1) I am answering all questions using training data.

a)Number of different part of speech tag types defined: 45

b)The ten most common tuples centered around the word “to”:

('%', 'to', '$')
('according', 'to', 'the')
(‘priced', 'to', 'yield')
('expected', 'to', 'be')
('plans', 'to', 'sell')
('consented', 'to', 'findings')
('ordered', 'to', 'disgorge')
('%', 'to', '8')
(',', 'to', 'be')
('seem', 'to', ‘be')

c)'to' is defined as:
TO
JJ
IN

It is surprising that ‘to’ is used as an adjective(JJ).

d)The twenty most common syntactic patterns surrounding the word “to”:

('NN', 'TO', 'VB')
('VBN', 'TO', 'VB')
('NNS', 'TO', 'VB')
('VBD', 'TO', 'VB')
('VBZ', 'TO', 'VB')
('JJ', 'TO', 'VB')
('CD', 'TO', 'CD')
('NN', 'TO', 'DT')
('VBP', 'TO', 'VB')
('VBG', 'TO', 'VB')
('VB', 'TO', 'VB')
('NN', 'TO', 'CD')
('NNP', 'TO', 'VB')
('RB', 'TO', 'VB')

('VBN', 'TO', 'DT')
('NN', 'TO', '$')
('JJ', 'TO', 'DT')
('CD', 'TO', 'VB')
('VBG', 'TO', 'DT')
('NNS', 'TO', 'DT')

e)The word is down.

The other tag is IN.

The sentence when down is used as RP:

The White House said Mr. Bush decided to grant duty-free status for 18 categories , but turned down such treatment for other types of watches `` because of the potential for material injury to watch producers located in the U.S. and the Virgin Islands .

The sentence when down is used as IN:

the Dell models use , generally have been coming down as chip prices have fallen .

In the first sentence, down has been used as RP(particle). Particles do not change. In the second sentence down is used as preposition.

2a) Changing the initial starting probabilities as described in the question (I made p(H|C)=0.8, p(C|H)=0.8, p(H|H)=0.1 and p(C|C)=0.1) it was observed that the final values of the transition probabilities at which the EM algorithm converged had p(H|C) > p(C|C) and p(C|H) >p(H|H). This shows that changing the initialization values can result in different minima and hence it is necessary to run the EM algorithm with many different random restarts. I also observed that the perplexity of the new model is 3.2129 and that of the default model is 2.827, hence proving that the new initialization values result in a local minima for the perplexity.

b) Upon using the values p(H|C)=P(C|H)=P(C|C)=P(H|H)=0.45 as initializing values for the model, I found that the perplexity is 2.8348 and the final iteration values of the emission and transition probabilities are very close to the values given in the excel sheet with the default initialization values. This shows that both of these initialization values lead to the same minima which indicates that this value of perplexity might be a global minima.

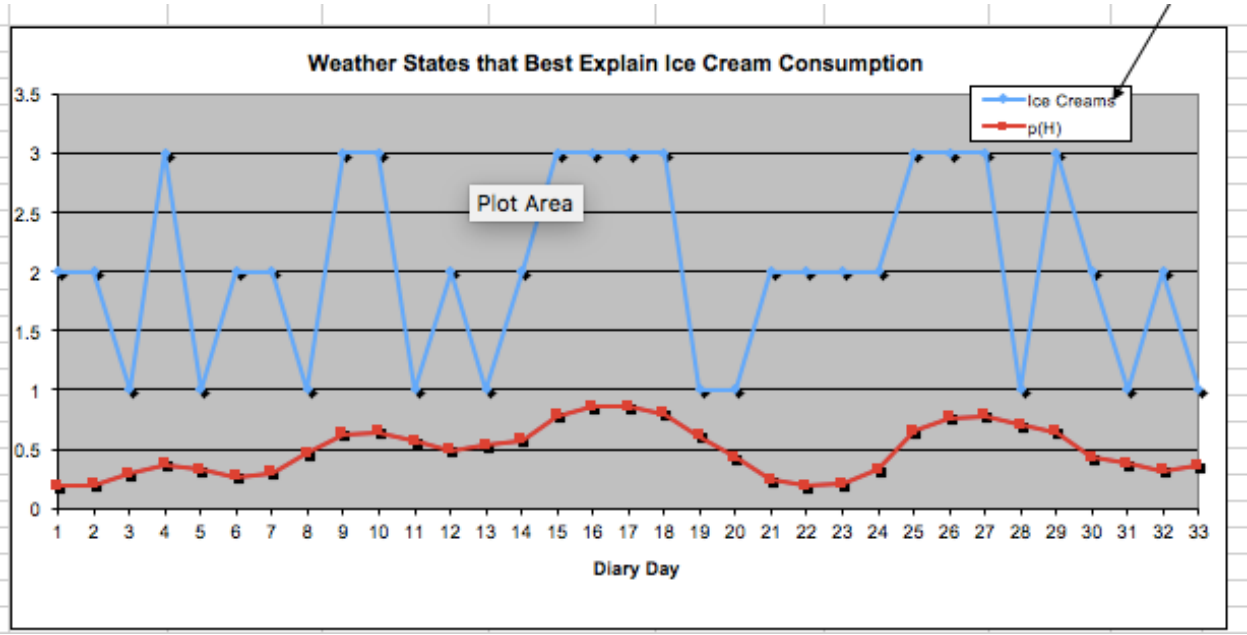
c) Upon using the values of P(1|C)=P(2|H)=0.1 and P(2|C)=P(1|H)=0.7, this iteratively leads to a final model whose perplexity is 2.8271 which is very very close to 2.8270 for the default initialization values. This implies that this set of initialization values can also lead to a good model but intuitively the final converged values of the emission probabilities (P(1|C)= 1.6E-04, P(2|C)=0.5341, P(1|H)=0.6407 and P(2|H)=0.1481) does not make sense. This phenomenon occurred because the initialization values were themselves counter intuitive. This shows that while working on the EM algorithm, the initialization values are very important and if we have any intuitive ideas about the final values , then we should implement it in the algorithm in the form of initialization values.

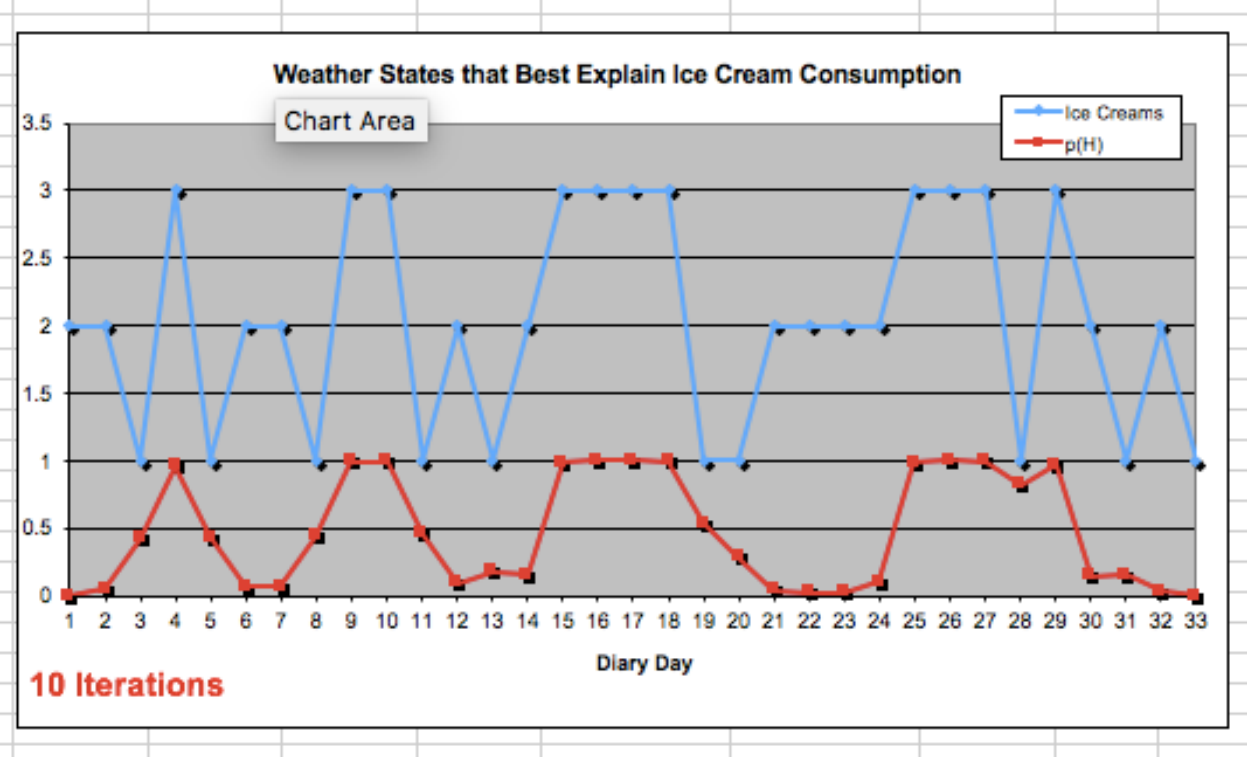
d) Upon using the values P(1|C)=P(2|C)=P(1|H)=P(2|H)=0.001 we found that the perplexity of the model was 3.3128 which is much larger than the default perplexity which is about 2.827 and the final converged emission probabilities are all 0.33. Since we are not considering any hidden nodes in this model , this model is a trivial one where all the final emission probabilities are same and hence the perplexity is so high. This also indicates that weather is a very important latent parameter that needs to considered for this model.

e) The answer to this question is similar to the previous one. If we have more additional factors that can affect the number of ice creams Jason can eat such as the cost and his budget, we can include these factors in the conditional probability table . This leads to an increasingly complex model but we can also model the training data very closely. Hence the perplexity will be much lower if the number of parameters is increased but we also need to look out for overfitting which will happen if we have too many parameters

f) Here is the initial probabilities used for this dataset. Since the number of times the person is eating a single ice-cream is happening at regular intervals and it is not happening continuously, I guessed that the person simply loves to eat ice-cream and he has a high probability of eating 2 ice-cream even in cold weather. The perplexity of this model is 3.055. The weather as predicted from the model is the following in the order of days Cold,Cold,Cold,Hot,Cold,Cold,Cold,Cold,Hot,Hot,Cold,Cold,Cold,Cold,Hot,Hot,Hot,Hot,Hot,Cold,Cold,Cold,Cold,Cold,Hot,Hot,Hot,Hot,Hot,Cold,Cold,Cold,Cold.

	p(...|C)	p(...|H)	p(...|START)
p(1|...)	0.3	0.3	
p(2|...)	0.5	0.2	
p(3|...)	0.2	0.5	
p(C|...)	0.8	0.1	0.5
p(H|...)	0.1	0.8	0.5
p(STOP|...)	0.1	0.1	0





The initial reconstruction of the weather is given in the following graph
The final graph after 10 iterations looks something like this

After 10 iterations, P(1|H)=0.2453 . At every re-estimation step, it decreased from the initial 0.3 which we used. This is because in hot weather, he is much lesser inclined than ever to eat only a single ice-cream.

3)This paper deals with tagging part of speech labels in the context of a sentence in a semi-supervised scenario when both tagged text using hand and untagged raw data is available. The author claims that using more tagged text is always favorable for improving the accuracy of the model except when the amount of tagged text itself is limited. The author has conducted several experiments to support this claim . As usual there are two measures for the quality of a tagging procedure i.e. on the sentence level and on the word level. They have defined the triclass model for predicting the tags where they have used the following probabilistic graph model. For a tag t_i , its parents are the tags t_{i+1} and t_{i+2} and the parent of a particular word is just its own tag. This mimics a trigram markov model where each tag depends on the previous 2 tags. They have used the Forward Backward algorithm to calculate the maximum likelihood of the tagged sequence and find the model that maximizes the probability of the tagged text. The authors show experiments in the first section that validate their claim that ML training reduced the perplexity with increasing iterations but the tagging accuracy is related to number of tagged sentences available. As a matter of fact using just 100 tagged sentences results in an accuracy of 92.6% after 10 iterations. The authors have also shown 2 different constrained ML models 1>tw-constraint where p(tag|word) does not change after an iteration.
2>t-constraint where p(tag) does not change after an iteration.
The authors show that such constrained ML models does not degrade the RF model (supervised model) as much as the standard ML models.

Finally, this paper is really interesting and significant since it emphasizes on the fact that even if there is no training data available, a few manually obtained training data can dramatically improve the performance and accuracy of the tagging model. It also shows that certain simple constraints on the ML model can also improve the performance.

4) Per token accuracy: 0.883054892601
Per sentence accuracy: 0.237734343837

The top 5 POS tags that are most commonly identified correctly:

#
$
TO
:
,

The top 5 POS tags that are most commonly identified incorrectly:
LS
SYM
UH
FW
RP

The criteria for identifying correct tags is correct tag/(correct tag+incorrect tag)
The criteria for identifying incorrect tags is incorrect tag/(correct tag+incorrect tag)

I am computing probabilities as bigram counts. p(w|t)=(count(w,t)+lambda)/(count(t)+V*lambda), where V is the vocabulary. I have done the tuning on development set and computed the parameter as 0.03.

To improve the probabilities I can use trigram probabilities i.e., p(w | t1,t2) and p(t3 | t1,t2). Also I can use better shooting techniques like interpolation or back off.

5) I have implemented “Regular” (“soft”) EM, evaluated with both Viterbi and posterior decoding.

I have used training data to get the initial counts. And then used only first 10000 words from raw data for each iteration of EM to reduce the time complexity of each iteration. The accuracy would have been improved if a larger subset was used.

I have evaluated the perplexity and accuracy of Viterbi decoding on the test data at the end of each iteration.

First iteration,

```
Per token accuracy:0.747701566878
Per sentence accuracy:0.0885520542481
Perplexity:6.97488914849e+86
```

Second iteration,

```
Per token accuracy:0.736536266473
Per sentence accuracy:0.080175508576
Perplexity:5.30616258307e+87
```

Third iteration,

```
Per token accuracy:0.723855971775
Per sentence accuracy:0.0765855604308
Perplexity:2.74259528174e+88
```

Fourth iteration,

```
Per token accuracy:0.720908996576
Per sentence accuracy:0.0745911447946
Perplexity:4.80288237185e+88
```

Fifth iteration,

```
Per token accuracy:0.720535436339
Per sentence accuracy:0.0733944954128
Perplexity:5.58132872299e+88
```

Sixth iteration,

```
Per token accuracy:0.720431669607
Per sentence accuracy:0.0737933785401
Perplexity:5.87271236977e+88
```

Seventh iteration,

```
Per token accuracy:0.720390162914
Per sentence accuracy:0.0741922616673
Perplexity:6.0322973434e+88
```

Eighth iteration,

```
Per token accuracy:0.72117879008
Per sentence accuracy:0.0737933785401
Perplexity:6.13000128514e+88
```

Ninth iteration,

```
Per token accuracy:0.720348656221
```

```
Per sentence accuracy:0.0737933785401
Perplexity:6.15716516363e+88
```

Tenth iteration,

```
Per token accuracy:0.720348656221
Per sentence accuracy:0.0737933785401
Perplexity:6.1800750043e+88
```

Finally I have evaluated accuracy of posterior decoding on test data.

```
Per token accuracy:0.819943965965
Per sentence accuracy:0.134024730754
```

I have used lambda smoothing with lambda=0.03.

Ideas to improve the algorithm:
Firstly, if a larger subset of raw data could have been used to improve the accuracy.
Second, using trigram probabilities instead of bigram probabilities should improve the accuracy that is the the parents of a particular tag are the previous 2 generated tags. This is the model experimented with in the paper in Question 3 which gives a number of new ideas to improve the algorithm such as using tw constraint or t constraint.
Third, better smoothing techniques can be used like interpolation or back off.

Local copies of training, raw and test data is used in the code. To run the code, just change the paths while calling the methods.

I have used logsumexp from scipy.misc.