Natural Language Processing - Project 2

Zhanhao Chen
Oct 29, 2017

Questions

1. Proof
   Initialization:
   $\because \hat{\alpha}_1(i) = c_1 * \tilde{\alpha}_1(j) = c_1 * \pi_i b_{iO_1} = c_1 * \alpha_1(j)$
   $\therefore$ when t = 1, $\hat{\alpha}_t(j) = \prod_{s=1}^{t} c_s \alpha_t(j)$ is true.

   Recursion:
   $\hat{\alpha}_t(j) = c_t * \tilde{\alpha}_t(j) = c_t * \sum_i \hat{\alpha}_{t-1}(i) a_{ij} b_{jO_t}$
   When $\hat{\alpha}_{t-1}(j) = \prod_{s=1}^{t-1} c_s \alpha_{t-1}(j)$,
   $\hat{\alpha}_t(j) = c_t * \sum_i \prod_{s=1}^{t-1} c_s \alpha_{t-1}(j) a_{ij} b_{jO_t} = \prod_{s=1}^{t} c_s \sum_i \alpha_{t-1}(i) a_{ij} b_{jO_t} = \prod_{s=1}^{t} c_s \alpha_t(j)$

   $\therefore \hat{\alpha}_t(j) = \prod_{s=1}^{t} c_s \alpha_t(j)$

   $\because \sum_j \hat{\alpha}_t(j) = 1$
   $\therefore \sum_j \prod_{s=1}^{t} c_s \alpha_t(j) = 1$
   $\therefore \sum_j \alpha_t(j) = \frac{1}{\prod_{s=1}^{t} c_s}$

2. Proof
   Initialization:
   $\because \hat{\beta}_T(i) = c_T * \tilde{\beta}_T(j) = c_T * \beta_T(j)$
   $\therefore$ when t = T, $\hat{\beta}_t(j) = \prod_{s=t}^{T} c_s \beta_t(j)$ is true.

   Recursion:
   $\hat{\beta}_t(j) = c_t * \tilde{\beta}_t(j) = c_t * \sum_i \hat{\beta}_{t+1}(i) a_{ij} b_{jO_{t+1}}$
   When $\hat{\beta}_{t+1}(j) = \prod_{s=t+1}^{T} c_s \beta_{t+1}(j)$,
   $\hat{\beta}_t(j) = c_t * \sum_i \prod_{s=t+1}^{T} c_s \beta_{t+1}(j) a_{ij} b_{jO_{t+1}} = \prod_{s=t}^{T} c_s \sum_i \beta_{t+1}(i) a_{ij} b_{jO_{t+1}} = \prod_{s=t}^{T} c_s \beta_t(j)$

   $\therefore \hat{\beta}_t(j) = \prod_{s=t}^{T} c_s \beta_t(j)$

   $\because \sum_j \hat{\beta}_t(j) = 1$
   $\therefore \sum_j \prod_{s=t}^{T} c_s \beta_t(j) = 1$
   $\therefore \sum_j \beta_t(j) = \frac{1}{\prod_{s=t}^{T} c_s}$

3. Proof
   $\xi_t(i,j) = \frac{\alpha_t(i) \beta_{t+1}(j) a_(ij) b_{jO_{t+1}}}{P(O|\lambda)}$
   Replace $P(O|\lambda)$ with $\sum_j \alpha_T(j) = \sum_j \beta_1(j) = \frac{1}{\prod_{s=t}^{T} c_s}$

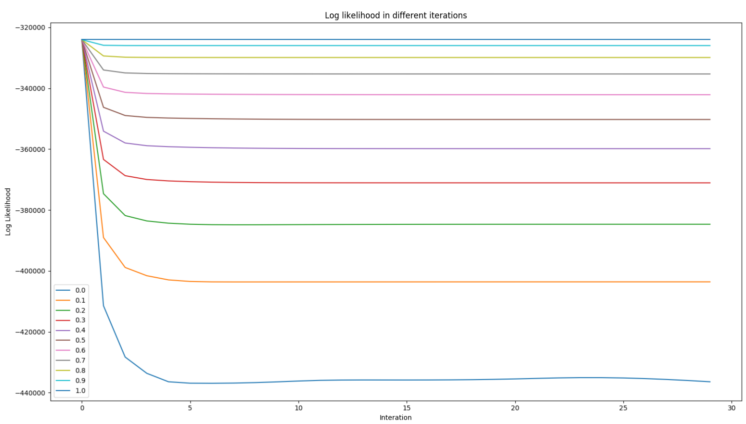
Replace $\alpha_t(j)$ with $\frac{\hat{\alpha}_t(j)}{\prod_{s=1}^{t} c_s}$
Replace $\beta_{t+1}(j)$ with $\frac{\hat{\beta}_{t+1}(j)}{\prod_{s=t+1}^{T} c_s}$

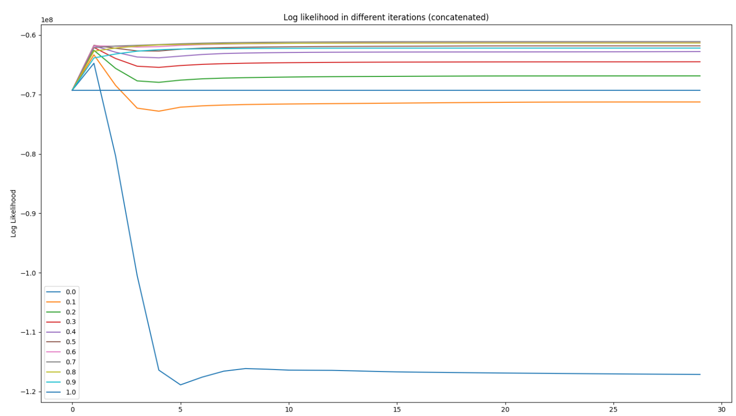
$$\xi_t(i,j) = \frac{\hat{\alpha}_t(i)\hat{\beta}_{t+1}(j)a_(ij)b_{jO_{t+1}}}{\prod_{s=t}^{T} c_s * P(O|\lambda)} = \hat{\alpha}_t(i)\hat{\beta}_{t+1}(j)a_(ij)b_{jO_{t+1}}$$

4. Proof
$\because \sum_j a_{ij} = 1$ for both A_L and A_D
$\therefore \sum_j a_{ij} = \mu \sum_j a_{Lij} + (1-\mu)\sum_j a_{Dij} = 1$
$\therefore A = \mu A_L + (1-\mu) A_D$ defines appropriate probability distributions.
The similar proof can be used to prove B and π.

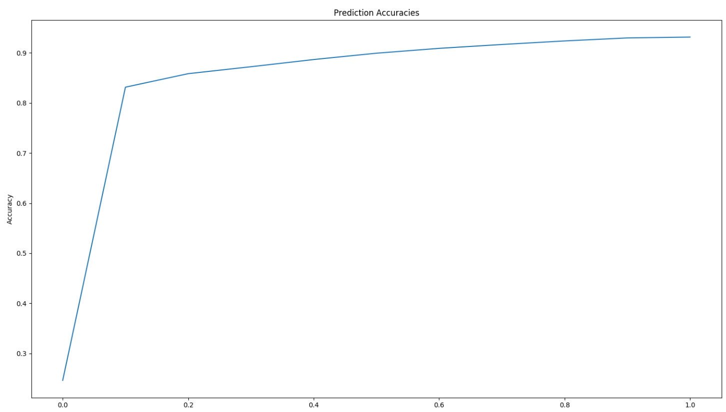
Performance

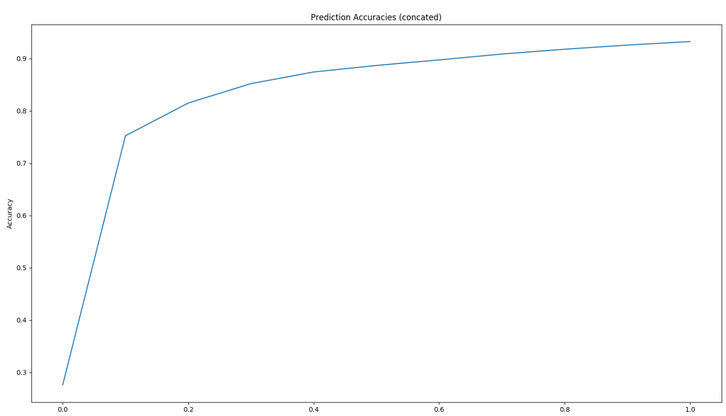
1. log-likelihood Plot





2. Accuracy Rate Plot





Improvement

1. Improvement for Accuracy
   During the maximization, I will do the smoothing for both γ and ξ. Because they are the expected number, we could treat them as the count in MLE. So I add 0.1 to each expected value and then do the scaling. This operation could prevent NaN showing as well as improve accuracy significantly.

2. Improvement for Speed
   In order to achieve speed up, I use multi-thread as well as multi-process

design.

Multi-process could be achieve easily by modifying the run bash file. The computer do HMM for different mu. But the bottleneck is memory. On my computer (8 cores, 16G memory), I could at most run 4 HMM parallelly. So the average CPU usage is 50%.

To achieve multi-thread, I create several threads in HMM to do the expectation for different sentence parallelly. But it will consume more space for each HMM to save the intermediate In this case, all of the 8 CPUs are running to solve 1 HMM. But the average usage of each CPU is still 50%.

Then I combine both designs. Then it achieve 100% usage for all of the 8 CPUs with at most 7.2G memory consumption.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
4302	jason	20	0	7448800	2.354g	16616	S	261.5	15.3	79:13.57	java
4304	jason	20	0	7447772	2.334g	16624	S	255.5	15.2	78:49.11	java
4303	jason	20	0	7382236	2.398g	16612	S	246.2	15.6	78:55.83	java

The figure above shows the performance. I can run three HMM program parallelly. Each program consume at most 2.4G memory and use about 250% of CPU, which means the total usage rate is more than 750%.

The program will take 1 minute and 13 seconds to finish the HMM for test.txt and 156 minutes and 51 seconds for concated.txt.