Chapter 2. Foundation of Mathematical Algorithm

Jingbo Xia
College of Informatics, HZAU

Outline

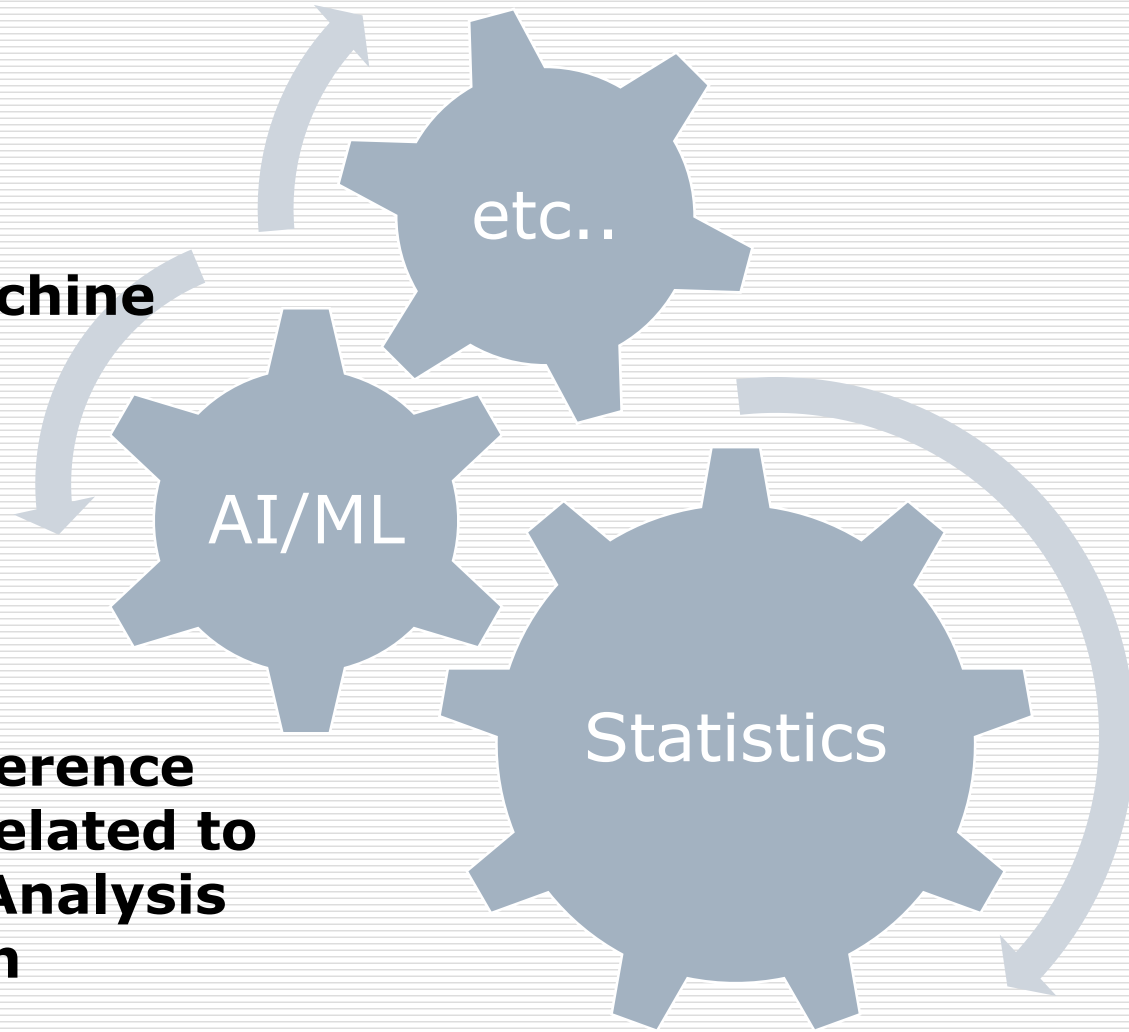
- Intro of Mathematical Modelling Idea of NLP problem
- The First Main Idea: Statistic Based Modelling
- The Second Main Idea: Machine Learning Modelling
- Metric for Evaluation

Outline

- Intro of Mathematical Modelling Idea of NLP problem
- The First Main Idea: Statistic Based Modelling
- The Second Main Idea: Machine Learning Modelling
- Metric for Evaluation

Main branches of Mathematical modelling used in NLP

- ❑Neural Networks
- ❑Decision Tree
- ❑Support Vector Machine
- ❑Deep Learning
- ❑…



- ❑ Bayesian Inference
- ❑ Regression related to Association Analysis
- ❑ Markov Chain
- ❑ MCMC
- ❑ …

Outline

- Intro of Mathematical Modelling Idea of NLP problem
- The First Main Idea: Statistic Based Modelling
- The Second Main Idea: Machine Learning Modelling
- Metric for Evaluation

There are a couple of Statistical methods:

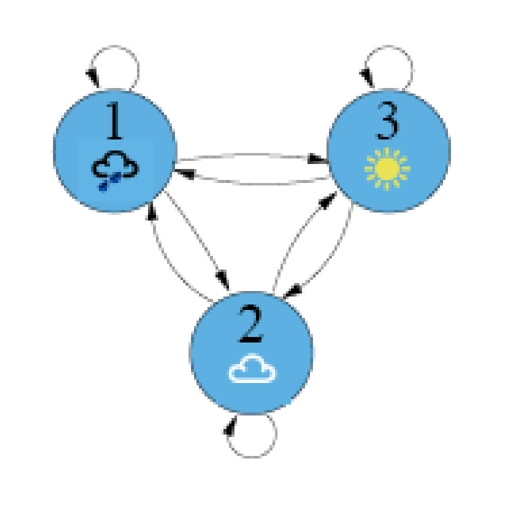
- Bayesian Inference
- Regression related to Association Analysis
- Markov Chain
- MCMC

…

A Mini course of HMM

Hidden Markov models

- Probability fundamentals
- Markov models
- Hidden Markov models
  - Likelihood calculation



Probability fundamentals

- Normalization
  - discrete and continuous
- Independent events
  - joint probability
- Dependent events
  - conditional probability
- Bayes' theorem
  - posterior probability
- Marginalization
  - discrete and continuous

Normalisation

Discrete: probability of all possibilities sums to one:

$$\sum_{\text{all } X} P(X) = 1. \qquad (1)$$

Continuous: integral over entire probability density function (pdf) comes to one:

$$\int_{-\infty}^{\infty} p(x)\, dx = 1. \qquad (2)$$

Joint probability

The joint probability that two independent events occur is the product of their individual probabilities:

$$P(A, B) = P(A)\, P(B). \qquad (3)$$

Conditional probability

If two events are dependent, we need to determine their conditional probabilities. The joint probability is now

$$P(A,B) = P(A)\,P(B|A), \qquad (4)$$

where $P(B|A)$ is the probability of event B given that A occurred; conversely, taking the events the other way

$$P(A,B) = P(A|B)\,P(B). \qquad (5)$$

Bayes' theorem

Equating the RHS of eqs. 4 and 5 gives

$$P(B|A) = \frac{P(A|B)\,P(B)}{P(A)}. \qquad (6)$$

For example, in a word recognition application we have

$$P(w|\mathcal{O}) = \frac{p(\mathcal{O}|w)\,P(w)}{p(\mathcal{O})}, \qquad (7)$$

which can be interpreted as

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}. \qquad (8)$$

- The posterior probability is used to make Bayesian inferences;
- The conditional likelihood describes how likely the data were for a given class;
- The prior allows us to incorporate other forms of knowledge into our decision (like a language model);
- The evidence acts as a normalization factor and is often discarded in practice (as it is the same for all classes).

Marginalization

Discrete: probability of event B, which depends on A, is the sum over A of all joint probabilities:

$$P(B) = \sum_{\text{all } A} P(A, B) = \sum_{\text{all } A} P(B|A)\, P(A). \qquad (9)$$

Continuous: similarly, the nuisance factor x can be eliminated from its joint pdf with *y*:

$$p(y) = \int_{-\infty}^{\infty} p(x, y)\, dx = \int_{-\infty}^{\infty} p(y|x) p(x)\, dx. \qquad (10)$$

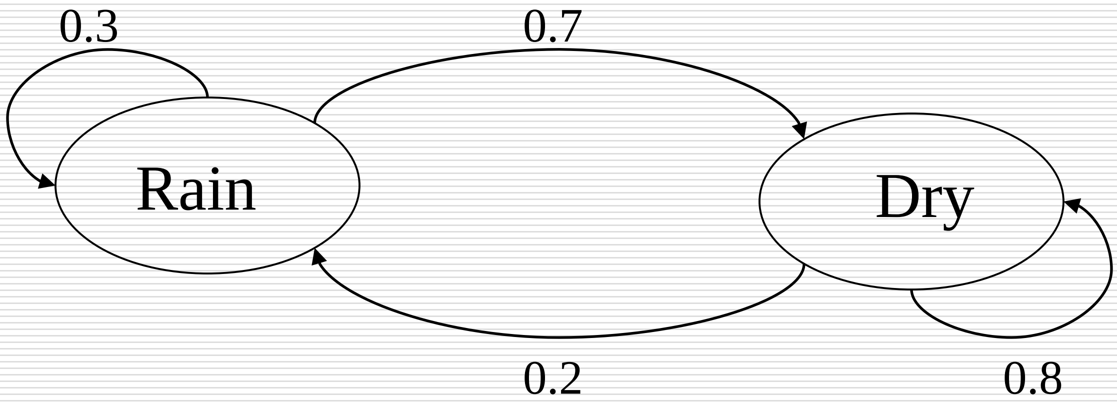
Introduction to Markov Models

- Set of states: $\{s_1, s_2, \ldots, s_N\}$
- Process moves from one state to another generating a sequence of states : $s_{i1}, s_{i2}, \ldots, s_{ik}, \ldots$
- Markov chain property: probability of each subsequent state **depends only on what was the previous state**:

$$P(s_{ik} \mid s_{i1}, s_{i2}, \ldots, s_{ik-1}) = P(s_{ik} \mid s_{ik-1})$$

- To define Markov model, the following probabilities have to be specified: **transition probabilities** $a_{ij} = P(s_i \mid s_j)$ and **initial probabilities** $\pi_i = P(s_i)$

Example of Markov Model



- Two states : 'Rain' and 'Dry'.
- Transition probabilities: P('Rain'|'Rain')=0.3 , P('Dry'|'Rain')=0.7 , P('Rain'|'Dry')=0.2, P('Dry'|'Dry')=0.8
- Initial probabilities: say P('Rain')=0.4 , P('Dry')=0.6 .

Calculation of sequence probability

• By Markov chain property, probability of state sequence can be found by the formula:

$$P(s_{i1}, s_{i2}, \ldots, s_{ik}) = P(s_{ik} \mid s_{i1}, s_{i2}, \ldots, s_{ik-1}) P(s_{i1}, s_{i2}, \ldots, s_{ik-1})$$

$$= P(s_{ik} \mid s_{ik-1}) P(s_{i1}, s_{i2}, \ldots, s_{ik-1}) = \ldots$$

$$= P(s_{ik} \mid s_{ik-1}) P(s_{ik-1} \mid s_{ik-2}) \ldots P(s_{i2} \mid s_{i1}) P(s_{i1})$$

• Suppose we want to calculate a probability of a sequence of states in our example, {'Dry','Dry','Rain',Rain'}.

P({'Dry','Dry','Rain',Rain'}) =

P('Rain'|'Rain') P('Rain'|'Dry') P('Dry'|'Dry') P('Dry')

=0.3*0.2*0.8*0.6

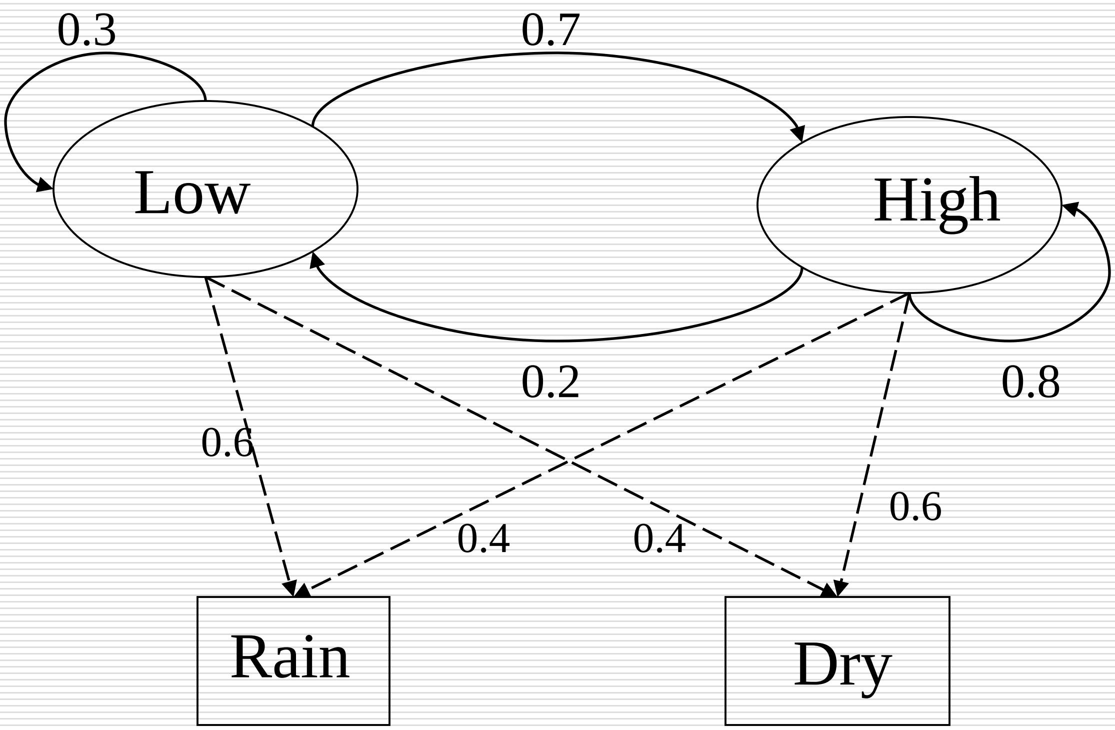
Hidden Markov models.

- Set of states: $\{s_1, s_2, \ldots, s_N\}$
- Process moves from one state to another generating a sequence of states : $s_{i1}, s_{i2}, \ldots, s_{ik}, \ldots$
- Markov chain property: probability of each subsequent state depends only on what was the previous state:

$$P(s_{ik} \mid s_{i1}, s_{i2}, \ldots, s_{ik-1}) = P(s_{ik} \mid s_{ik-1})$$

- States are not visible, but each state randomly generates one of M observations (or visible states) $\{o_1, o_2, \ldots, o_M\}$
- To define hidden Markov model, the following probabilities have to be specified:
  - matrix of transition probabilities A=(a_{ij}), a_{ij}= P(s_i | s_j)
  - matrix of observation probabilities B=($b_i(o_m)$), $b_i(o_m)$ = P(o_m | s_i)
  - initial probabilities π=(π$_i$), π$_i$ = P(s$_i$) . Model is represented by λ=(A, B, π).

Example of Hidden Markov Model



Two states : ‘Low’ and ‘High’ **atmospheric pressure**.

Example of Hidden Markov Model

1. Two states : ‘Low’ and ‘High’ atmospheric pressure.
2. Two observations : ‘Rain’ and ‘Dry’.
3. Transition probabilities: $P(\text{‘Low’}|\text{‘Low’})=0.3$, $P(\text{‘High’}|\text{‘Low’})=0.7$, $P(\text{‘Low’}|\text{‘High’})=0.2$, $P(\text{‘High’}|\text{‘High’})=0.8$
4. Observation probabilities : $P(\text{‘Rain’}|\text{‘Low’})=0.6$, $P(\text{‘Dry’}|\text{‘Low’})=0.4$, $P(\text{‘Rain’}|\text{‘High’})=0.4$, $P(\text{‘Dry’}|\text{‘High’})=0.6$.
5. Initial probabilities: say $P(\text{‘Low’})=0.4$, $P(\text{‘High’})=0.6$.

Calculation of observation sequence probability

```
library(HMM)
hmm = initHMM(c("Low","High"),
c("Rain","Dry"),  c(.4,.6),
      transProbs=matrix(c(.3,.2,.7,.8),2),
    emissionProbs=matrix(c(.6,.4,.4,.6),2))

# Sequence of observations
observations = c("Dry","Rain","Rain")
# Calculate Viterbi path
viterbi = viterbi(hmm,observations)
print(viterbi)
```

```r
library(HMM)
hmm = initHMM(c("Low","High"),
c("Rain","Dry"),  c(.4,.6),
     transProbs=matrix(c(.3,.2,.7,.8),2),

emissionProbs=matrix(c(.6,.4,.4,.6),2))

# Sequence of observations
observations = c("Dry","Rain","Rain")
# Calculate Viterbi path
viterbi = viterbi(hmm,observations)
print(viterbi)
```

```
> print(viterbi)
[1] "High" "High" "High"
```

```
> hmm
$States
[1] "Low"  "High"

$Symbols
[1] "Rain" "Dry"

$startProbs
 Low High
 0.4  0.6

$transProbs
      to
from   Low High
  Low  0.3  0.7
  High 0.2  0.8

$emissionProbs
      symbols
states Rain Dry
  Low   0.6 0.4
  High  0.4 0.6
```

Calculate the path

- Suppose we want to calculate a probability of a sequence of observations in our example, {'Dry','Rain'}.
- Consider all possible hidden state sequences:

P({'Dry','Rain'}) = P({'Dry','Rain'} , {'Low','Low'}) + P({'Dry','Rain'} , {'Low','High'}) + P({'Dry','Rain'} , {'High','Low'}) + P({'Dry','Rain'} , {'High','High'})

where first term is :

P({'Dry','Rain'} , {'Low','Low'})=

P({'Dry','Rain'} | {'Low','Low'}) P({'Low','Low'}) =

P('Dry'|'Low')P('Rain'|'Low') P('Low')P('Low'|'Low)

= 0.4*0.6*0.4*0.3

Three main issues using HMMs

Evaluation problem.

Compute likelihood $P(\mathcal{O}|\lambda)$ a set of observations with an given HMM model, $\lambda = (A, B, \pi)$

Decoding problem.

Decode a state sequence by calculating the most likely path X^* given observation sequence and a HMM model.

Learning problem.

Optimize the template patterns by training the parameters in the models, $\Lambda = \{\lambda\}$

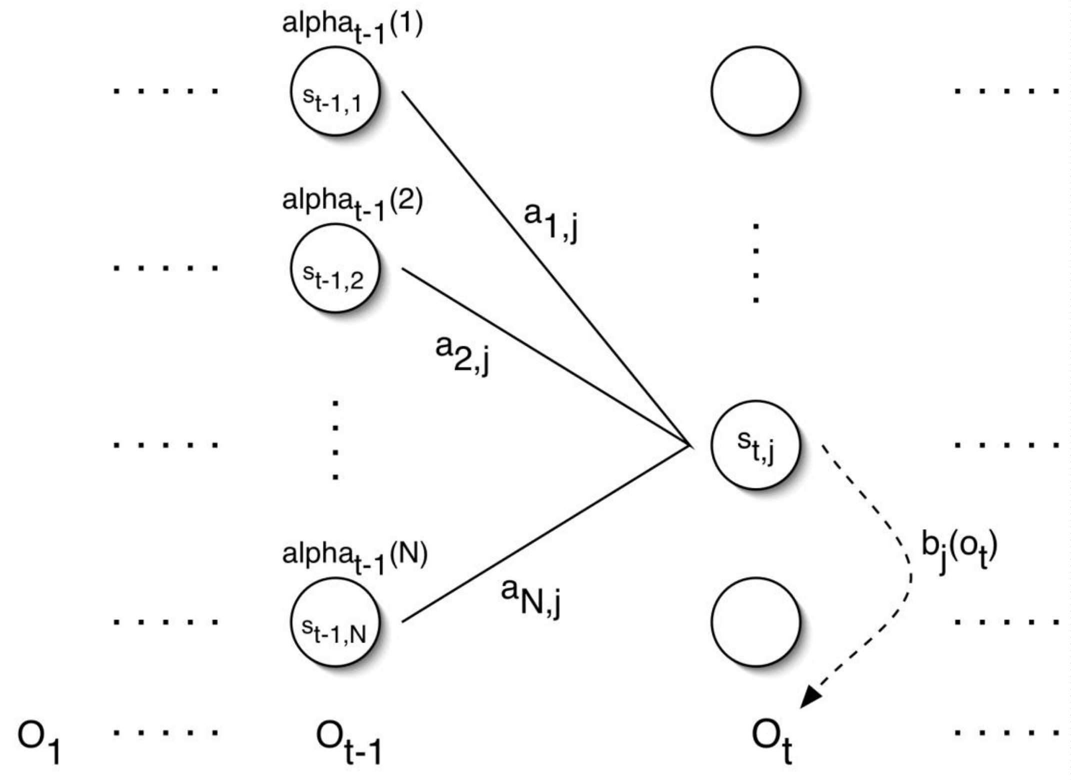
Task 1: Likelihood of an Observation Sequence

- What is $P(O \mid \lambda)$?
- The likelihood of an observation sequence is the sum of the probabilities of all possible state sequences in the HMM.
- Naïve computation is very expensive. Given T observations and N states, there are N^T possible state sequences.
- Even small HMMs, e.g. T=10 and N=10, contain 10 billion different paths
- Solution to this and Task 2 is to use dynamic programming

Forward Probabilities

- What is the probability that, given an HMM, at time *t* the state is *i* and the partial observation $o_1 \ldots o_t$ has been generated?

$$\alpha_t(i) = P(o_1 \ldots o_t, q_t = s_i \mid \lambda) \qquad \alpha_t(i) = P(o_1 \ldots o_t, X_t = i \mid \lambda)$$



$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i)\, a_{ij} \right] b_j(o_t)$$

Forward Algorithm

- Initialization: $$\alpha_1(i) = \pi_i b_i(o_1) \quad 1 \le i \le N$$

- Induction:

$$\alpha_t(j) = \left\lfloor \sum_{i=1}^{N} \alpha_{t-1}(i)\, a_{ij} \right\rfloor b_j(o_t) \quad 2 \le t \le T, 1 \le j \le N$$

- Termination:

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

Forward Algorithm Complexity

- In the naïve approach to solving problem 1 it takes on the order of $2T*N^T$ computations
- The forward algorithm takes on the order of N^2T computations

Task 2: Decoding

- The solution to Task 1 (Evaluation) gives us the sum of all paths through an HMM efficiently.
- For Task 2, we want to find the path with the highest probability.
- We want to find the state sequence $Q=q_1 \ldots q_T$, such that

$$Q = \arg\max_{Q'} P(Q' \mid O, \lambda)$$

Viterbi Algorithm

- Similar to computing the forward probabilities, but instead of summing over transitions from incoming states, compute the maximum
- Forward:

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i)\, a_{ij}\right] b_j(o_t)$$

- Viterbi Recursion:

$$\delta_t(j) = \left[\max_{1\le i\le N} \delta_{t-1}(i)\, a_{ij}\right] b_j(o_t)$$

Viterbi Algorithm

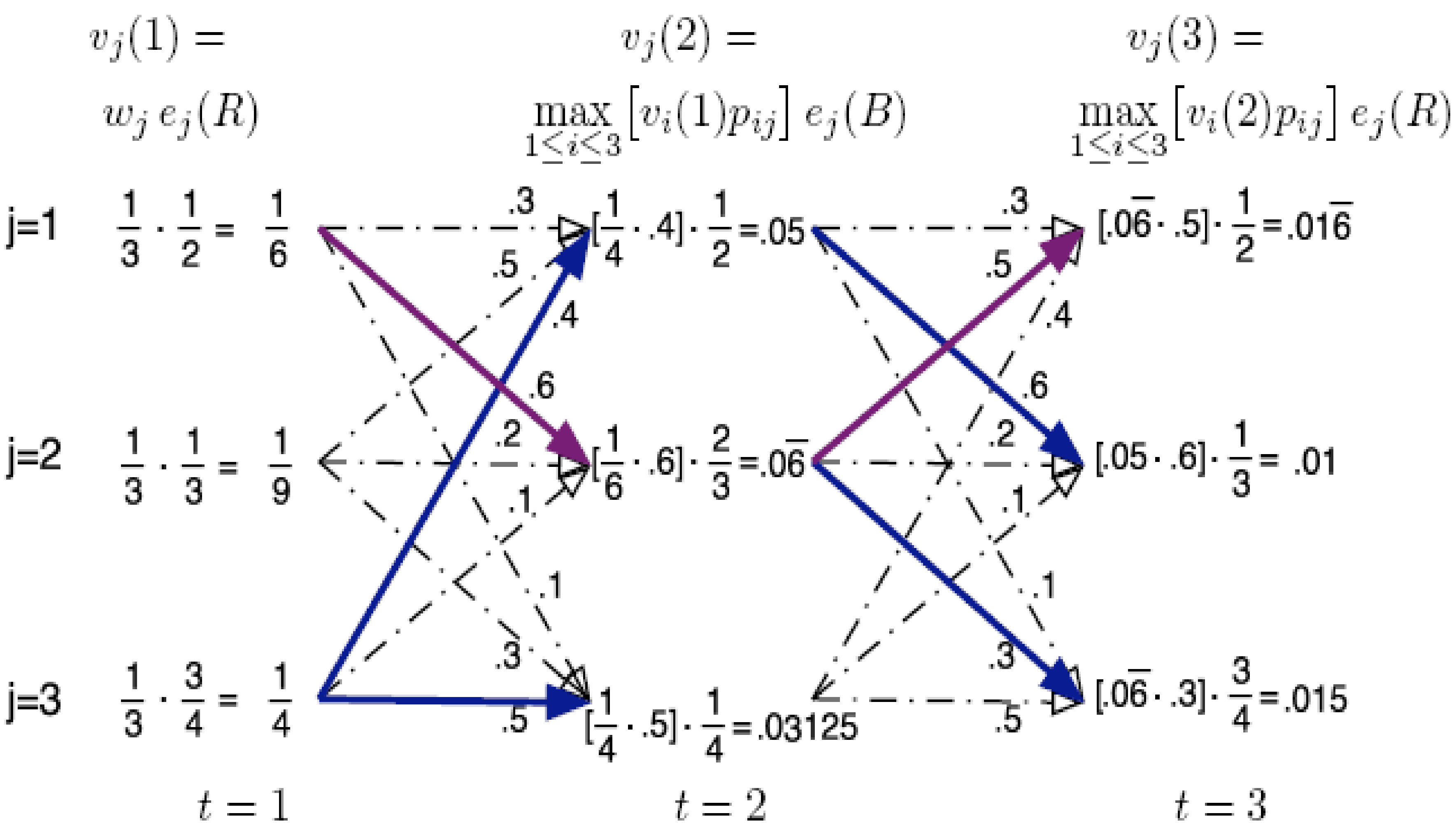
- Initialization: $\delta_1(i) = \pi_i b_i(o_1) \quad 1 \leq i \leq N$
- Induction:

$$\delta_t(j) = \left[\max_{1\leq i\leq N} \delta_{t-1}(i)\, a_{ij}\right] b_j(o_t)$$

$$\psi_t(j) = \left[\arg\max_{1\leq i\leq N} \delta_{t-1}(i)\, a_{ij}\right] \quad 2 \leq t \leq T, 1 \leq j \leq N$$

- Termination: $p^* = \max_{1\leq i\leq N} \delta_T(i) \quad q_T^* = \arg\max_{1\leq i\leq N} \delta_T(i)$
- Read out path: $q_t^* = \psi_{t+1}(q_{t+1}^*) \quad t = T-1,\ldots,1$

Example of Viterbi Algorithm



Outline

- Intro of Mathematical Modelling Idea of NLP problem
- The First Main Idea: Statistic Based Modelling
- **The Second Main Idea: Machine Learning Modelling**
- Metric for Evaluation

There are dozens of ML methods:

- Neural Networks
- Decision Tree
- Support Vector Machine
- Deep Learning
- …

Outline

- Intro of Mathematical Modelling Idea of NLP problem
- The First Main Idea: Statistic Based Modelling
- The Second Main Idea: Machine Learning Modelling
- **Metric for Evaluation**

Different Costs

- In practice, different types of classification errors often incur different costs
- Examples:
  - Terrorist profiling
    "Not a terrorist" correct 99.99% of the time
  - Medical diagnostic tests: does X have leukemia?
  - Loan decisions: approve mortgage for X?
  - Web mining: will X click on this link?
  - Promotional mailing: will X buy the product?
  - …

Different Cost Measures

- The confusion matrix

		Predicted class	
		Yes	No
Actual class	Yes	TP: True positive	FN: False negative
	No	FP: False positive	TN: True negative

- Machine Learning methods usually minimize FP+FN
- TPR (True Positive Rate): TP / (TP + FN)
- FPR (False Positive Rate): FP / (TN + FP)
- Error rate: (FP+FN)/All

Classification with costs

Confusion matrix 1

		Predicted P	Predicted N
Actual	P	20	10 ← FN
	N	30 (FP)	90

Confusion matrix 2

		Predicted P	Predicted N
Actual	P	10	20
	N	15	105

Error rate: 40/150
Accuracy: 110/150
True Positive rate: 20/30
False Positive rate: 30/120

Error rate:?
Accuracy:?
TPR:?
FPR:?

Precision and Recall

- **Precision**: fraction of retrieved docs that are relevant = P(relevant|retrieved)
- **Recall**: fraction of relevant docs that are retrieved

  = P(retrieved|relevant)

	Relevant	Nonrelevant
Retrieved	TP	FP
Not Retrieved	FN	FN

- Precision P = tp/(tp + fp)
- Recall R = tp/(tp + fn)

Should we merely use the accuracy measure for evaluation?

- Given a query, an engine classifies each doc as "Relevant" or "Nonrelevant"
- The **accuracy** of an engine: the fraction of these classifications that are correct
  - (TP + TN) / (TP + FP + FN + TN)
- **Accuracy** is a commonly used evaluation measure in machine learning classification work
- Why is this not a very useful evaluation measure in IR?

Precision/Recall

- You can get high recall (but low precision) by retrieving all docs for all queries!
- Recall is a non-decreasing function of the number of docs retrieved

- In a good system, precision decreases as either the number of docs retrieved or recall increases
  - This is not a theorem, but a result with strong empirical confirmation

A combined measure: *F*

- Combined measure that assesses precision/recall tradeoff is **F measure** (weighted harmonic mean):

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

- People usually use balanced F_1 measure
  - i.e., with $\beta = 1$ or $\alpha = ½$

F-Score

$$F = \frac{2PR}{P + R}$$

Reference

- Speech Recognition and Hidden Markov Models. CPSC4600@UTC/CSE
- CS276, Information Retrieval and Web Search, Pandu Nayak and Prabhakar Raghavan