Generative *vs.* Discriminative Models for Classification

Ko, Youngjoong
Computer Engineering, Dong-A University

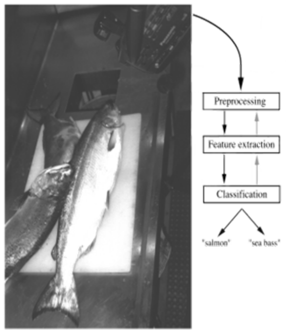
http://web.donga.ac.kr/yjko
youngjoong.ko@gmail.com

Contents

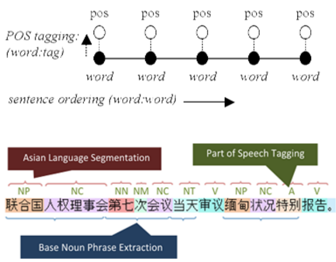
Introduction

- Examples of Pattern Classification

Single Classification:

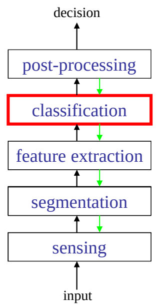


Sequence Classification:



Introduction

- Pattern classification (Duda & Hart)



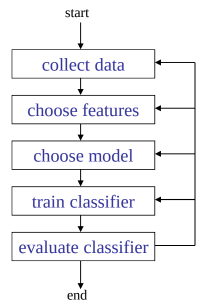
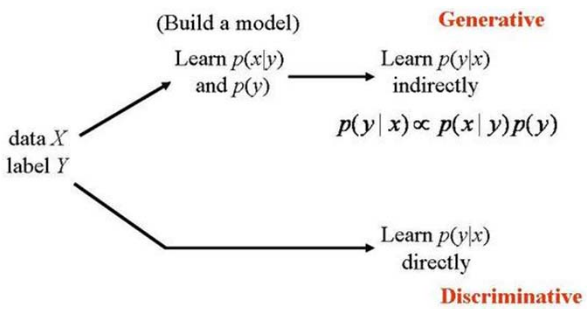


Fig1. The process of the pattern classification system *Fig2. The design cycle of the pattern classification system*

Introduction

❖ Generative vs. Discriminative Models



Introduction

❖ A Simple Example of Generative vs. Discriminative Models

- A form (x,y) : (1,0), (1,0), (2,0), (2,1)
- *p(x,y):* to be transformed into *p(y|x)* by applying Bayes rule and to generate likely *(x,y)* pairs

	y=0	y=1
x=1	1/2	0
x=2	1/4	1/4

- *p(y|x)*: natural distribution for classifying a given example *x* into a class *y*

	y=0	y=1
x=1	1	0
x=2	1/2	1/2

Taxonomy of two Models

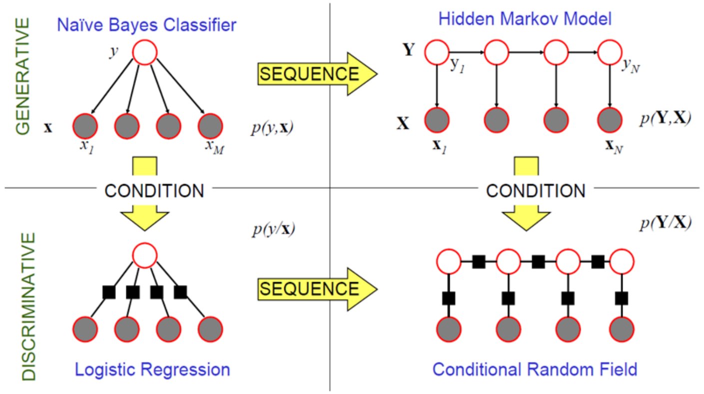
❖ **Generative Models**

- To model class-conditional pdfs and prior probabilities
- "Generative" since sampling can generate synthetic data points
- Popular models:
  - Gaussians, Naïve Bayes, Mixtures of multinomials
  - Mixtures of Gaussians, Mixtures of experts, Hidden Markov Models (HMM)
  - Sigmoidal belief networks, Bayesian networks, Markov random fields

❖ **Discriminative Models**

- Directly estimate posterior probabilities
- No attempt to model underlying probability distributions
- Focus computational resources on given task–better performance
- Popular models:
  - Logistic regression (MEM), SVMs
  - Traditional neural networks, Nearest neighbor
  - Conditional Random Fields (CRF)

Graphical Model Relationship

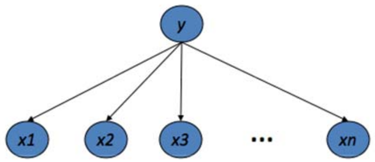


Generative Model: Naïve Bayes

- To learn a bayes classifier, we need to model P(x|*y) and P(y)*
  - We can assume that x_i*'s are conditionally independent given y,*

$$P(x_1, x_2, \ldots, x_n \mid y) = \prod_{i=1}^{n} P(x_i \mid y)$$

  - *This is called the **Naïve Bayes assumption***



Generative Model: Naïve Bayes

- Learning
  - Need to estimate the following probability distributions (via counting)

$p(y)$ Prior distribution of y

$p(x_i \mid y)$ Class conditional distribution of x_i

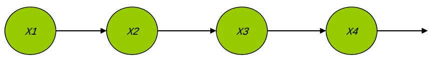
- Predicting
  - Given **x =(x_1, x_2, …, x_d), compute p(y|x)**

$$p(y \mid \mathbf{x}) = \frac{p(y)\,p(\mathbf{x} \mid y)}{p(\mathbf{x})} \propto p(y) \prod_i p(x_i \mid y)$$

  - Apply decision theory to make final prediction of *y*

Markov Chain (Sequence Classification)

- Markov chain
  - Often we want to consider a sequence of random variables that aren't independent, but rather the value of each variable depends on previous elements in the sequence
- Markov Assumption
  - A sequence of states: X_1, X_2, X_3, …
  - The transition from X_{t-1} to X_t depends only on X_{t-1} (Markov Property).
    - The transition probabilities are the same for any t (stationary process)



Markov Model

- **Markov Properties**
  - Limited Horizon:

$$P(X_{t+1} = s_k \mid X_1, \ldots, X_t) = P(X_{t+1} = s_k \mid X_t)$$

Time invariant (stationary): $= P(X_2 = s_k \mid X_1)$

- **Stochastic Transition Matrix**

$$a_{ij} = P(X_{t+1} = s_j \mid X_t = s_i)$$

$$\text{where, } \mathrm{a_{ij}} \ge 0, \forall i,j \text{ and } \sum_{j=1}^{N} a_{ij} = 1, \forall i$$

- **Initial states** $\pi_i = P(X_1 = s_i)$ *where* $\sum_{i=1}^{N} \pi_i = 1$

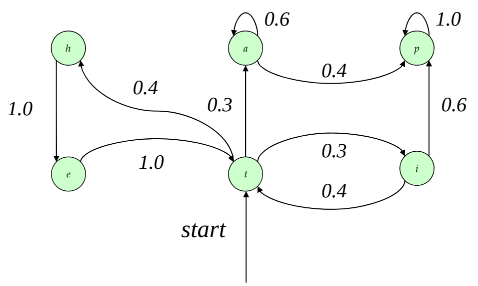
Markov Model

- **Examples**
  - N-gram models in NLP
  - Valid phone sequences in speech recognition
  - Sequences of speech acts in dialog systems
- **Bigram model**
  - Bigram models are rather inaccurate language models.
    - Ex) the word after "a" is much more likely to be "missile" if the word preceding "a" is "launch".
  - The Markov assumption is pretty bad.
  - If we could condition on a few previous words, life gets a bit better:
  - E.g., we could predict "missile" is more likely to follow "launch a" than "saw a".
  - This would require a "second order" Markov model.

Markov Model

$$
\begin{aligned}
P(X_1,\ldots,X_T) &= P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_1,X_2)\ldots P(X_T \mid X_1,\ldots,X_{T-1}) \\
&= P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_2)\ldots P(X_T \mid X_{T-1}) \\
&= \pi_{X_1}\prod_{t=1}^{T-1} a_{X_t X_{t+1}}
\end{aligned}
$$

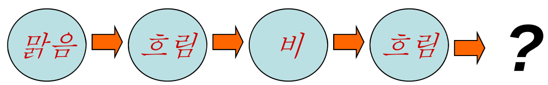
Markov Model



$$
\begin{aligned}
P(t,i,p) &= P(X_1=t)P(X_2=i \mid X_1=t)P(X_3=p \mid X_2=i) \\
&= 1.0 \times 0.3 \times 0.6 = 0.18
\end{aligned}
$$

Markov Model

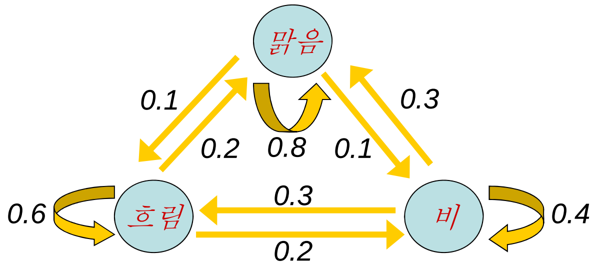
- Markov Model 개념
  - 내일의 날씨는 어떻게 될까요?



Markov Model

❖ Markov Model 개념

➢ 내일의 날씨는 어떻게 될까요?



Markov Model

❖ Markov Model 개념

➢ 내일의 날씨는 어떻게 될까요?

		내일날씨		
		맑음	흐림	비
오늘날씨	맑음	0.8	0.1	0.1
	흐림	0.2	0.6	0.2
	비	0.3	0.3	0.4

Markov Model

❖ Length of the observation sequence : T

❖ Observable states :

➢ 1, 2, …, N

❖ Observed sequence :

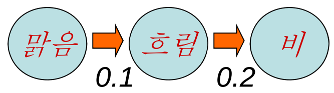
➢ $O_1, O_2, \ldots, O_t, \ldots O_T$

❖ Markov property

➢ $P(O_{t+1} = i \mid O_1, O_2, \ldots, O_{t-1}, O_t) = P(O_{t+1} = i \mid O_t)$

Markov Model

- $P(O_1, O_2, \ldots, O_T)$
  $= P(O_1)P(O_2|O_1)P(O_3|O_1,O_2)\ldots P(O_T|O_1,\ldots,O_{T-1})$
  $= P(O_1)P(O_2|O_1)P(O_3|O_2)\ldots P(O_T|O_{T-1})$



P(맑음,흐림,비)

= P(맑음)P(흐림|맑음)P(비|흐림)

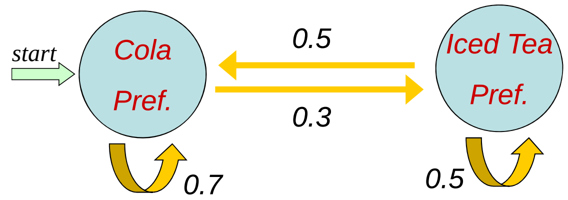
= 1.0 x 0.1 x 0.2 = 0.02

Hidden Markov Models

- Why hidden?
  - You don't know the state sequence that the model passes through, but only some probabilistic function of it.
  - A natural extension to the Markov chain introduces a nondeterministic process that generates output observation symbols in any given state.
- The observation is a probabilistic function of the state
  - new model is known as a ***hidden Markov model***
  - Can be viewed as double embedded stochastic process with an underlying stochastic process not directly observable
- HMM is basically a Markov chain where the output observation is a random variable X generated according to a output probabilistic function associated with each state

Hidden Markov Models

- Hidden Markov Model 개념
  - 이상한 음료수 자판기(Crazy soft drink machine)



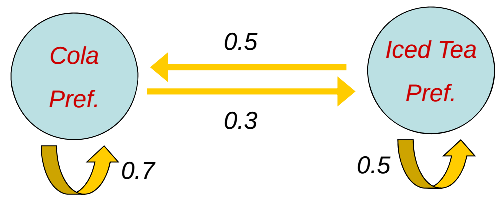
Hidden Markov Model

- Visible Markov Model
  - If, when you put in your coin, the machine always put out a cola if it was in the cola preferring state and an iced tea when it was in the iced tea preferring state
  - But instead, it only has a tendency to do this.
  - So we need symbol emission probabilities for the observations

$$P(O_t = k \mid X_t = s_i, X_{t+1} = s_j) = b_{ijk}$$

  - For this machine, the output is actually independent of s_j

Hidden Markov Model

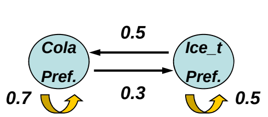


	Cola	*Ice_t*	*Lemon*
Cola Pref.	***0.6***	***0.1***	***0.3***
Ice_t Pref.	***0.1***	***0.7***	***0.2***

	Cola Pref.	*Ice_t Pref.*
Cola Pref.	***0.7***	***0.3***
Ice_t Pref.	***0.5***	***0.5***

Hidden Markov Model

❖ 자판기가 Cola Pref.에서 작동하기 시작할 때, {Lemon, Ice_t} 순서로 음료가 나올 확률은?



	Cola	Ice_t	Lemon
Cola Pref.	0.6	0.1	0.3
Ice_t Pref.	0.1	0.7	0.2

Cola Pref., Cola Pref. : (0.7x0.3)x(0.7x0.1)+
Cola Pref., Ice_t Pref. : (0.7x0.3)x(0.3x0.1)+
Ice_t Pref., Ice_t Pref. : (0.3x0.3)x(0.5x0.7)+
Ice_t Pref., Cola Pref. : (0.3x0.3)x(0.5x0.7) = 0.084

Hidden Markov Model

❖ *Notation for an Hidden Markov Models*

- *T = length of the observation sequence,*
  $\{O_1, O_2, \ldots, O_t, \ldots, O_T\}$ (자판기 동작 횟수)
- *N = number of states in the model (자판기 상태 수)*
- *L = number of observation symbols (자판기 음료 종류)*
- *S = a set of states, {s} (자판기 상태집합)*
  $s_t = i$: state i at time t
- *A = state transition probability matrix (자판기 상태변화)*
  $a_{IJ} = P(s_{t+1} = J \mid s_t = I)$
- *B = Observation probability distribution (음료수 확률분포)*
  $b_J(O_t) = P(O_t \mid s_t = J)$
- π = Initial state distribution (초기 상태 분포): $\pi_i = P(s_1 = i)$
- λ = hidden markov model : $\lambda = P(A, B, \pi)$

Hidden Markov Model

❖ Two assumptions in the first-order hidden Markov model

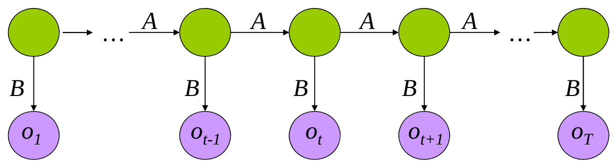
- ***Markov assumption*** for the Markov chain

$$P(s_t \mid s_1^{t-1}) = P(s_t \mid s_{t-1})$$

- ***Output-independence*** assumption
  - Probability that a particular symbol is emitted at time *t* depends only on the state s_t
  - Independent of the past observations

$$P(X_t \mid X_1^{t-1}, s_1^t) = P(X_t \mid s_t)$$

HMM Formalism



{S, L, Π, A, B}

- ❖ ***$\Pi = \{\pi_i\}$ are the initial state probabilities***
- ❖ ***S : $\{s_1 \ldots s_N\}$ are the values for the hidden states***
- ❖ ***L : $\{l_1 \ldots l_M\}$ are the values for the observations***
- ❖ ***A = $\{a_{ij}\}$ are the state transition probabilities***
- ❖ ***B = $\{b_{ik}\}$ are the observation state probabilities***

Hidden Markov Model

❖ If it is not possible to observe the sequence of states of a Markov model, but, only the sequence of emitted alphabets or signals, the model is called HMM

➢ We can guess the best state sequence;

argmax$_S$ P(S | O), where O : the sequence of observed alphabet.

= argmax$_S$ {P(O | S) P(S)} / P(O)

= argmax$_S$ P(O | S) P(S)

Three fundamental questions for HMM

❖ **Given a sequence of observed signals O = {o_1, …, o_T} and model μ =(A, B, π)**

➢ Evaluation problem:

- compute the prob. of observing P(O|μ) this particular signal sequence.

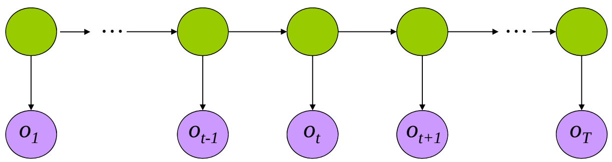
➢ Decoding problem:

- determine the most probable state sequence S = s_1, …, s_T that can give rise to this signal sequence.

➢ Learning or estimation Problem:

- Determine the set of model parameter μ = (A, B,π) maximizing the prob. of this signal sequence P(O| μ).

Evaluation



Given an observation sequence and a model, compute the probability of the observation sequence

$$O = (o_1 \ldots o_T),\ \mu = (A, B, \Pi)$$

Compute $P(O \mid \mu)$

Finding the probability of an observation

❖ Decoding

$$P(O \mid \mu) = \sum_X P(O, X \mid \mu)$$

X=(X_1,…,X_{t+1})

any state sequence

$$= \sum_X P(O \mid X, \mu) P(X \mid \mu)$$

$$P(O \mid X, \mu) = \prod_{t=1}^{T} P(o_t \mid X_t, X_{t+1}, \mu) = b_{X_1 X_2 o_1} b_{X_2 X_3 o_2} \cdots b_{X_T X_{T+1} o_T} = \prod_{t=1}^{T} b_{X_t X_{t+1} o_t}$$

$$P(X \mid \mu) = \pi_{X_1} a_{X_1 X_2} a_{X_2 X_3} \cdots a_{X_T X_{T+1}} = \pi_{X_1} \prod_{t=1}^{T} a_{X_t X_{t+1}}$$

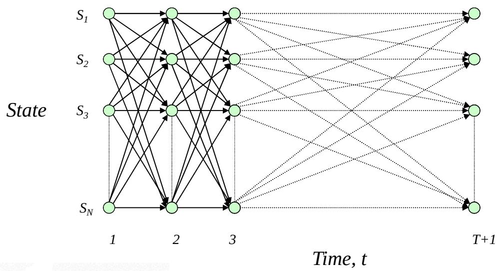
$$= \sum_{X_1 \cdots X_{T+1}} \pi_{X_1} \prod_{t=1}^{T} a_{X_t X_{t+1}} b_{X_t X_{t+1} o_t}$$

Requires multiplications $(2T+1) \cdot N^{T+1}$

➢ But, unfortunately, direct evaluation of the resulting expression is hopelessly inefficient.

Dynamic Programming

- The general technique for the secret to avoiding this complexity
  - We remember partial results rather than re-computing them
    - Ex) chart parsing in computational linguistics
    - Lattice (or Trellis)



Forward procedure

- forward variables

$$\alpha_i(t) = P(o_1 o_2 \cdots o_{t-1}, X_t = i \mid \mu)$$

  - is stored at (s_i, t) in the trellis
  - expresses the total probability of ending up in state s_i at time t
  - is calculated by summing probabilities for all incoming arcs at a trellis node
    - Initialization $\quad \alpha_i(1) = \pi_i, \quad 1 \le i \le N$
    - Induction $\quad \alpha_j(t+1) = \sum_{i=1}^{N} \alpha_i(t) a_{ij} b_{ijo_t}, \quad 1 \le t \le T, 1 \le j \le N$
    - total $\quad P(O \mid \mu) = \sum_{i=1}^{N} \alpha_i(T+1)$

Requires multiplications $2N^2T$

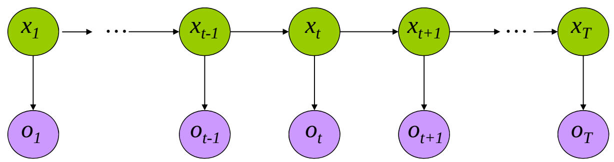
The backward procedure

- Backward variables

$$\beta_i(t) = P(o_t \cdots o_T \mid X_t = i, \mu)$$

  - The total probability of seeing the rest of the observation sequence given that we were in state s_i at time t.
  - Combination of forward and backward probabilities is vital for solving the third problem of parameter re-estimation
    - Initialization $\quad \beta_i(T+1) = 1, \quad 1 \le i \le N$
    - Induction $\quad \beta_i(t) = \sum_{j=1}^{N} a_{ij} b_{ijo_t} \beta_j(t+1), \quad 1 \le t \le T, 1 \le i \le N$
    - total $\quad P(O \mid \mu) = \sum_{i=1}^{N} \pi_i \beta_i(1)$

Evaluation Solution



$$P(O \mid \mu) = \sum_{i=1}^{N} \alpha_i(T)$$ ***Forward Procedure***

$$P(O \mid \mu) = \sum_{i=1}^{N} \pi_i \beta_i(1)$$ ***Backward Procedure***

$$P(O \mid \mu) = \sum_{i=1}^{N} \alpha_i(t) \beta_i(t)$$ ***Combination***

Combining them

$$P(O, X_t = i \mid u) = P(o_1 \ldots o_T, X_t = i \mid u)$$
$$= P(o_1 \ldots o_{t-1}, X_t = i, o_t \ldots o_T \mid u)$$
$$= P(o_1 \ldots o_{t-1}, X_t = i \mid u)$$
$$\times P(o_t \ldots o_T \mid o_1 \ldots o_{t-1}, X_t = i, u)$$
$$= \alpha_i(t)\beta_i(t)$$

Therefore:

$$P(O \mid u) = \sum_{i=1}^{N} \alpha_i(t)\beta_i(t), \; 1 \le t \le T+1$$

Finding the best state sequence

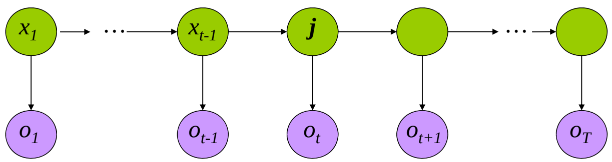
- Choosing the states individually
  - For each t, we would find X_t that maximizes $P(X_t \mid O, \mu)$

$$\gamma_i(t) = P(X_t = i \mid O, \mu)$$
$$= \frac{P(X_t = i, O \mid \mu)}{P(O \mid \mu)}$$
$$= \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)}$$

  - The individually most likely state

$$\hat{X}_t = \arg\max_{1 \le i \le N} \gamma_i(t), \quad 1 \le t \le T+1$$

Decoding solution: finding best sequence



***Viterbi** algorithm* $\arg\max_X P(X \mid O)$

$$\delta_j(t) = \max_{x_1 \ldots x_{t-1}} P(x_1 \ldots x_{t-1}, o_1 \ldots o_{t-1}, x_t = j, o_t)$$

The state sequence which maximizes the probability of seeing the observations to time t-1, landing in state j, and seeing the observation at time t

Viterbi algorithm (Cont.)

1. Initialization

$$\delta_j(1) = \pi_j, \quad 1 \le j \le N$$

2. Induction

$$\delta_j(t+1) = \max_{1 \le i \le N} \delta_i(t) a_{ij} b_{ijo_t}, \quad 1 \le j \le N$$

Store backtrace

$$\psi_j(t+1) = \arg\max_{1 \le i \le N} \delta_i(t) a_{ij} b_{ijo_t}, \quad 1 \le j \le N$$

3. Termination and path readout (by backtracking)

$$\hat{X}_{T+1} = \arg\max_{1 \le i \le N} \delta_i(T+1)$$
$$\hat{X}_t = \psi_{\hat{X}_{t+1}}(t+1)$$
$$P(\hat{X}_t) = \max_{1 \le i \le N} \delta_i(T+1)$$

Learning problem solution

- **The values of the model parameters:** $\mu = (A, B, \pi)$
  - Using Maximum Likelihood Estimation, we want to find the values that maximize:

$$\arg\max_{\mu} P(O_{training} \mid \mu)$$

  - There is no known analytic method to choose to maximize P(O|μ).
  - We can locally maximize it by an iterative hill-climbing algorithm
    - *Baum-Welch* or *Forward-Backward* algorithm
    - It is a special case of the Expectation Maximization (EM) method
      - ✓ Start with the probability of the observation sequence using some model (perhaps randomly chosen model)
      - ✓ We iteratively calculate which state transitions and symbol emissions were probably used the most.
      - ✓ By increasing the probability of those, we can choose a revised model which gives a higher probability to the observation sequence.
      - ✓ This maximization process is often referred to as training the model on training data

Baum-Welch algorithm

- Probability of traversing a certain arc

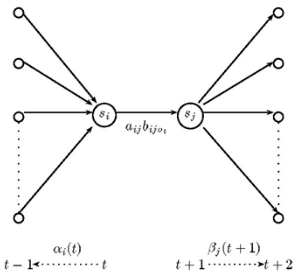


Figure 9.7 The probability of traversing an arc. Given an observation sequence and a model, we can work out the probability that the Markov process went from state s_i to s_j at time t.

Baum-Welch algorithm (Cont.)

- State transition probability
  - Probability of traversing a certain arc at time *t* given observation sequence *O*

$$p_t(i,j) = P(X_t = i, X_{t+1} = j \mid O, \mu) = \frac{P(X_t = i, X_{t+1} = j, O \mid \mu)}{P(O \mid \mu)}$$

$$= \frac{\alpha_i(t) a_{ij} b_{ijo_t} \beta_j(t+1)}{\sum_{m=1}^{N} \alpha_m(t)\beta_m(t)} = \frac{\alpha_i(t) a_{ij} b_{ijo_t} \beta_j(t+1)}{\sum_{m=1}^{N}\sum_{n=1}^{N} \alpha_m(t) a_{mn} b_{mno_t} \beta_n(t+1)}$$

$$\sum_{t=1}^{T} \gamma_i(t) \quad = \text{expected number of transitions from state } i \text{ in } O$$

$$\sum_{t=1}^{T} p_t(i,j) \quad = \text{expected number of transitions from state } i \text{ to } j \text{ in } O$$

  - Note that $\gamma_i(t) = \sum_{j=1}^{N} p_t(i,j)$
  - Expectations(counts), If sum over the time index

Learning problem solution: Baum-Welch algorithm

- Parameter estimation
  - Given an observation sequence, find the model that is most likely to produce that sequence.
  - Given a model and observation sequence, update the model parameters to better fit the observations.

- Re-estimation : from $\mu = (A, B, \Pi)$, derive $\hat{\mu} = (\hat{A}, \hat{B}, \hat{\Pi})$

$$\hat{\pi}_i = \text{expected frequency in state } i \text{ at time } t = 1$$
$$= \gamma_i(1)$$

$$\hat{a}_{ij} = \frac{\text{expected \# of transitions from state } i \text{ to } j}{\text{expected \# of transitions from state } i} = \frac{\sum_{t=1}^{T} p_t(i,j)}{\sum_{t=1}^{T} \gamma_i(t)}$$

$$\hat{b}_{ijk} = \frac{\text{expected \# of transitions from state } i \text{ to } j \text{ with } k \text{ observed}}{\text{expected \# of transitions from state } i \text{ to } j} = \frac{\sum_{\{t: o_t = k, 1 \le t \le T\}} p_t(i,j)}{\sum_{t=1}^{T} p_t(i,j)}$$

HMM Conclusion

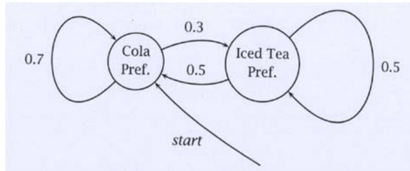
$$\textit{from } \mu = (A, B, \pi), \textit{ we derive } \hat{\mu} = (\hat{A}, \hat{B}, \hat{\pi})$$
$$\textit{As proved by Baum, we have that:}$$
$$P(O \mid \hat{\mu}) \geq P(O \mid \mu)$$

- This is a general property of the EM algorithm
- Iterating through a number of rounds of parameter reestimation will improve our model
- One continues reestimating the parameters until results are no longer improving significantly. But this process of parameter reestimation does not guarantee that we will find the best model.

❖ **HMM Applications**
- POS Tagging
- Speech recognition

HMM Calculation Exercise

❖ The state transition and observation probabilities of the crazy soft drink machine



		CP	IP
Π	CP	1.0	
	IP	0.0	
A	CP	0.7	0.3
	IP	0.5	0.5

	cola	iced tea (ice_t)	lemonade (lem)
CP	0.6	0.1	0.3
IP	0.1	0.7	0.2

HMM Calculation Exercise

❖ Variable Calculations for O=(*lem, ice_t, cola*)

	Output			
	lem	ice_t	cola	
Time (t):	1	2	3	4
$\alpha_{CP}(t)$	1.0	0.21	0.0462	0.021294
$\alpha_{IP}(t)$	0.0	0.09	0.0378	0.010206
$P(o_1 \cdots o_{t-1})$	1.0	0.3	0.084	0.0315
$\beta_{CP}(t)$	0.0315	0.045	0.6	1.0
$\beta_{IP}(t)$	0.029	0.245	0.1	1.0
$P(o_1 \cdots o_T)$	0.0315			
$\gamma_{CP}(t)$	1.0	0.3	0.88	0.676
$\gamma_{IP}(t)$	0.0	0.7	0.12	0.324

HMM Calculation Exercise

❖ Variable Calculations for O=(*lem, ice_t, cola*)

$\widehat{X}_t$	CP	IP	CP	CP
$\delta_{CP}(t)$	1.0	0.21	0.0315	0.01323
$\delta_{IP}(t)$	0.0	0.09	0.0315	0.00567
$\psi_{CP}(t)$		CP	IP	CP
$\psi_{IP}(t)$		CP	IP	CP
$\hat{X}_t$	CP	IP	CP	CP
$P(\hat{X})$	0.019404			

HMM Calculation Exercise

❖ Reestimation from Baum-Welch algorithm

		1			2			3		
		CP	IP	γ_1	CP	IP	γ_2	CP	IP	γ_3
i	CP	0.3	0.7	1.0	0.28	0.02	0.3	0.616	0.264	0.88
	IP	0.0	0.0	0.0	0.6	0.1	0.7	0.06	0.06	0.12

		Original			Reestimated		
Π	CP	1.0			1.0		
	IP	0.0			0.0		
		CP	IP		CP	IP	
A	CP	0.7	0.3		0.5486	0.4514	
	IP	0.5	0.5		0.8049	0.1951	
		cola	ice_t	lem	cola	ice_t	lem
B	CP	0.6	0.1	0.3	0.4037	0.1376	0.4587
	IP	0.1	0.7	0.2	0.1463	0.8537	0.0

Discriminative Model: MEM Background

❖ Maximum Entropy Model (MEM)

- More widely known as multinomial logistic regression
- Belong to the family of classifiers known as the exponential or log-linear classifiers
  - Extract some set of features from the input and combine them linearly
  - Linear regression and logistic regression

$$p(c|x) = \frac{1}{Z}\exp\left(\sum_i w_i f_i\right)$$

Discriminative Model: MEM Background

❖ Linear Regression

- Regression *vs.* Classification
  - Output of regression: real-valued
  - Output of classification: one of a discrete set of classes
- An example for regression
  - Real estate ads: lower prices (fantastic, cute, or charming), higher prices (maple or granite)

Number of vague adjectives	Amount house sold over asking price
4	0
3	\$1000
2	\$1500
2	\$6000
1	\$14000
0	\$18000

Figure 6.17 Some made-up data on the number of vague adjectives (*fantastic, cute, charming*) in a real estate ad, and the amount the house sold for over the asking price.

Discriminative Model: MEM Background

❖ Linear Regression

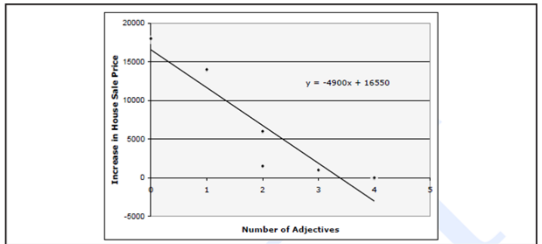


Figure 6.18 A plot of the (made-up) points in Fig. 6.17 and the regression line that best fits them, with the equation $y = -4900x + 16550$.

- Prediction score:

$$\text{price} = w_0 + \sum_{i=1}^{N} w_i \times f_i$$

Discriminative Model: MEM Background

Linear Regression

- General form: $y = \sum_{i=0}^{N} w_i \times f_i$

dot product: $a \cdot b = \sum_{i=1}^{N} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$

$$y = w \cdot f$$

- Learning in linear regression
  - Each observation x would have a feature vector f, and we would train the weight vector w to minimize the prediction error from 1 or 0
  - Sum-squared error

$$\text{cost}(W) = \sum_{j=0}^{M} \left(y_{pred}^{(j)} - y_{obs}^{(j)} \right)^2$$

Discriminative Model: MEM Background

Logistic Regression

- Need to change the real-valued outcome of linear regression into classification (one from a small set of discrete values)
  - Probabilistic classification for binary classification

$$P(y = true|x) = \sum_{i=0}^{N} w_i \times f_i = w \cdot f$$

  - Problem: The left hand lies between 0 and 1 but the right hand lies between -∞ and ∞.
  - Solution: Using odds (ratio of two probabilities) and logit function (log of the odds)

$$\frac{p(y = true)|x}{1 - p(y = true|x)} = w \cdot f$$

$$\ln\left(\frac{p(y = true|x)}{1 - p(y = true|x)}\right) = w \cdot f$$

Discriminative Model: MEM Background

Logistic Regression

- To estimate the logit of the probability rather than the probability

$$\ln\left(\frac{p(y = \text{true}|x)}{1 - p(y = \text{true}|x)}\right) = w \cdot f$$

$$\frac{p(y = \text{true}|x)}{1 - p(y = \text{true}|x)} = e^{w \cdot f}$$

$$p(y = \text{true}|x) = (1 - p(y = \text{true}|x))e^{w \cdot f}$$

$$p(y = \text{true}|x) = e^{w \cdot f} - p(y = \text{true}|x)e^{w \cdot f}$$

$$p(y = \text{true}|x) + p(y = \text{true}|x)e^{w \cdot f} = e^{w \cdot f}$$

$$p(y = \text{true}|x)(1 + e^{w \cdot f}) = e^{w \cdot f}$$

$$p(y = \text{true}|x) = \frac{e^{w \cdot f}}{1 + e^{w \cdot f}}$$

$$p(y = false|x) = \frac{1}{1 + e^{w \cdot f}}$$

$$p(y = \text{true}|x) = \frac{\exp(\sum_{i=0}^{N} w_i f_i)}{1 + \exp(\sum_{i=0}^{N} w_i f_i)}$$

$$p(y = \text{false}|x) = \frac{1}{1 + \exp(\sum_{i=0}^{N} w_i f_i)}$$

Discriminative Model: MEM Background

Logistic Regression

- Logistic function

$$p(y = \text{true}|x) = \frac{e^{w \cdot f}}{1 + e^{w \cdot f}} = \frac{1}{1 + e^{-w \cdot f}}$$

$$\frac{1}{1 + e^{-x}}$$

*

$$p(y = \text{false}|x) = \frac{e^{-w \cdot f}}{1 + e^{-w \cdot f}}$$

  - This function maps values from to lie between 0 and 1.

- Classification of logistic regression

$$p(y = true|x) > p(y = false|x)$$

$$\frac{p(y = true|x)}{p(y = false|x)} > 1$$

$$\frac{p(y = true|x)}{1 - p(y = true|x)} > 1$$

$$e^{w \cdot f} > 1$$

$$w \cdot f > 0$$

  - This is the equation of a hyper-plane (a generalization of a line to N dimension)

$$\sum_{i=0}^{N} w_i f_i > 0$$

Maximum Entropy Modeling

- **Multinomial logistic regression**
  - From binary value (0 or 1) to many discrete values
  - Called MaxEnt in speech and language processing

$$p(c|x) = \frac{1}{Z}\exp\sum_i w_i f_i \qquad Z = \sum_C p(c|x) = \sum_{c'\in C}\exp\left(\sum_{i=0}^{N} w_{c'i} f_i\right)$$

$$p(c|x) = \frac{\exp\left(\sum_{i=0}^{N} w_{ci} f_i\right)}{\sum_{c'\in C}\exp\left(\sum_{i=0}^{N} w_{c'i} f_i\right)}$$

Maximum Entropy Modeling

- **Indicator function**
  - A feature that takes on only the value 0 and 1
  - Ex) POS tagging

Secretariat/NNP is/BEZ expected/VBN to/TO **race**/<u>??</u> tomorrow/

$$f_1(c,x) = \begin{cases} 1 & \text{if } word_i = \text{"race"} \ \& \ c = \text{NN} \\ 0 & \text{otherwise} \end{cases} \qquad f_5(c,x) = \begin{cases} 1 & \text{if } word_i = \text{"race"} \ \& \ c = \text{VB} \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(c,x) = \begin{cases} 1 & \text{if } t_{i-1} = \text{TO} \ \& \ c = \text{VB} \\ 0 & \text{otherwise} \end{cases} \qquad f_6(c,x) = \begin{cases} 1 & \text{if } t_{i-1} = \text{TO} \ \& \ c = \text{NN} \\ 0 & \text{otherwise} \end{cases}$$

$$f_3(c,x) = \begin{cases} 1 & \text{if suffix}(word_i) = \text{"ing"} \ \& \ c = \text{VBG} \\ 0 & \text{otherwise} \end{cases}$$

$$f_4(c,x) = \begin{cases} 1 & \text{if is\_lower\_case}(word_i) \ \& \ c = \text{VB} \\ 0 & \text{otherwise} \end{cases}$$

Maximum Entropy Modeling

- **Indicator function**
  - Ex) POS tagging (Continue)

Secretariat/NNP is/BEZ expected/VBN to/TO **race**/<u>??</u> tomorrow/

		f1	f2	f3	f4	f5	f6
VB	f	0	1	0	1	1	0
VB	w		.8		.01	.1	
NN	f	1	0	0	0	0	1
NN	w	.8					-1.3

Figure 6.19 Some sample feature values and weights for tagging the word *race* in (6.81).

$$P(NN|x) = \frac{e^{.8}e^{-1.3}}{e^{.8}e^{-1.3} + e^{.8}e^{.01}e^{.1}} = .20$$

$$P(VB|x) = \frac{e^{.8}e^{.01}e^{.1}}{e^{.8}e^{-1.3} + e^{.8}e^{.01}e^{.1}} = .80$$

Maximum Entropy Modeling

- **Classification in MaxEnt**
  - A generalization of classification in (Boolean) logistic regression
  - MaxEnt naturally gives us a probability distribution over the classes

$$\hat{c} = \operatorname*{argmax}_{c\in C} P(c|x)$$

  - Any kind of complex feature has to be defined by hand
    - Ex)

$$f_{125}(c,x) = \begin{cases} 1 & \text{if } word_{i-1} = \langle s \rangle \ \& \ \text{isupperfirst}(word_i) \ \& \ c = \text{NNP} \\ 0 & \text{otherwise} \end{cases}$$

Maximum Entropy Modeling

❖ Learning in MaxEnt

- To find the parameters w that maximize the log likelihood of the M training samples

$$\hat{w} = \underset{w}{\operatorname{argmax}} \sum_i \log P(y^{(i)} | x^{(i)})$$

- Important aspect is a kind of smoothing of the weights called regularization
  - To penalize large weights: a MaxEnt model will learn very high weights that overfit the training data

$$\hat{w} = \underset{w}{\operatorname{argmax}} \sum_i \log P(y^{(i)} | x^{(i)}) - \alpha R(w)$$

$$R(W) = \sum_{j=1}^{N} w_j^2$$

$$\hat{w} = \underset{w}{\operatorname{argmax}} \sum_i \log P(y^{(i)} | x^{(i)}) - \alpha \sum_{j=1}^{N} w_j^2$$

Maximum Entropy Modeling

❖ Why We Call It Maximum Entropy

- We want to assign a tag to the word “zzfish”
  - No constraint would be the equiprobable distribution.

NN	JJ	NNS	VB	NNP	IN	MD	UH	SYM	VBG	POS	PRP	CC	CD	...
$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	...

  - Learn only one fact: the set of possible tags is NN, JJ, NNS and VB

$$P(NN) + P(JJ) + P(NNS) + P(VB) = 1$$

NN	JJ	NNS	VB	NNP	IN	MD	UH	SYM	VBG	POS	PRP	CC	CD	...
$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	0	0	0	0	0	0	0	...

  - Learn one more fact: two constraints

$$P(NN) + P(JJ) + P(NNS) + P(VB) = 1$$
$$P(\text{word is } \mathit{zzfish} \text{ and } t_i = \text{NN or } t_i = \text{NNS}) = \frac{8}{10}$$

NN	JJ	NNS	VB	NNP	...
$\frac{4}{10}$	$\frac{1}{10}$	$\frac{4}{10}$	$\frac{1}{10}$	0	...

Maximum Entropy Modeling

❖ Why We Call It Maximum Entropy?

- We want to assign a tag to the word “zzfish”
  - Learn one more fact : three constraints

$$P(NN) + P(JJ) + P(NNS) + P(VB) = 1$$
$$P(\text{word is } \mathit{zzfish} \text{ and } t_i = \text{NN or } t_i = \text{NNS}) = \frac{8}{10}$$
$$P(VB) = \frac{1}{20}$$

NN	JJ	NNS	VB
$\frac{4}{10}$	$\frac{3}{20}$	$\frac{4}{10}$	$\frac{1}{20}$

  - The optimization problem of finding this distribution as follows: (Berger et al. 1996)

"To select a model from a set $\mathcal{C}$ of allowed probability distributions, choose the model $p^* \in \mathcal{C}$ with maximum entropy $H(p)$":

$$p^* = \underset{p \in \mathcal{C}}{\operatorname{argmax}} H(p)$$

$$H(x) = -\sum_x P(x) \log_2 P(x)$$