### Problem 1

**Question 1**

Part a)

We are asked to find

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  |
|  | a | (0.100)(0.400)=0.040 | (0.120000)\*(0.100000)(0.350000)=0.004200 |
| n | (0.400)(0.300)=0.120 | (0.020000)\*(0.400000)(0.500000)=0.004000 |
| o | (0.200)(0.100)=0.020 | (0.120000)\*(0.100000)(0.500000)=0.006000 |
| t | (0.300)(0.100)=0.030 | (0.040000)\*(0.100000)(0.600000)=0.002400 |

Highest probability = 0.006000

Part b)

Using the same construction:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | |  |  | 3 |
|  | a | 0.02 | 0.0048 | 0.00048 |
| n | 0.04 | 0.0012 | 0.0012 |
| o | 0.02 | 0.024 | 0.00048 |
| t | 0.12 | 0.0012 | 0.00216 |

Highest prob = 0.002160

Part c)

Using the same construction:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  |  | 3 | 4 |
|  | a | 0.02 | 0.0144 | 0.00021 | 0.000035 |
| n | 0.04 | 0.006 | 0.00018 | 0.00036 |
| o | 0.04 | 0.0036 | 0.0018 | 0.000035 |
| t | 0.09 | 0.0012 | 0.000864 | 0.000054 |

Highest prob = 0.000360

**Question 2**

Using the Viterbi Algorithm, for the first character, the run time is . For the other characters, the run time is . So the total runtime is .

The brute force method is exponential: .

The Viterbi Algorithm is much faster with large k.

**Question 3**

First, we’ll show how to compute several smaller probabilities:

Essentially, probability of going through each is the product of the conditional probability of getting to from . This is true under the assumption of first order Markov model.

Essentially, probability of getting y in the tth iteration is the sum of all ways of getting to . This is a sum of exponential number of terms, as you need to sum up all paths that get to .

Note that this can be calculated rather efficiently with DP, where we could transform the above equation into:

Then we could iteratively calculate for all possible characters. This could also be seen as a restatement of Law of Total Probability.

Essentially, all ways to get to times the probability of emitting from each . This is again a restatement of Law of Total Probability. is calculated above and is known.

This is just Bayes rule.

Now, back to what we actually need to show. Under the first order Markov model, we need to compute

We already know how to find from above.

The only thing we need now is show how to find :

Thus, we could calculate all intermediate quantities, and it is possible to calculate

**Question 4**

The HMM-based approach (by simply replacing characters with words in the above formulation) would do very poorly against Google translate for three major reasons. Firstly, this formulation pays no attention to grammar (i.e. bank could be a verb and noun). Secondly, this formulation requires the two sentences be the same length, which is almost never the case in real life. Lastly, the formulation requires the two sentences to have the same structure; there are cases where one sentence must be inverted to be translated to another, such as Japanese-English translation, where the verb in Japanese comes close to the end of the sentence.

### Problem 2

**Question 1**

We firstly notice that the total number of hypothesis is . We also realize that the size of training sample is , and the prediction error of , we have:

For all h, . Then, we know that the total prediction error can be bounded by with probability .

We want to bound this by . Then we have:

Thus, the prediction error could be bounded by .

**Question 2**

We know that

Therefore, since ,

Or

We will transform this into VC dimension, since we need a bound on the number of classifiers :

We also know that VCDim(H) of a linear classifier H is d+1. So, for a 100-dimensional problem, d is 101.

Hence,

**Question 3**

We know from the lecture that . Then in our case, we need to calculate such that for , , which gives us:

In our case, , we have , and thus has to be at least 3768.

**Question 4**

Part a)

We will construct a set of size d in d dimension, such that it can be shattered.

For a point in the set , assign the bit value 1 and everything else value 0. For example for , . Then we’ll show that if we want to classify k points as positive: . Then a sphere of radius 1 centered at , where , will contain all points in and exclude all points in .

There are two things to prove with the construction:

1. All points in is not contained in the sphere

Notice that by construction, all points are orthogonal to each other. Therefore a linear combination of points in (call this ) will be orthogonal to any points in . Then note that the distance between and a point in will be since . So, the sphere will no longer contain a point in .

1. All points in is contained in the sphere

Again, we use the fact that all points are orthogonal. Call a linear combination of points in (same as before). The distance from the center of the sphere to a point in :

Since we constructed , then

So we know that .

Therefore,

which means a sphere of radius 1, centered at will contain it.

Part b)

We can show that any spherical classifier in dimension could be reduced to a linear classifier in dimension. Therefore, since we know the result that linear classifier in dimension has the VC-Dimension of , and spherical classifier in dimension is simply a special case for the general linear classifier in dimensions, we can conclude that spherical classifier in dimension has the VC-Dimension no more than , which is the desired result.

To show we can reduce the spherical classifier with dimension , we essentially need to find , such that , where for all . We also realize that , since we know both and are positive, squaring them will not change the inequality.

Expand the value in the left hand side, we have:

Since we have the freedom of choosing the transformation function that converts into in dimension, we could simply pick such that for , and for . Thus, the original left hand side becomes:

Then, if we denote , for and for , we realize immediately that:

Thus, we successfully constructed such that the result of the resulting dimensional linear classifier is the same as the dimensional spherical classifier given.

Since we could reduce any spherical classifier in dimension to a linear classifier in dimension, following the argument given in the beginning of the setup, we know that the VC-Dimension of dimension spherical classifier cannot be more than .