Learning HMM parameters

Given: m observed sentences, each looks like $O = [o_1, \ldots, o_T], o_t \in V$ (lengths vary).

Find: HMM $\lambda = [A, B, \pi]$

The problem is easy if the hidden states $Q = [q_1, \ldots, q_T], q_t \in S$ are known. This is called "supervised learning". Using MLE, we have

$$a_{ij} = \frac{C(i \to j)}{C(i)} \qquad b_{i,o} = \frac{C(i \to o)}{C(i)} \qquad \pi_i = \frac{C(q_1 = i)}{m}$$

There are hidden states not observed, so can't directly estimate λ .

Ideas: assume a rough estimation of λ and estimate hidden states,

use the estimation to refine λ . Do these two steps iteratively until convergence.

flip a coin for m times and estimate the Pr(head) = $\frac{\#\text{ heads}}{m}$

hard counts = $\sum_{i=1}^{m} \underbrace{\mathbb{I}(\text{flip } i = \text{head})}_{0/1}$

soft counts = $\sum_{i=1}^{m} \underbrace{\Pr(\text{flip}_i = \text{head})}_{\in [0,1]}$

Learning HMM parameters

Estimating A : $\in [0,1]^{N \times N}$, N : # of POS-tags

- We don't know the POS tag sequence definitely and $a_{ij} = \frac{C(i \to j)}{C(i)}$ is not available.
- Rather, we need to find "soft" counts, using probabilities estimated based on $\lambda = [A, B, \pi]$
- Find $\boxed{\text{probability}}$ of co-occurrence of two tags (i, j) at location t :

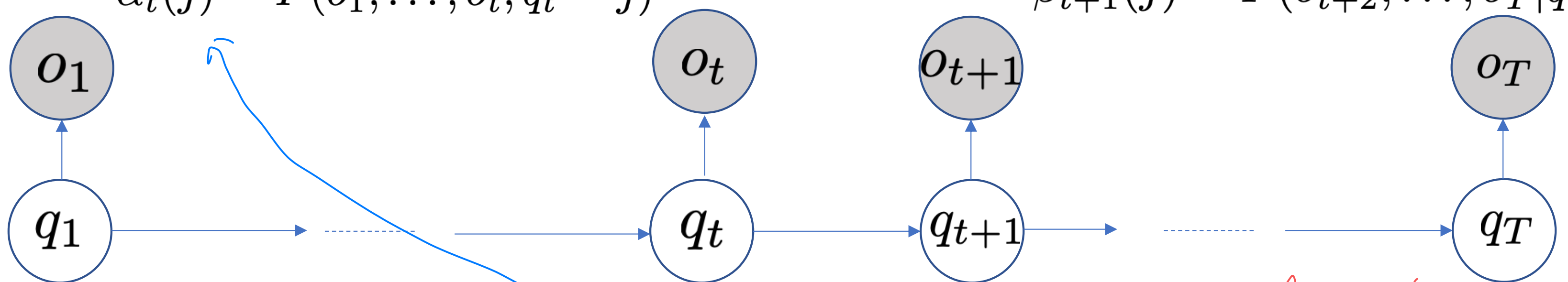
replace the 0/1 counts

$$\xi_t(i,j) = \Pr(q_t = i, q_{t+1} = j | O, \lambda)$$

/ksi/ $= \frac{\Pr(q_t = i, q_{t+1} = j, O | \lambda)}{\Pr(O|\lambda)}$ ← forward / backward

$$\boxed{a_{ij} = \frac{\sum_{k=1}^{m} \sum_{t=1}^{T} \xi_t^{(m)}(i,j)}{\sum_j \sum_{k=1}^{m} \sum_{t=1}^{T} \xi_t^{(m)}(i,j)}}$$

$$\alpha_t(j) = P(o_1, \ldots, o_t, q_t = j) \qquad \beta_{t+1}(j) = P(o_{t+2}, \ldots, o_T | q_{t+1} = j)$$



$\Pr(q_t = i, q_{t+1} = j, O | \lambda) = \boxed{\Pr(q_t = i, o_1, \cdots, o_t | \lambda)} \times \underbrace{\Pr(q_{t+1} = j | q_t = i, \lambda)}_{A_{ij} \text{ (from prev. est.)}}$

$\times \underbrace{\Pr(o_{t+1} | q_{t+1} = j, \lambda)}_{B_j(o_{t+1})} \times \boxed{\Pr(o_{t+2} \cdots o_T | q_{t+1} = j, \lambda)}$

$\beta_1(1) = \Pr(o_2 \cdots o_T | q_1 = 1)$

$\Pr(O|\lambda) = \sum_{i=1}^{N} \Pr(q_1 = i) \Pr(o_1 | q_1 = i) \beta_1(i)$

Learning HMM parameters

$\Pr(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$

Estimating B :

- Similarly, $b_{i,o} = \dfrac{C(i \to o)}{C(i)}$ is not available.
- Find probability of co-occurrence of tag i and word o at location t :
  - find probability $\gamma_t(i) = \Pr(q_t = i | O, \lambda)$

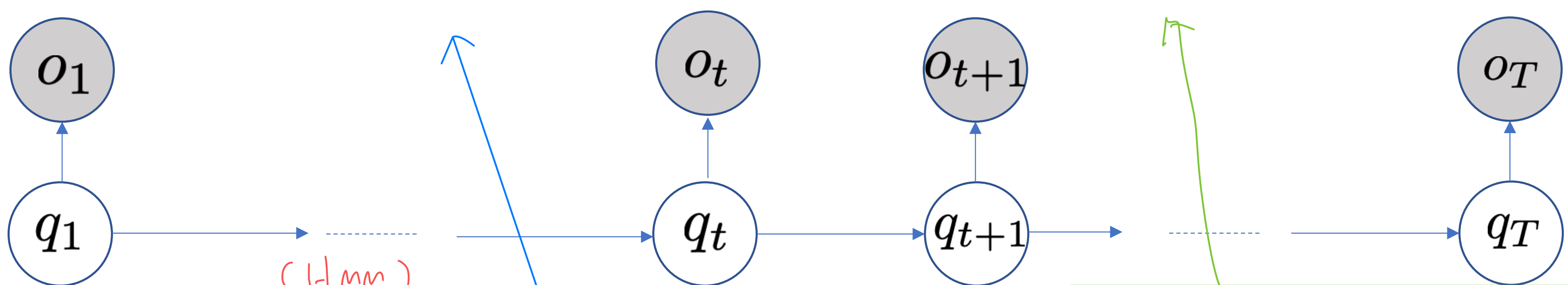
$$= \frac{\Pr(q_t = i, O | \lambda)}{\Pr(O|\lambda) \leftarrow \text{Forward / backward Alg}}$$

$$b_{io} = \frac{\sum_{k=1}^{m} \sum_{t=1, o_t = o}^{T} \gamma_t^{(m)}(i)}{\sum_{k=1}^{m} \sum_{t=1}^{T} \gamma_t^{(m)}(i)}$$

$\begin{bmatrix} \alpha_t(1) \\ \alpha_t(2) \\ \vdots \\ \alpha_t(N) \end{bmatrix}$ $\begin{bmatrix} \beta_t(1) \\ \vdots \\ \beta_t(N) \end{bmatrix}$

$\alpha_t(j) = P(o_1, \ldots, o_t, q_t = j)$ $\quad\quad$ $\beta_t(i) = P(o_{t+1}, \ldots, o_T | q_t = i)$

$\begin{bmatrix} \alpha_T(1) \\ \vdots \\ \alpha_T(N) \end{bmatrix}$



$\Pr(q_t = i, O | \lambda) = \boxed{\Pr(q_t = i, o_1 \cdots o_t | \lambda)} \times \boxed{\Pr(o_{t+1} \cdots o_T | q_t = i, \lambda)}$

Have some labeled seq, then initialize λ using MLE on the labeled data.
(semi-supervised learning)

Learning HMM parameters

EM (expectation-maximization) algorithm: (b/c we have hidden random vars) can't use MLE)

Given: m observed sentences, possibly labeled

Initialize $\lambda = [A, B, \pi]$ either randomly or using the POS-tags if available.

loop until convergence where does it converge to? Typically a local optimum, and you can't find global optimum

E-step: run forward and backward algorithms to find

$$\alpha_t(j) = P(o_1, \ldots, o_t, q_t = j) \qquad \beta_t(i) = P(o_{t+1}, \ldots, o_T | q_t = i)$$

M-step: find

$$\xi_t(i,j) = \Pr(q_t = i, q_{t+1} = j | O, \lambda)$$

$$\gamma_t(i) = \Pr(q_t = i | O, \lambda)$$

$$\boxed{a_{ij} = \frac{\sum_{k=1}^m \sum_{t=1}^T \xi_t^{(m)}(i,j)}{\sum_i \sum_{k=1}^m \sum_{t=1}^T \xi_t^{(m)}(i,j)}} \qquad \boxed{b_{io} = \frac{\sum_{k=1}^m \sum_{t=1, o_t = o}^T \gamma_t^{(m)}(i)}{\sum_{k=1}^m \sum_{t=1}^T \gamma_t^{(m)}(i)}}$$

starting probabilities are left as an exercise.

$\pi(i) = ?$

Learning HMM parameters

A running example of EM.

- Same cheating dealer example. Observe sequence of H/T. Estimate the cheating model
  - transiting between cheating and no cheating,
  - the probability of heads under the two modes.

state 1 $\triangleq$ cheating

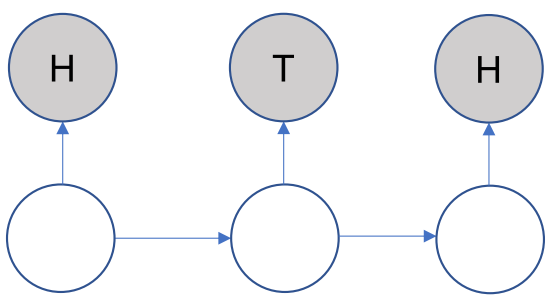
2 $\triangleq$ no cheating

Input: a single sequence

$[o_1\ o_2\ o_3] = [H, T, H]$

Initialize $A = \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix}$, $B = \begin{bmatrix} 1/4 & 3/4 \\ 1/2 & 1/2 \end{bmatrix}$, $\pi = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}$

Loop until convergence;



E-Step: for $t = 1, \dots, 3$,

for $i = 1, 2$

find $\alpha_t(i)$, $\beta_t(i)$

using forward/backward Alg.

M-step: for $t = 1, \dots, 3$

for (i, j) pairs,

find $\xi_t(i,j)$

find $\gamma_t(i)$;

re-estimate A, B, π, go to E-step

Natural Language Processing
CSE 325/425

Sihong Xie

Lecture 9:
- Logistic regression for POS tagging

These are all called "discriminative" models rather than HMM

$Pr(Q|O,\lambda)$

MEMM
CRF
CRF-LSTM ← neural network

Long-short Term Memory

$Pr(Q|O,\lambda) = \frac{Pr(O,Q|\lambda)}{Pr(O|\lambda)}$ "generative" model

Drawbacks of HMM

HMM has very restricted parameter forms

- transition probabilities only models tag-tag relationships.
  - what if I want to know how NNP (proper noun) influences the prediction of VBN (verb, past particle)?

Secretariat	is	expected	to	race	tomorrow.
NNP	VBZ	VBN	TO	VB	RB

Simple but restricted parametrization

$$a_{ij} = Pr(q_{t+1}=j \mid q_t=i)$$

- emission probabilities only models word-tag relationships.
  - what if I want to use other features, such as
    - *“zzfish”* can have suffix feature “ish” or “fish” to predict adjective (like “*selfish*”) or noun (like “*kingfish*”).

$$b_{io} = Pr(o_t=o \mid q_t=i)$$

“Integer Index”

$Z = \exp\left\{\sum_{i=1}^{d} \theta_i f_i(\vec{x}, y = False)\right\} + \exp\left\{\sum_{i=1}^{d} \theta_i f_i(\vec{x}, y = True)\right\}$

Logistic regression for POS-tagging

These drawbacks can be addressed by a logistic regression model.

$y \in \{True, False\}$

$$\Pr(y = True|\mathbf{x};\boldsymbol{\theta}) = \frac{1}{Z}\exp\left\{\sum_{i=1}^{d}\theta_i f_i(\mathbf{x}, y = True)\right\}$$

- Input: a feature vector $\mathbf{f}(\mathbf{x}, y = True)$ $= \begin{bmatrix} f_1 \\ \vdots \\ f_d \end{bmatrix} \in \{0, 1\}^d$
- Output: a probability distribution over classes.
- Parameters: $\boldsymbol{\theta} \in \mathbb{R}^d$

- Example: predict if a credit card application should be approved.
  - Input: [annual income $\overset{\theta_1}{>}$ 50,000k, living $\overset{\theta_2}{\text{in}}$ city, $\overset{\theta_3}{\text{has}}$ job, has a $\overset{\theta_4}{\text{house}}$]
  - Output: probability of approval = 90%

Sigmoid function

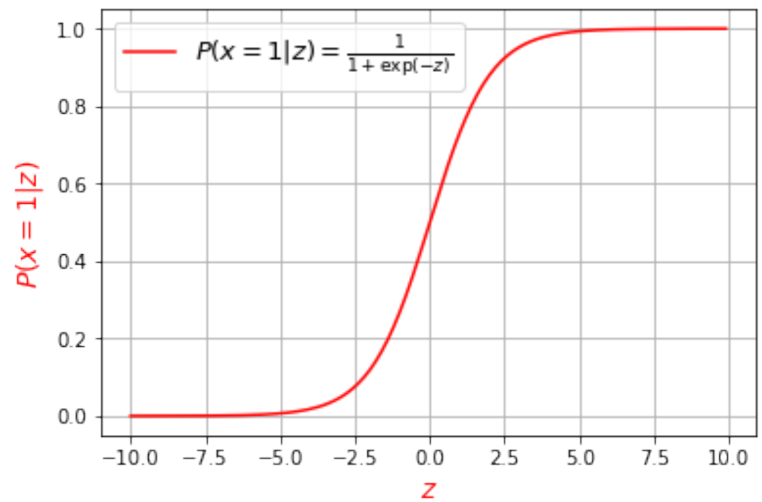
$\vec{u} = [u_1, \cdots u_d]$
$\vec{v} = [v_1 \cdots v_d]$
$\langle \vec{u}, \vec{v} \rangle = \sum_{i=1}^{d} (u_i v_i) \in \mathbb{R}$

The mapping from the inner product $\boldsymbol{\theta}^\top \mathbf{f}(\mathbf{x}, y = True)$ $= \sum_{i=1}^{d} \theta_i f_i(x, y = True)$

to the probability $\sigma(\boldsymbol{\theta}^\top \mathbf{f}(\mathbf{x}, y = True)) = \Pr(y = 1|\mathbf{x}; \boldsymbol{\theta})$

is done by the sigmoid function

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$



The range of sigmoid is (0,1) and the domain is $(-\infty, \infty)$

Multi-class logistic regression

To predict one out of more than two POS tags for a word o_t , we need multi-class logistic regression

$c \in \{1, \ldots, N\}$

- The probability of POS tag $q_t = c$ given some feature vector of word o_t :

$$\boxed{\sum_c \Pr(q_t = c | o_t ; \theta) = 1} \quad \Pr(q_t = c | o_t; \boldsymbol{\theta}) = \underbrace{\frac{1}{Z} \exp\left\{\sum_{i=1}^{d} \theta_i^c f_i(o_t, q_t = c)\right\}}_{\text{Soft max function}}$$

$\vec{\theta}^1 \in \mathbb{R}^d$ for pos tag 1

$\vdots$

$\vec{\theta}^N \in \mathbb{R}^d$ for pos tag N

- where the normalizing factor is

$$Z = \sum_{c' \in \{\text{All tags}\}} \exp\left\{\sum_{i=1}^{d} \theta_i^{c'} f_i(o_t, q_t = c')\right\}$$

$$\theta = [\vec{\theta}^1 \ldots \vec{\theta}^N] \in \mathbb{R}^{d \times N}$$

to do sth　　I do sth to sb.

TO VERB　　TO NN

Multi-class logistic regression

SLP 2

$t \in S$

o_t

- Examples
  - $f_1 : c = \text{NN AND } o_t = \text{"race"}$
  - $f_2 : c = \text{VB AND } q_{t-1} = \text{"TO"}$
  - $f_3 : c = \text{VBG AND } o_t = \text{ends with "ing"}$ gerund of verb
  - $f_4 : c = \text{VB AND } o_t = \text{is lower-case}$
  - $f_5 : c = \text{VB AND } o_t = \text{"race"}$
  - $f_6 : c = \text{NN AND } q_{t-1} = \text{"TO"}$

X =

Secretariat	is	expected	to	race	tomorrow.
NNP	VBZ	VBN	TO	VB	RB

Q =

o_t　c

$f_1(\text{"race"}, NN) = 1, f_2(\text{"race"}, NN) =$ 　VB or NN

		f_1	f_2	f_3	f_4	f_5	f_6
$c = \text{NN}$	$\mathbf{f}(o_t, \text{NN})$	1	0	0	0	0	1
	$\boldsymbol{\theta}_{\text{NN}}$	0.8	1	-2	3	0.1	-1.3
$c = \text{VB}$	$\mathbf{f}(o_t, \text{VB})$	0	1	0	1	1	0
	$\boldsymbol{\theta}_{\text{VB}}$	0.9	0.8	-1	0.01	0.1	0
$c = \text{VBG}$	$\mathbf{f}(o_t, \text{VBG})$	0	0	0	0	0	0
	$\boldsymbol{\theta}_{\text{VBG}}$	1	0.3	9	0.3	-0.4	-3.4

- Prob of NN

$$= \frac{1}{Z} \exp\left\{ \sum_{i=1}^{6} \theta_i^{NN} \times f_i(\text{"race"}, NN) \right\}$$

$$= \frac{1}{Z} \exp\{ 0.8 \times 1 + 1 \times 0 + (-2) \times 0 + 3 \times 0 + 0.1 \times 0 + (-1.3) \times 1 \}$$

$$= \frac{1}{Z} \exp\{0.8 - 1.3\} = \frac{1}{Z} \exp\{-0.5\}$$

$$Z = \exp\left\{ \sum_{i=1}^{6} \theta_i^{NN} \times f_i(\text{"race"}, NN) \right\} + \exp\left\{ \sum_{i=1}^{6} \theta_i^{VB} \times f_i(\text{"race"}, VB) \right\} + \exp\left\{ \sum_{i=1}^{6} \theta_i^{VBG} \times f_i(\text{"race"}, VBG) \right\}$$

Training multi-class logistic regression

Use MLE

- Training data: all pairs of (o_t, q_t) from all POS-tagged sentencens.
- Construct feature vectors $\mathbf{f}(o_t, q_t = c)$
- Find the log-likelihood

$$\ell(\boldsymbol{\theta}; \text{Training corpus}) = \sum_{t=1}^{m} \sum_{c \in \{\text{All tags}\}} [q_t = c] \log \Pr(q_t = c | o_t; \boldsymbol{\theta})$$

- Stochastic gradient descent
  - The gradient of the negative log-likelihood of one training pair (o_t, q_t) w.r.t. $\boldsymbol{\theta}_c$

$$\frac{\partial}{\partial \boldsymbol{\theta}_c}[-\log \Pr(q_t = c | o_t; \boldsymbol{\theta})] = -\{[q_t = c] - \Pr(q_t = c | o_t; \boldsymbol{\theta})\}\mathbf{f}(o_t, q_t = c)$$

  - Taking one gradient descent step moves the parameter closer to the feature vector.