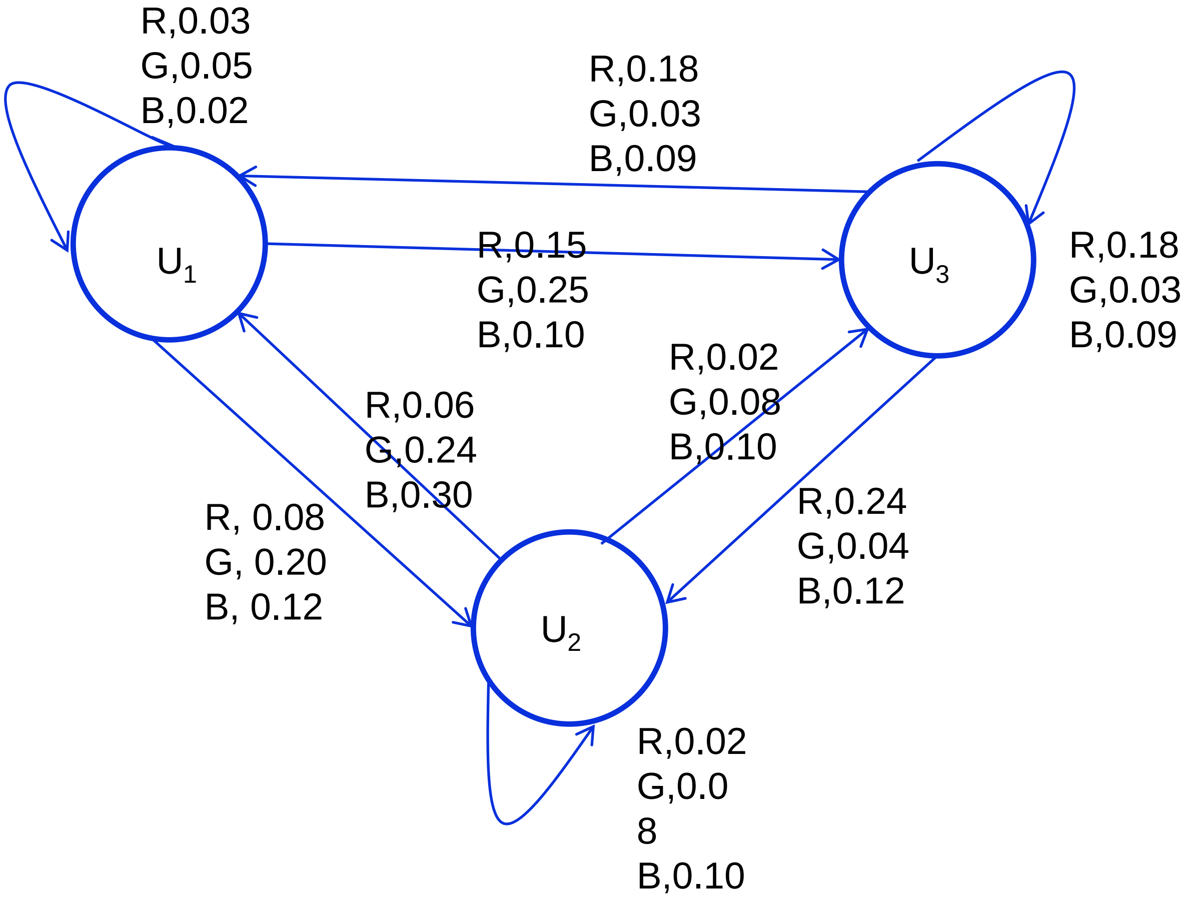
After S_8 the next state is S_9 with probability 1, i.e., $P(S_9|S_8)=1$

O_0 is ε-transition

Back to the automaton of Urns (1/2)

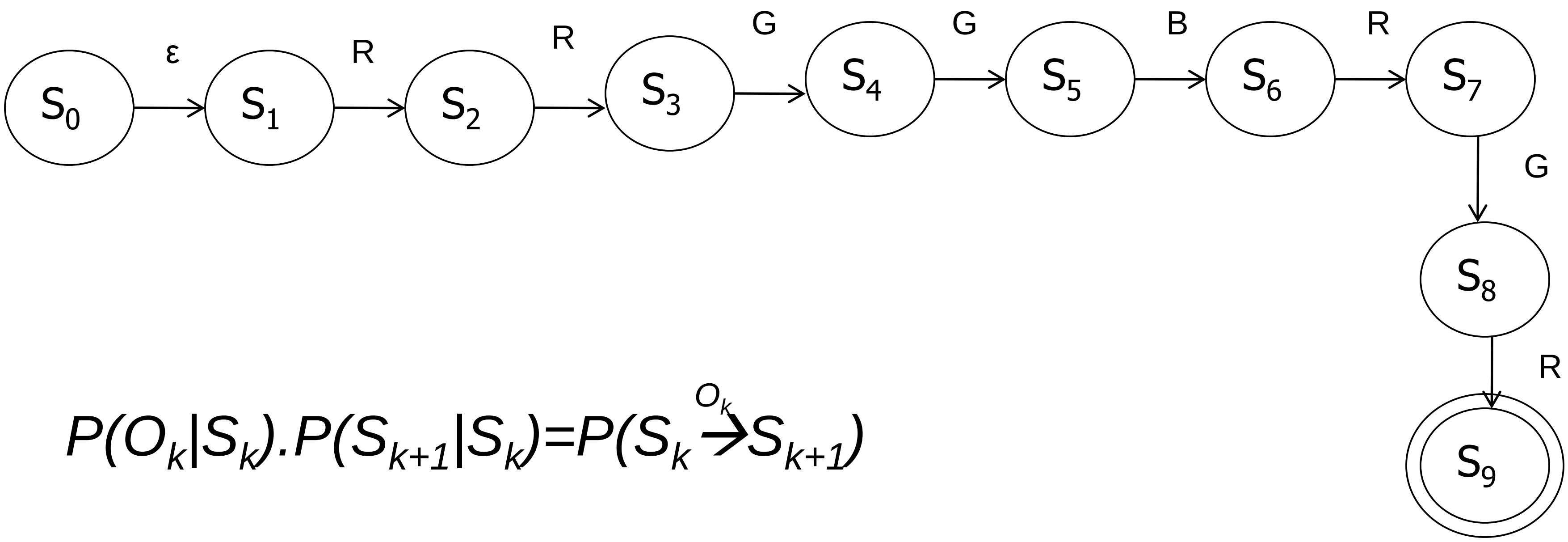


Automaton of urns (2/2)



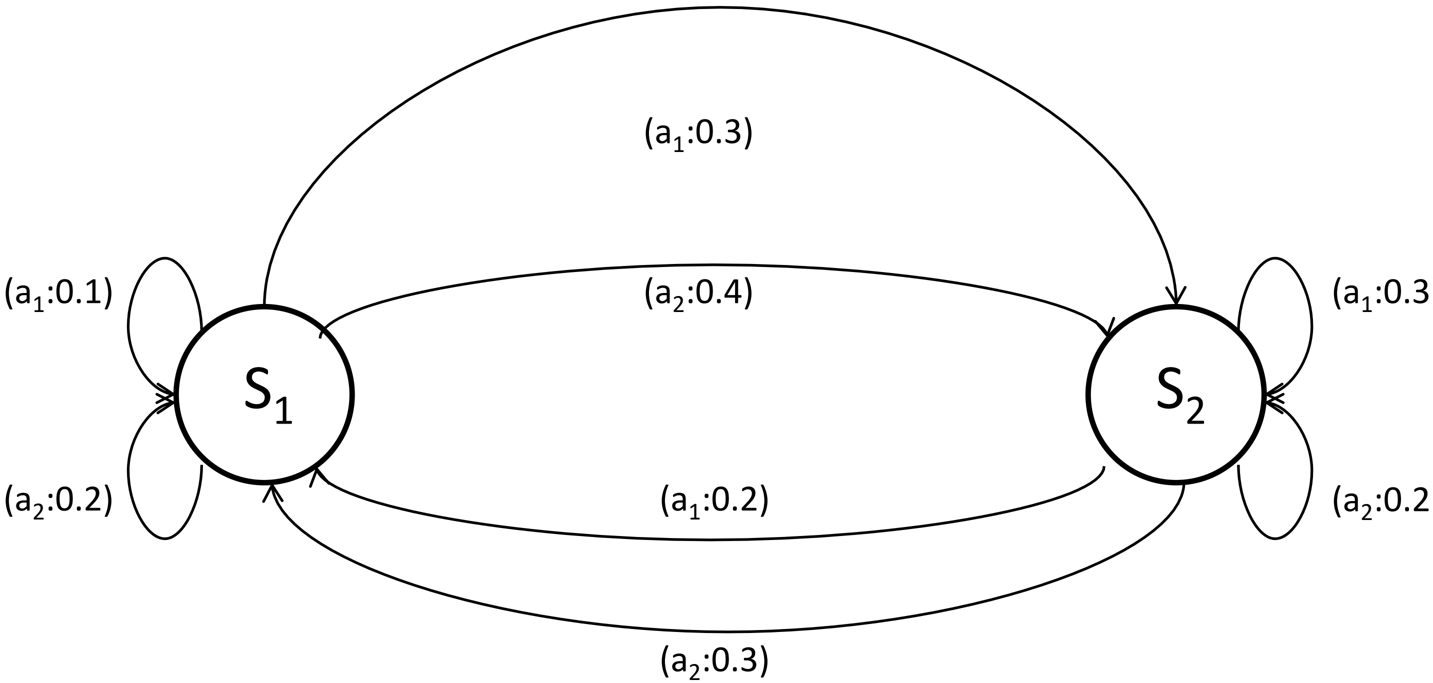
Introducing useful notation

	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	
Obs:	ε	R	R	G	G	B	R	G	R	
State:	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9



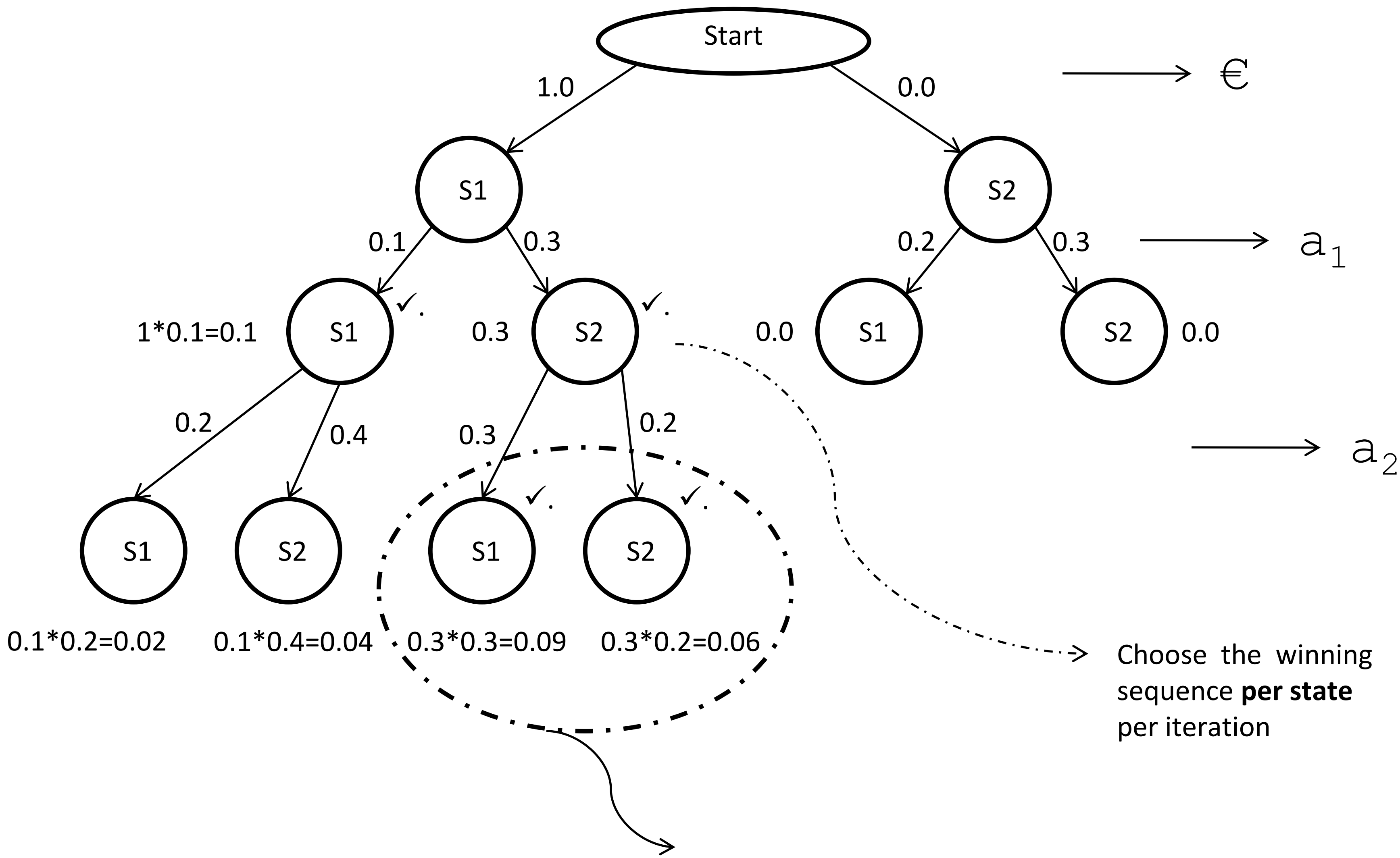
$$P(O_k|S_k).P(S_{k+1}|S_k)=P(S_k \xrightarrow{O_k} S_{k+1})$$

Probabilistic FSM

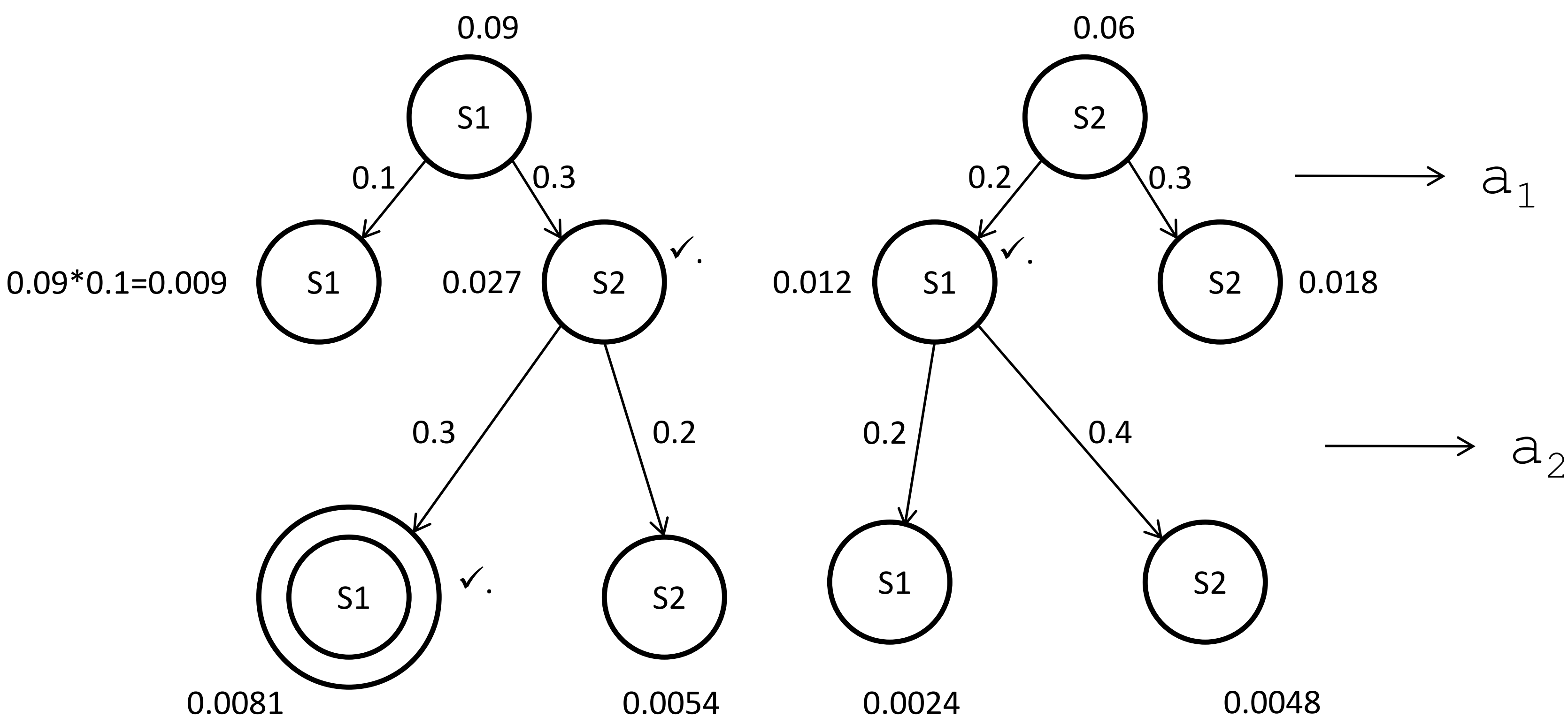


The question here is:
"what is the most likely state sequence given the output sequence seen"

Developing the tree



Tree structure contd…



The problem being addressed by this tree is $S^* = \arg\max_{s} P(S \mid a_1 - a_2 - a_1 - a_2, \mu)$

a1-a2-a1-a2 is the output sequence and μ the model or the machine

Path found:
(working backward)

$$S_1 \xrightarrow{a_1} S_2 \xrightarrow{a_2} S_1 \xrightarrow{a_1} S_2 \xrightarrow{a_2} S_1$$

Problem statement: Find the best possible sequence

$$S^* = \arg\max_{s} P(S \mid O, \mu)$$

$$where,\ S \rightarrow \text{State Seq},\ O \rightarrow \text{Output Seq},\ \mu \rightarrow \text{Model or Machine}$$

$$\text{Model or Machine} = \{\underbrace{S_0}_{\text{Start symbol}}, \underbrace{S}_{\text{State collection}}, \underbrace{A}_{\text{Alphabet set}}, \underbrace{T}_{\text{Transitions}}\}$$

T is defined as $P(S_i \xrightarrow{a_k} S_j) \quad \forall i, j, k$

Tabular representation of the tree

Latest symbol observed / Ending state	€	a_1	a_2	a_1	a_2
S_1	1.0	(1.0*0.1,0.0*0.2) =(**0.1**,0.0)	(0.02, **0.09**)	(0.009, **0.012**)	(0.0024, **0.0081**)
S_2	0.0	(1.0*0.3,0.0*0.3) =(**0.3**,0.0)	(0.04,**0.06**)	(**0.027**,0.018)	(0.0048,0.0054)

Note: Every cell records the winning probability ending in that state

Final winner

The bold faced values in each cell shows the sequence probability ending in that state. Going backward from final winner sequence which ends in state S_2 (indicated By the 2^{nd} tuple), we recover the sequence.

Algorithm

(following James Alan, Natural Language Understanding (2nd edition), Benjamin Cummins (pub.), 1995

Given:

1. The HMM, which means:
   a. Start State: S_1
   b. Alphabet: $A = \{a_1, a_2, \dots a_p\}$
   c. Set of States: $S = \{S_1, S_2, \dots S_n\}$
   d. Transition probability $P(S_i \xrightarrow{a_k} S_j) \quad \forall_{i,j,k}$
      which is equal to $P(S_j, a_k \mid S_i)$
2. The output string $a_1 a_2 \dots a_T$

To find:

The most likely sequence of states $C_1C_2 \dots C_T$ which produces the given output sequence, *i.e.*, $C_1C_2 \dots C_T = \underset{C}{\arg\max}[P(C \mid a_1, a_2, \dots a_T, \mu)]$

Algorithm contd…

Data Structure:

1. A N*T array called SEQSCORE to maintain the winner sequence always (N=#states, T=length of o/p sequence)
2. Another N*T array called BACKPTR to recover the path.

Three distinct steps in the Viterbi implementation

1. Initialization
2. Iteration
3. Sequence Identification

1. Initialization

SEQSCORE(1,1)=1.0

BACKPTR(1,1)=0

For(i=2 to N) do

    SEQSCORE(i,1)=0.0

[expressing the fact that first state is S_1]

2. Iteration

For(t=2 to T) do

  For(i=1 to N) do

    SEQSCORE(i,t) = $Max_{(j=1,N)}$

$$[SEQSCORE(j,(t-1)) * P(Sj \xrightarrow{a_k} Si)]$$

    BACKPTR(I,t) = index *j* that gives the MAX above

3. Seq. Identification

C(T) = i that maximizes SEQSCORE(i,T)

For i from (T-1) to 1 do

C(i) = BACKPTR[C(i+1),(i+1)]

Optimizations possible:

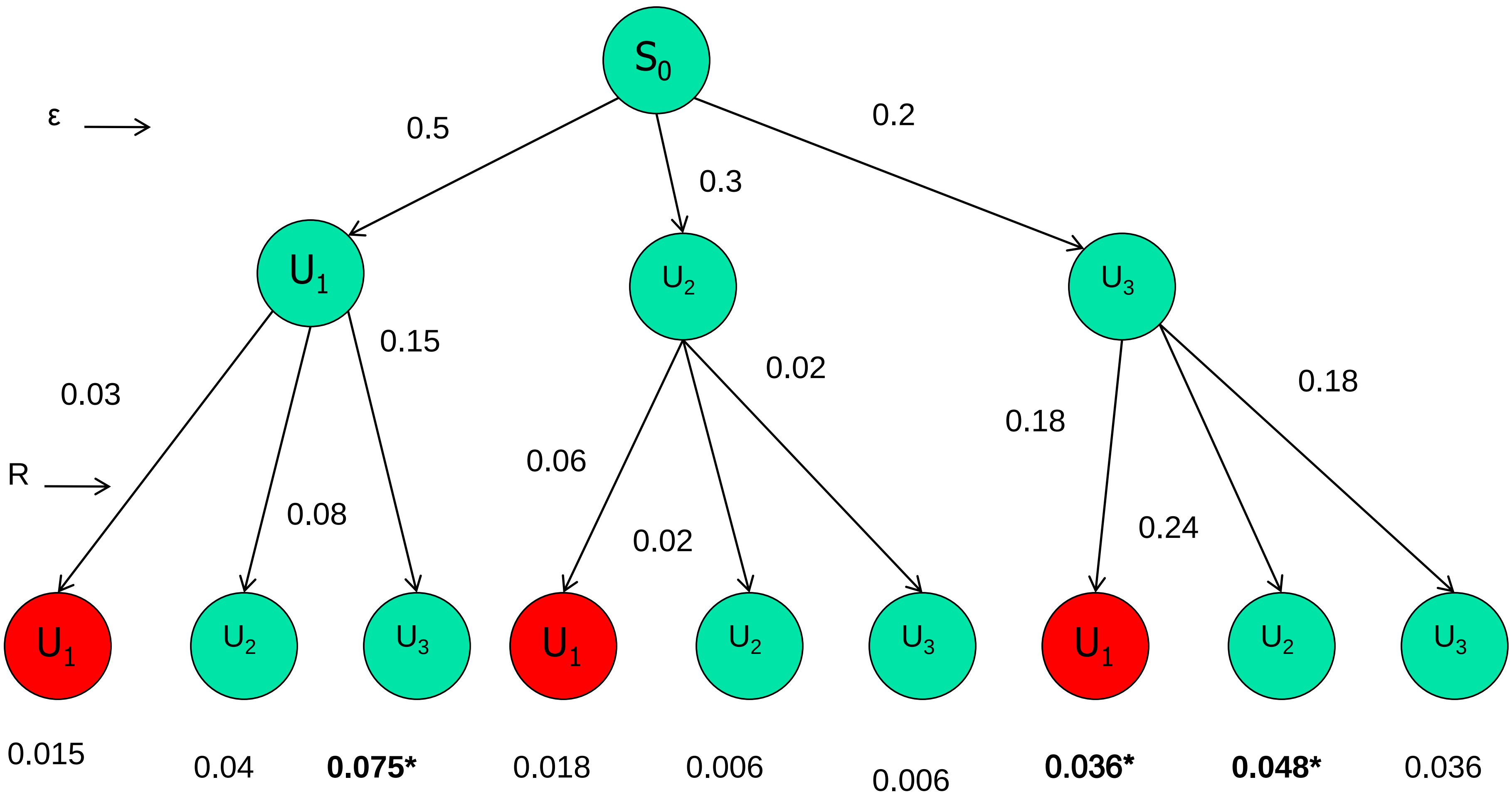
1. BACKPTR can be 1*T
2. SEQSCORE can be T*2

Homework:- Compare this with A*, Beam Search [Homework]

Reason for this comparison:

Both of them work for finding and recovering sequence

Viterbi Algorithm for the Urn problem (first two symbols)



: winner sequences

Markov process of order>1 (say 2)

	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	
Obs:	ε	R	R	G	G	B	R	G	R	
State:	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9

Same theory works

$P(S).P(O|S)$

$= P(O_0|S_0).P(S_1|S_0).$
$[P(O_1|S_1).\quad P(S_2|S_1S_0)].$
$[P(O_2|S_2).\quad P(S_3|S_2S_1)].$
$[P(O_3|S_3).P(S_4|S_3S_2)].$
$[P(O_4|S_4).P(S_5|S_4S_3)].$
$[P(O_5|S_5).P(S_6|S_5S_4)].$
$[P(O_6|S_6).P(S_7|S_6S_5)].$
$[P(O_7|S_7).P(S_8|S_7S_6)].$
$[P(O_8|S_8).P(S_9|S_8S_7)].$

We introduce the states S_0 and S_9 as initial and final states respectively.

After S_8 the next state is S_9 with probability 1, i.e., $P(S_9|S_8S_7)=1$

O_0 is ε-transition

Probability of observation sequence

Why probability of observation sequence?: Language modeling problem

Probabilities computed in the context of corpora

1. P("The sun rises in the east")
2. P("The sun rise in the east")
   - Less probable because of grammatical mistake.
3. P(The svn rises in the east)
   - Less probable because of lexical mistake.
4. P(The sun rises in the west)
   - Less probable because of semantic mistake.

Uses of language model

1. Detect well-formedness
   - Lexical, syntactic, semantic, pragmatic, discourse
2. Language identification
   - Given a piece of text what language does it belong to.

     Good morning - English

     Guten morgen - German

     Bon jour - French
3. Automatic speech recognition
4. Machine translation

How to compute $P(o_0 o_1 o_2 o_{3 \ldots} o_m)$

$$P(O) = \sum_S P(O, S) \qquad \text{Marginalization}$$

Computing $P(o_0 o_1 o_2 o_3 \ldots o_m)$

$$
\begin{aligned}
P(O,S) &= P(S)P(O \mid S) \\
&= P(S_0 S_1 S_2 \ldots S_{m+1})P(O_0 O_1 O_2 \ldots O_m \mid S) \\
&= P(S_0).P(S_1 \mid S_0).P(S_2 \mid S_1)\ldots.P(S_{m+1} \mid S_m). \\
&\qquad P(O_0 \mid S_0).P(O_1 \mid S_1)\ldots..P(O_m \mid S_m) \\
&= P(S_0)[P(O_0 \mid S_0).P(S_1 \mid S_0)]\ldots..[P(O_m \mid S_m).P(S_{m+1} \mid S_m)]
\end{aligned}
$$

Forward and Backward Probability Calculation

Forward probability *F(k,i)*

- Define *F(k,i)*= Probability of being in state S_i having seen $o_0 o_1 o_2 \ldots o_k$
- $F(k,i)=P(o_0 o_1 o_2 \ldots o_k, S_i)$
- With *m* as the length of the observed sequence
- $P(observed\ sequence)=P(o_0 o_1 o_2 .. o_m)$

  $=\Sigma_{p=0,N} P(o_0 o_1 o_2 .. o_m , S_p)$

  $=\Sigma_{p=0,N} F(m , p)$ {N +1 is the no of states}

Forward probability (contd.)

$F(k, q)$

$= P(o_0 o_1 o_2 .. o_k , S_q)$

$= P(o_0 o_1 o_2 .. o_k , S_q)$

$= P(o_0 o_1 o_2 .. o_{k-1} , o_k , S_q)$

$= \Sigma_{p=0,N} P(o_0 o_1 o_2 .. o_{k-1} , S_p , o_k , S_q)$

$= \Sigma_{p=0,N} P(o_0 o_1 o_2 .. o_{k-1} , S_p).$
$\quad P(o_k , S_q | o_0 o_1 o_2 .. o_{k-1} , S_p)$

$= \Sigma_{p=0,N} F(k-1,p). P(o_k , S_q | S_p)$

$= \Sigma_{p=0,N} F(k-1,p). P(S_p \xrightarrow{o_k} S_q)$

$O_0 \quad O_1 \quad O_2 \quad O_3 \ldots \quad O_k \quad O_{k+1} \quad \ldots \quad O_{m-1} \quad O_m$

$S_0 \rightarrow S_1 \quad S_2 \quad S_3 \quad \ldots \quad S_p \rightarrow S_q \quad \ldots \quad S_m \quad S_{final}$

Backward probability $B(k,i)$

- Define $B(k,i)$= Probability of seeing $o_k o_{k+1} o_{k+2} \ldots o_m$ given that the state was S_i
- $B(k,i) = P(o_k o_{k+1} o_{k+2} \ldots o_m \mid S_i)$
- With m as the length of the whole observed sequence
- $P(observed\ sequence) = P(o_0 o_1 o_2 .. o_m)$

  $= P(o_0 o_1 o_2 .. o_m / S_0)$

  $= B(0,0)$

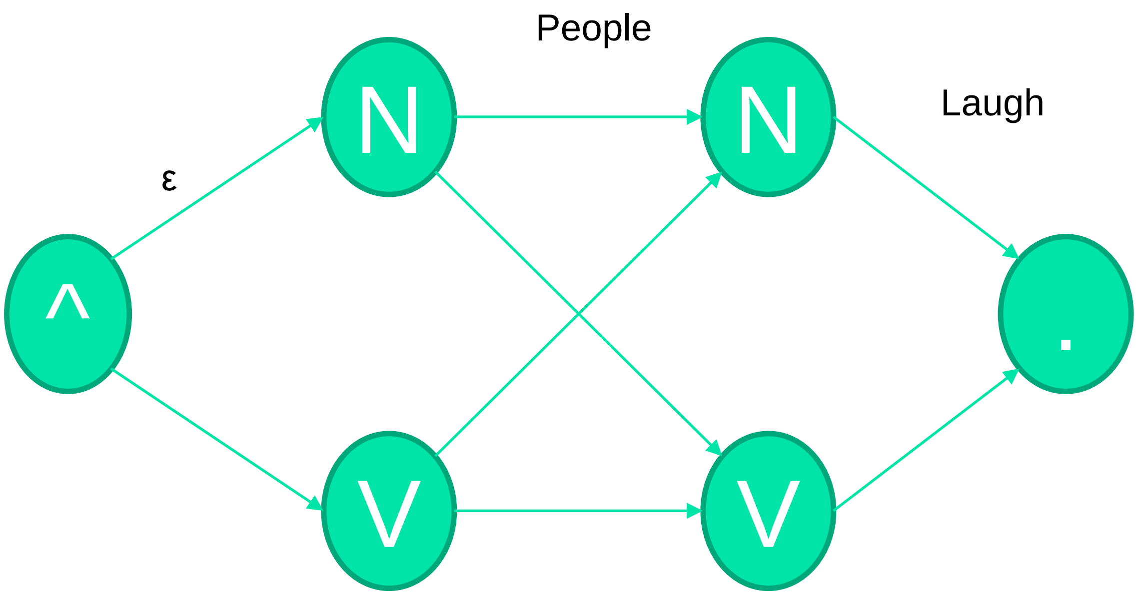
Backward probability (contd.)

$$
\begin{aligned}
&B(k, p) \\
&= P(o_k o_{k+1} o_{k+2} \ldots o_m \mid S_p) \\
&= P(o_{k+1} o_{k+2} \ldots o_m , o_k / S_p) \\
&= \Sigma_{q=0,N}\, P(o_{k+1} o_{k+2} \ldots o_m , o_k , S_q / S_p) \\
&= \Sigma_{q=0,N}\, P(o_k , S_q / S_p)\, P(o_{k+1} o_{k+2} \ldots o_m / o_k , S_q , S_p) \\
&= \Sigma_{q=0,N}\, P(o_{k+1} o_{k+2} \ldots o_m{}_{o_k} / S_q).\, P(o_k , S_q / S_p) \\
&= \Sigma_{q=0,N}\, B(k+1, q).\, P(S_p \xrightarrow{o_k} S_q)
\end{aligned}
$$

$$
\begin{array}{ccccccccccc}
 & O_0 & O_1 & O_2 & O_3 \ldots & & O_k & O_{k+1} & \ldots\ O_{m-1} & O_m & \\
S_0 \rightarrow & S_1 & S_2 & S_3 & \ldots & S_p \rightarrow & S_q & \ldots & & S_m & S_{final}
\end{array}
$$

How Forward Probability Works

- Goal of Forward Probability: To find P(O) [the probability of Observation Sequence].
- E.g. ^ People laugh .



Translation and Lexical Probability Tables

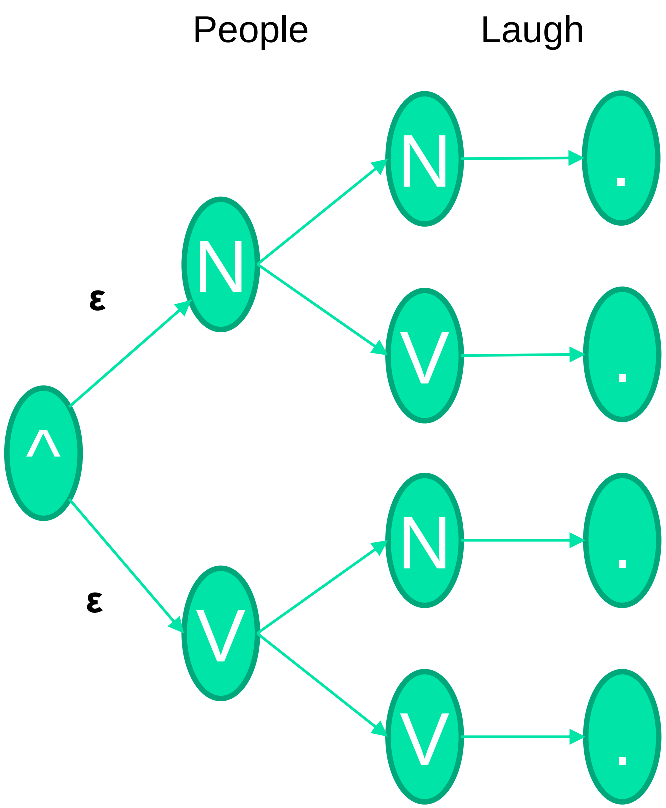
	^	N	V	.
^	0	0.7	0.3	0
N	0	0.2	0.6	0.2
V	0	0.6	0.2	0.2
.	1	0	0	0

	ε	People	Laugh
^	1	0	0
N	0	0.8	0.2
V	0	0.1	0.9
.	1	0	0

Inefficient Computation:

$$P(O) = \sum_{S} P(O,S) = \prod P(S_i \xrightarrow{O_j} S_{i+1})$$

Computation in various paths of the Tree



ε **People** **Laugh**

Path 1: ^ N N .

P(Path1) = (1.0x0.7)x(0.8x0.2)x(0.2x0.2)

ε **People** **Laugh**

Path 2: ^ N V .

P(Path2) = (1.0x0.7)x(0.8x0.6)x(0.9x0.2)

ε **People** **Laugh**

Path 3: ^ V N .

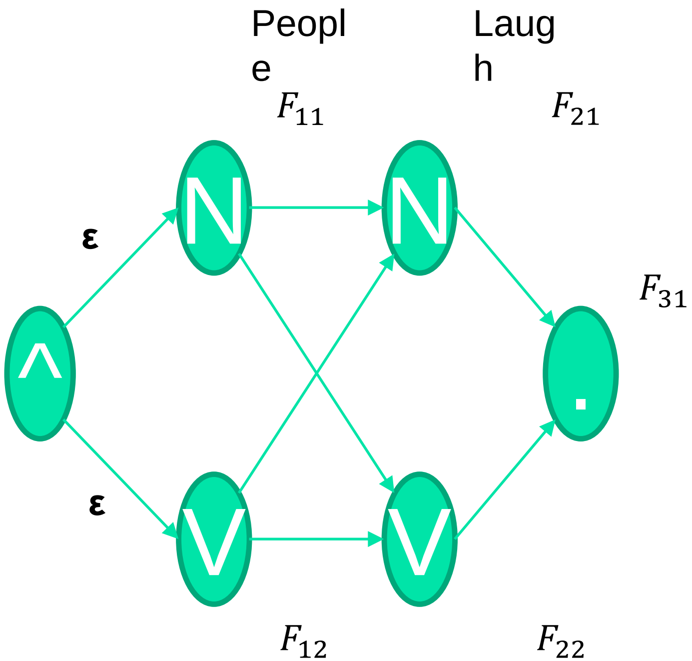
P(Path3) = (1.0x0.3)x(0.1x0.6)x(0.2x0.2)

ε **People** **Laugh**

Path 4: ^ V V .

P(Path4) = (1.0x0.3)x(0.1x0.2)x(0.9x0.2)

Computations on the Trellis



$F = accumulated\ F\ x\ output\ probability\ x\ transition\ \ probability$

F_{11} = 0.7x1.0

F_{12} = 0.3x1.0

F_{21} = F_{11} x (0.2x0.3) + F_{12} x (0.6x0.1)

F_{22} = F_{11} x (0.6x0.8) + F_{12} x (0.2x0.1)

F_{31} = F_{21} x (0.2x0.2) + F_{22} x (0.2x0.9)

Number of Multiplications

Tree

- Each path has 5 multiplications + 1 addition.
- There are 4 paths in the tree.
- Therefore, total of 20 multiplications and 3 additions.

Trellis

- F_{11}, -> 1 multiplication
- F_{12}, -> 1 multiplication
- F_{21} = F_{11} x (1 mult) + F_{12} x (1 mult)

  = 4 multiplications + 1 addition
- Similarly, for F_{22} and F_{31}, 4 multiplications and 1 addition each.
- So, total of 14 multiplications and 3 additions.

Complexity

Let |S| = #States

And |O| = Observation length - |{^, .}|

- Stage 1 of Trellis: |S| multiplications
- Stage 2 of Trellis: |S| nodes; each node needs computation over |S| arcs.
  - Each Arc = 1 multiplication
  - Accumulated F = 1 more multiplication
  - Total $2|S|^2$ multiplications
- Same for each stage before reading ‘ . ’
- At final stage (‘ . ‘) -> 2|S| multiplications
- Therefore, total multiplications = $|S| + 2|S|^2 (|O| - 1) + 2|S|$

Summary : Forward Algorithm

1. Accumulate F over each stage of trellis.
2. Take sum of F values multiplied by $P(S_i \xrightarrow{O_j} S_{i+1})$.
3. Complexity = $|S| + 2|S|^2 (|O| - 1) + 2|S|$

   $= 2|S|^2 |O| - 2|S|^2 + 3|S|$

   $= O(|S|^2 . |O|)$

i.e., linear in the length of input and quadratic in number of states.

Exercise

1. Backward Probability
   a) Derive Backward Algorithm.
   b) Compute its complexity.
2. Express P(O) in terms of both Forward and Backward probability.

Maximum Likelihood Estimate

Explanation with coin tossing

- A coin is tossed 100 times, Head appears 40 times
- P(H)= 0.4
- Why?
- Because of maximum likelihood

N tosses, K Heads, parameter *P(H)=p*

- Construct Maximum Likelihood Expression
- Take log likelihood and take derivative
- Equate to 0 and Get *p*

$$L = p^K (1-p)^{N-K}$$

$$\Rightarrow LL = \log(L) = K \log p + (N-K)\log(1-p)$$

$$\Rightarrow \frac{d(LL)}{dp} = \frac{K}{p} - \frac{N-K}{1-p}$$

$$\Rightarrow \frac{d(LL)}{dp} = 0 \quad gives \quad p = \frac{K}{N}$$

Exercise

- Following the process for finding the probability of Head from N tosses of coin yielding K Heads, prove that the transition probabilities can be found from MLE
- Most important: get the likelihood expression
- Use chapter 2 of the book
  - Pushpak Bhattacharyya: Machine translation, CRC Press, Taylor & Francis Group, Boca Raton, USA, 2015, ISBN: 978-1-4398-9718-8

Computing P(.) values from annotated data

Let us suppose annotated corpus has the following sentence

I	**have**	**a**	**brown**	**bag**	**.**
PRN	**VB**	**DT**	**JJ**	**NN**	**.**

$$P(NN \mid JJ) = \frac{Number\_of\_times\_JJ\_followed\_by\_NN}{Number\_of\_times\_JJ\_appeared}$$

$$P(Brown \mid JJ) = \frac{Number\_of\_times\_Brown\_tagged\_as\_JJ}{Number\_of\_times\_JJ\_appeared}$$

Expectation Maximization

From

Pushpak Bhattacharyya, *Machine Translation*, CRC Press, 2015

Maximum Likelihood of Observations

- ***Situation 1: Throw of a Single Coin***
- The parameter is the probability *p* of getting heads in a single toss. Let *N* be the number of tosses. Then the observation *X* and the data or observation likelihood *D* respectively are:

$$X :< x_1, x_2, x_3, \ldots, x_{N-1}, x_N >$$

$$D = \prod_{i=1}^{N} p^{x_i} (1-p)^{1-x_i}, \text{ s.t. } \mathrm{x_i} = 1 \text{ or } 0, \text{ and } 0 \le \mathrm{p} \le 1$$

where x_i is an indicator variable assuming values 1 or 0 depending on the *ith* observation being heads or tail. Since there are *N* identically and independently distributed (*i.i.d.*) observations, *D* is the product of probabilities of individual observations each of which is a Bernoulli trial.

Single coin

Since exponents are difficult to manipulate mathematically, we take log of *D*, also called log likelihood of data, and maximize with regard to *p*. This yields

$$p = \frac{\sum_{i=1}^{N} x_i}{N} = \frac{M}{N};\ M = \# Heads, N = \# tosses$$

Throw of 2 coins

- Three parameters: probabilities p_1 and p_2 of heads of the two coins and the probability p of choosing the first coin (automatically, $1-p$ is the probability of choosing the second coin).
- N tosses and observations of heads and tails. Only, we do not know which observation comes from which coin.
- Indicator variable z_i is introduced to capture coin choice ($z_i=1$ if coin 1 is chosen, else 0). This variable is hidden, *i.e.*, we do not know its values.
- However, without it the likelihood expression would have been very cumbersome.

Data Likelihood

Data Likelihood,

$D \quad = P_{<p1,p2,p>}(X)=P_\theta(X),\ \theta=<p,p_1,p_2>$

$\quad\ = \Sigma_Z P_\theta(X,Z))$

$$X :< x_1, x_2, x_3, \ldots, x_{N-1}, x_N >$$

$$Z :< z_1, z_2, z_3, \ldots, z_{N-1}, z_N >$$

$$P_\theta(X, Z) = \prod_{i=1}^{N} \left[\left(p p_1^{\,x_i} (1 - p_1)^{1-x_i} \right)^{z_i} \left((1 - p) p_2^{\,x_i} (1 - p_2)^{1-x_i} \right)^{1-z_i} \right],$$

$$\text{s.t. } \mathrm{z_i}, \mathrm{x_i} = 1 \text{ or } 0, \text{ and } 0 \le \mathrm{p}, \mathrm{p_1}, \mathrm{p_2} \le 1$$

Invoke Jensen Inequality

We would like to work with *log*$P_\theta(X)$. However, there will be a Σ inside *log*. Fortunately, *log* is a concave function, so that

$$\log\left(\sum_{i=1}^{K} \lambda_i y_i\right) \geq \left(\sum_{i=1}^{K} \lambda_i \log(y_i)\right); \sum_{i=1}^{K} \lambda_i = 1$$

Log likelihood of Data

$LL(D)$= log likelihood of data

$$= log(P_\theta(X)) = log(\Sigma_Z P_\theta(X,Z))$$

$$= log[\Sigma_Z \lambda_Z (P_\theta(X,Z)/\lambda_Z)];\ \Sigma_Z \lambda_Z = 1$$

$$>= \Sigma_Z [\lambda_Z log[\ (P_\theta(X,Z)/\lambda_Z)]$$

After a number of intricate mathematical steps

$LL(D) >= E_{Z|X,\theta} log(P_\theta(X,Z))$, where $E(.)$ is the expectation function; note that the expectation is conditional on X.

Expectation of log likelihood

$$E_{Z|X}[\log(P_\theta(X,Z)]$$

$$= E_{Z|X}\left[\log\prod_{i=1}^{N}\left[\left(pp_1{}^{x_i}(1-p_1)^{1-x_i}\right)^{z_i}\left((1-p)p_2{}^{x_i}(1-p_2)^{1-x_i}\right)^{1-z_i}\right],\right]$$

$$= E_{Z|X}\left[\begin{array}{l}\sum_{i=}^{N} z_i\left(\log p + x_i \log p_1 + (1-x_i)\log(1-p_1)\right) + \\ (1-z_i)\left(\log(1-p) + x_i \log p_2 + (1-x_i)\log(1-p_2)\right)\end{array}\right]$$

$$= \sum_{i=1}^{N}\left[\begin{array}{l}E(z_i \mid x_i)\left(\log p + x_i \log p_1 + (1-x_i)\log(1-p_1)\right) + \\ (1-E(z_i \mid x_i))\left(\log(1-p) + x_i \log p_2 + (1-x_i)\log(1-p_2)\right)\end{array}\right]$$

s.t. $z_i, x_i = 1$ or 0, and $0 \le p, p_1, p_2 \le 1$

Derivation of E and M steps for 2 coin problem (1/2)- M step

Take partial derivative of $E_{Z|X,\theta}(.)$ (prev. slide) wrt p, p_1, p_2 and equate to 0.

$$p = \frac{\sum_{i=1}^{N} E(z_i \mid x_i)}{N}$$

$$p_1 = \frac{\sum_{i=1}^{N} E(z_i \mid x_i)x_i}{\sum_{i=1}^{N} E(z_i \mid x_i)}$$

$$p_2 = \frac{M - \sum_{i=1}^{N} E(z_i \mid x_i)x_i}{N - \sum_{i=1}^{N} E(z_i \mid x_i)}; M = \#Heads, N = \#tosses$$

Derivation of E and M steps for 2 coin problem (2/2)- E step

$E(z_i|x_i) = 1.P(z_i=1|x_i)+0.P(z_i=0|x_i)$

$=P(z_i=1|x_i)$

$$P(z_i = 1 \mid x_i) = \frac{P(z_i = 1, x_i)}{P(x_i)}$$

$$= \frac{p p_1^{x_i} (1 - p_1)^{1 - x_i}}{P(x_i, z_i = 1) + P(x_i, z_i = 0)}$$

$$= \frac{p p_1^{x_i} (1 - p_1)^{1 - x_i}}{p p_1^{x_i} (1 - p_1)^{1 - x_i} + (1 - p) p_2^{x_i} (1 - p_2)^{1 - x_i}}$$