STATISTICAL STRUCTURED PREDICTION
Question set 1

Ramon Ruiz Dolz

November 2018

1 Theoretical questions

1.1 Question 1

Given an input observation $x \epsilon X$ and an output hypothesis $y \epsilon Y$, a discriminative model $P(y \mid x)$ is equivalent of knowing the differences between both sets (e.g. a language). As a language is potentially infinite, this makes this problem computationally hard. One way to approach it is to have a finite structural model Y(M) and to search a $y \epsilon$ Y(M) that maximises the $P(y \mid x)$,

$$f(x) = argmax_{y \epsilon Y(M)} P(y|x) \tag{1}$$

An example of a discriminative model are the Conditional Random Fields (CRF). A CRF consists on a model quite similar to a Hidden Markov Model (HMM). A CRF is represented by a graph where each node represents a random variable and each edge represents the dependence between each pair of nodes connected.

On the other hand, a generative model is a model equivalent of knowing both sets (e.g. languages). This model, represented by the join probability $P(x, y)$ is equivalent as finding in all the finite model Y(M), a y that maximises the following product,

$$f(x) = argmax_{y \epsilon Y(M)} P(x|y)P(y) \tag{2}$$

One example of a generative model can be the probabilistic context-free grammars. This model builds a context free grammar with associated probabilities to each production. With that grammar it is possible to "build" all the language as if knowing it completely.

1.2 Question 2

The highest probability tree for the sentence *abab* is the following,

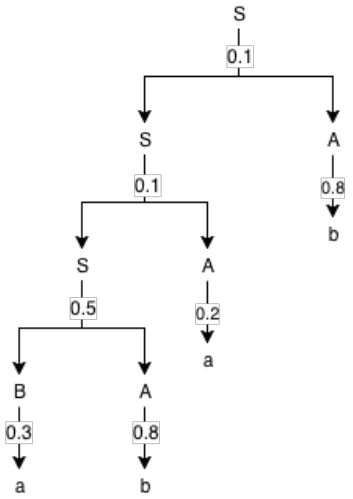

Figure 1: Probability tree

The probability for generating the string *abab* using this probability tree is,

$$\begin{aligned} P(abab) = P(S \to SA) * P(A \to b) * P(S \to SA) \\ * P(A \to a) * P(S \to BA) * P(A \to b) * P(B \to a) \\ = 0.1 * 0.8 * 0.1 * 0.2 * 0.5 * 0.8 * 0.3 \\ = 1.92 * 10^{-4} \end{aligned} \quad (3)$$

1.3 Question 3

Let assume L_p as the set of all probabilistic languages. A probabilistic language is a language that the sum of all probabilities is 1, meaning that is consistent. Let assume G_p a probabilistic grammar. We say that a probabilistic grammar G_p is consistent if $\sum_{x \epsilon L(G)} P_{G_p}(x) = 1$. It is important to emphasise that a probabilistic grammar can generate many languages that are not consistent. So, only the subset of consistent languages generated by a probabilistic grammar G_p will be part of the probabilistic languages set L_p.

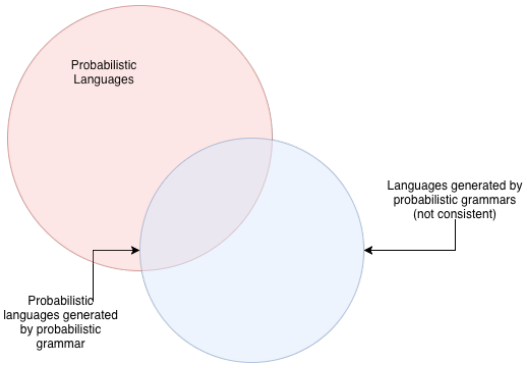

Figure 2: Sets of languages

Figure 2 depicts the probabilities explained on the last paragraph. Only the intersection between both sets, probabilistic languages and languages generated by a probabilistic grammar, are probabilistic languages generated by probabilistic grammars. All the languages outside the intersection but in the set of languages generated by a probabilistic grammar are inconsistent languages.

1.4 Question 4

The initialisation of *Outside* algorithm states:

$$\forall A \epsilon N, f(A, 0, T) = \delta(A, S) \tag{4}$$

Kronecker delta is a two variable function that returns a 1 if both variables are the same or 0 in other case:

$$\delta = \begin{cases} 1 & A = S \\ 0 & A \neq S \end{cases}$$

The initialisation is made this way because the trivial case is when the substate A is equal to the initial state S of the tree. In that case the probability must be the maximum, therefore it is set to 1.

The final result of the *Outside* algorithm is stated as:

$$P_{G_\theta}(x) = \sum_{A \epsilon N} f(A, i, i+1) * p(A \rightarrow x_{i+1}) \tag{5}$$

Therefore, the final result is the product of the sum of every right production and every left production multiplied by the own A production probability. The algorithm computes recursively the probabilities of all the *outside* productions so that it is possible to get the probability of a determined sentence x.

1.5 Question 7

Considering Viterbi, a new version of the algorithm for cases where all the right productions lead to a terminal symbol would be:

$$\hat{e}(A, i, i+l) = max_{B,C \epsilon N}\{p(A \rightarrow BC)\} * \hat{e}(B, i, i+l-1) * p(C \rightarrow b) \quad (6)$$

the most important difference is that, in *normal* Viterbi it was important to maximise both sub-trees B and C. In this version, since the left production will always be a tree and the right one a terminal symbol, that part of the algorithm is *static*. As can be observed, in this version there is no maximisation of the sub-trees so C will always be a terminal symbol production and, therefore, B will be the left tree except the rightmost terminal. Taking this modification into account, the computational cost of the algorithm will be reduced from O(n^3) to O(n^2).

2 Practical assignments

2.1 Statistical evaluation

The first part of the practical assignments consists on evaluating the performance of the 4 different grammatical models given to us. To evaluate the different models I have calculated the likelihood as a metric to determine the quality of each model. Concretely two probabilities have been computed, the direct and the Viterbi. Additionally I have also computed the test set perplexity for each model.

Model	**Log-likelihood**	**Viterbi log-likelihood**	**Test Set Perplexity**
Gm1	-9335.755	-9834.276	11336.193
Gm2	-9448.184	-10128.235	12685.117
Gm3	-7304.947	-9449.416	1487.641
Gm4	-5896.243	-11678.729	363.668

Table 1: Results of the statistical evaluation.

The results obtained can be observed in the Table 1. Three different metrics have been evaluated in order to do the statistical evaluation of the grammatical models. Both log-likelihood and Viterbi log-likelihood are metrics defined as,

$$logP_M(D) = \sum_{x \epsilon D} logP_M(x) \quad (7)$$

where the probability can be the standard or the Viterbi one. This metric indicates, when higher, more accurate probabilities. So, with log-likelihood the best model should be Gm4 and with Viterbi log-likelihood the best model is Gm3. The later metric taken into account is the test set perplexity. It is defined as,

$$2^{\frac{1}{N}\sum_x log_2 P(x)} \tag{8}$$

and represents the quality of a prediction of a model. In this case, the lower the value better the result. Taking into account the perplexity of the test set in each model, the best model should be Gm4.

2.2 Structural evaluation

The second part of the assignment is the structural evaluation. To evaluate the structure of the different models, since the original grammar is available to use, I have generated 1000 samples from each grammar. The experiment consists on checking how many of that samples are palindromes, in order to check the *quality* of the grammar.

Test Set	**Palindromes**
D1	358
D2	293
D3	677
D4	1000

Table 2: Results of the structural evaluation.

The results obtained from this experiments are shown in the Table 2. The seed used to do the experiment is 0. As the test set size is 1000, one possible deduction could be that, the grammar model 4 has generated a *perfect* test set, the grammar model 3 has generated an acceptable test set, and grammar models 1 and 2 have generated two different poor test sets. In order to make a deeper analysis, I have also calculated the accumulated probability mass (apm) of each test set. The apm consists on the sum of the probability of all palindromes generated by a determined grammar model.

Test Set	**apm**
D1	28.176
D2	26.128
D3	37.364
D4	29.505

Table 3: Accumulated probability mass of each test set.

The apm values associated to each test set can be observed in Table 3. Here we can observe the importance of doing this deeper analysis of the models. Before we said that grammar model 4 generated a *perfect* test set. Here it is possible to observe that, even though all the generated samples were palindromes, the probabilities associated to each one are lower than the probabilities of the test set obtained by grammar model 3. With less samples, higher accumulated probability. With this deeper analysis it is possible to conclude that,

even though Gm4 generated more palindromes, Gm3 obtained better samples. It is also possible to conclude that Gm1 and Gm2 are the weaker models from all the four models.