Independent Project #2

1.Hidden Markov Models

1.1 A toy problem

START	G	C	A	C	T	G	END
H	-1.7148,START	-3.8351,H	-6.3608,H	-8.4811,H	-11.0068,H	-12.6808,L	
L	-2.5257,START	-4.2405,H	-5.9553,H	-8.2579,L	-9.8673,L	-12.1699,L	
Prediction	H	H	L	L	L	L	NA

For the last state to 'END': v = -13.3739, bp = L
The first value in the cell is v_i(STATE), the second value is b_i(STATE)
This is carried out by the viterbi decode function implemented in `viterbi.py`

So decode result of the hidden states of the given DNA sequence is H, H, L, L, L, L

1.2 POS Tagging

1. K = 3
   V = 16024
   Process by using lower case and map digits to 'NUM', map term frequency less than K to 'UNK'
2. Results in `jw7jb-tprob.txt`
3. Results in `jw7jb-eprob.txt`
4. Results in `jw7jb-tprob-smoothed.txt` and `jw7jb-eprob-smoothed.txt`
5. The accuracy per token is 94.46%
6. Results in `jw7jb-viterbi.txt`
7.

Accuracy	α	β
94.46%	1.0	1.0
94.47%	10	1.0
94.50%	100	1.0
94.56%	1000	1.0
94.79%	1000	0.5
95%	1000	0.1
95.01%	1000	0.01
95.05%	500	0.05

The best result I can obtain is 95.05% with $\alpha = 500$ and $\beta = 0.05$

8.

Results in `jw7jb-viterbi-tuned.txt`

2.Conditional Random Fields

The originial model peformance on dev set is 0.457

1. With extra features:

- the first letter of the word
- the last letter of the word

The performance on dev set increases to 0.632

- the first two letters of the word
- the last two letters of the word

The performance on dev set increases to 0.676

- the next word
- the last word

The performance on dev set increases to 0.698

- the word is digit or not

The performance on dev set increases to 0.701

2. Switch to average perceptron with 20 iteration the performance on dev set increases to 0.873

I believe they brought the concept of structured perceptron to learn CRF parameter with averaged perceptron, on textbook 7.5.1 and online resources Recalled that perceptron alogrithm:

$$\hat{y} = argmax_{y \in Y} \theta \cdot f(x, y)$$
$$\theta^{(t+1)} \leftarrow \theta^{(t)} + f(x, y) - f(x, \hat{y})$$

Similarly in the case of CRF, we can define a function such that the probility distribution over the whole sequence is $logP(y|x) = C + \theta f(x, y)$

$$f(x, y) = \sum_{t=0}^{T} f(t, x, y) + \sum_{t=1}^{T} f(y_{t-1}, y_t)$$

Then we can learn such θ with the perceptron algorithm to maximize the probability distribution $logP(y|x)$

As in perceptron classification, averaging the weight $\theta^{(t)}$ learnt in each iteration t is crucial to get good performance, and we refer such strategy as averaged perceptron