INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY, DELHI

CSE561-Probabilistic Graphical Models : Assignment 1

Ashutosh Gupta and Sagar Verma

March 28, 2016

1 Problem Description

1.1 Optical Word Recognition

Optical word recognition problem is the identification of word from a given image of characters. OWR is an extension of Optical character recognition. Through various comptuer vision techniques individual characters can be obtained from an image with certain probability for each characer for a single image. To construct meaningful word from those characters we need prior probablities of occurence of one character after another. This whole problem can be represented as a graphical model and maximum probable explanation(MAP) of the model gives the probable word the image is showing.
In given problem we have 1000 image ids and for each image id we have 10 characters and their factors. This factors represent the certainity with which each character is the content of that image. Transition factors between every two characters is given. Skip factor is given if two same image ids means same characters which is 5.0 otherwise 1.0. Given these factors, the probability to the character variables of a word w according to model will be given by:

$$p(chars) = \frac{1}{z}\begin{pmatrix} \prod\limits_{\forall i} & \psi_o\big(img(i), char(i)\big) \\ \prod\limits_{i=0,\ldots,|w|-2} & \psi_t\big(char(i), char(i+1)\big) \\ \prod\limits_{i,j|img(i)=img(j)} & \psi_s\big(char(i), char(i)\big) \end{pmatrix}$$

where ψ_o is OCR factor, ψ_t is transition factor and ψ_s is skip factor.

2 Solution Approach

2.1 Undirected Graphical Model

Assume a word consisting n characters and it formed of n images. Graphically this can be represented as below

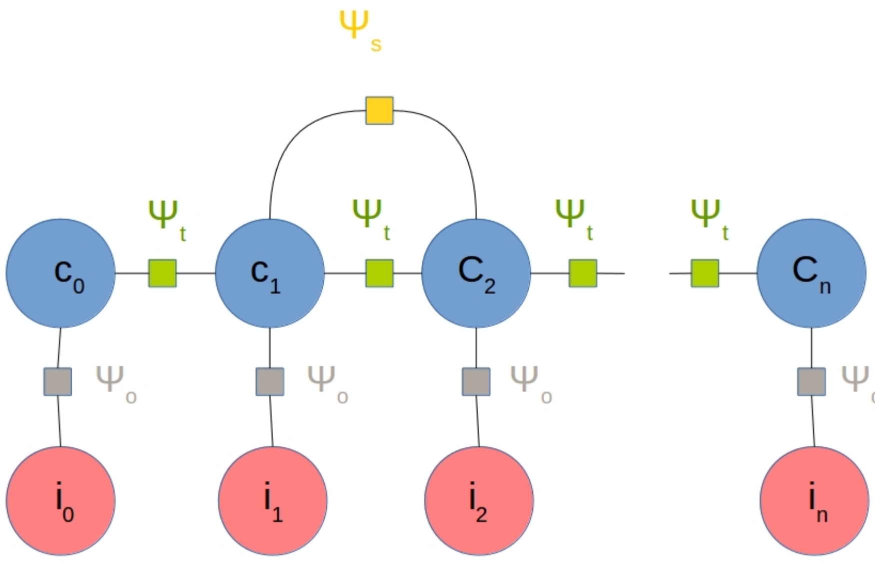

Figure 2.1: OWR problem as Undirected Graphical Model

2.1.1 OCR Model

OCR model contains only OCR factors i.e. ψ_o. Most probable explanation can be calculated as,

$$\begin{aligned}\tilde{p}(chars) &= \operatorname*{argmax}_{c_0,c_1,...c_n} \tilde{p}(c_0, c_1, ...c_n)\\ &= \operatorname*{argmax}_{c_0,c_1,...c_n} \Big(\prod_{\forall i} \psi_o\big(img(i), char(i)\big)\Big)\end{aligned}$$

Partition function, z can be calculated as,

$$z = \sum_{\psi_o} \Big(\prod_{\forall i} \psi_o\big(img(i), char(i)\big)\Big)$$

Pobability of word can be calculated as,

$$p(chars) = \frac{1}{z}\tilde{p}(chars)$$

2.2 Transition Model

Transition model contains both OCR factors and transition factors. Most probable explanation can be calculated as,

$$\begin{aligned}\tilde{p}(chars) &= \underset{c_0, c_1, \ldots c_n}{\operatorname{argmax}} \tilde{p}(c_0, c_1, \ldots c_n, i_0, i_1, \ldots, i_n) \\ &= \underset{c_0, c_1, \ldots c_n}{\operatorname{argmax}} \Big(\prod_{\forall i} \psi_o\big(img(i), char(i)\big) \prod_{i=0,\ldots,|w|-2} \psi_t\big(char(i), char(i+1)\big)\Big)\end{aligned}$$

Partition function, z can be calculated as,

$$z = \sum_{\psi_o \psi_t} \Big(\prod_{\forall i} \psi_o\big(img(i), char(i)\big) \prod_{i=0,\ldots,|w|-2} \psi_t\big(char(i), char(i+1)\big)\Big)$$

Pobability of word can be calculated as,

$$p(chars) = \frac{1}{z}\tilde{p}(chars)$$

2.3 Combined Model

Combined model contains OCR factors, transition factors and Skip factors. Most probable explanation can be calculated as,

$$
\begin{aligned}
\tilde{p}(chars) &= \underset{c_0,c_1,\ldots c_n}{\operatorname{argmax}} \tilde{p}(c_0, c_1, \ldots c_n, i_0, i_1, \ldots, i_n) \\
&= \underset{c_0,c_1,\ldots c_n}{\operatorname{argmax}} \Big(\prod_{\forall i} \psi_o\big(img(i), char(i)\big) \prod_{i=0,\ldots,|w|-2} \psi_t\big(char(i), char(i+1)\big) \\
&\quad \prod_{i,j|img(i)=img(j)} \psi_s\big(char(i), char(i)\big) \Big)
\end{aligned}
$$

Partition function, z can be calculated as,

$$
\begin{aligned}
z = \sum_{\psi_o \psi_t \psi_s} &\Big(\prod_{\forall i} \psi_o\big(img(i), char(i)\big) \prod_{i=0,\ldots,|w|-2} \psi_t\big(char(i), char(i+1)\big) \\
&\prod_{i,j|img(i)=img(j)} \psi_s\big(char(i), char(i)\big) \Big)
\end{aligned}
$$

Pobability of word can be calculated as,

$$
p(chars) = \frac{1}{z} \tilde{p}(chars)
$$

3 Results Obtained

Exhaustive inference can be done by calculating probabilities for all possible words for given set of images. To find most probable explanation(MPE) or MAP, juntion tree algorithm is used. Python has a pyMNS library which has JTA implimentation [1].

Model	Char Accuracy	Word Accuracy	Log Likelihood
OCR	0.53921	0.08653	-7.54175
Transition	0.66274	0.259615	-6.88575
Combined	0.71176	0.355769	-6.102387

Table 3.1: Model Accuracy

Table 3.1 shows character-wise accuracy, word-wise accuracy and average dataset log-likelihood for all three models. From the results we can see that combined model performs best out of the three models and OCR model performs the worst.

4 Further Fun

Varying the potentials has no effect on the word predicted or the accuracy calulated. Exhaustive inference for OCR model can be calulated using matrix multiplication which is faster than generating all possible words and calculating probabilites for each of them.

4.1 Directed Model

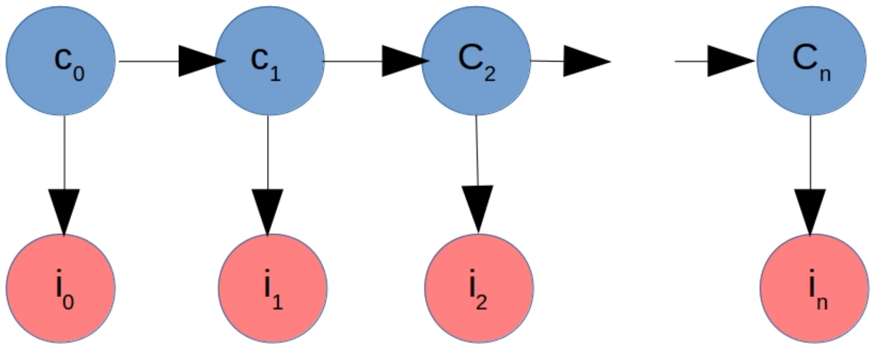

Figure 4.1: Directed Graphical Model

MAP can be calculated as,

$$\begin{aligned}\underset{c_0,c_1,...,c_n}{\operatorname{argmax}} P(c_0, c_1, ..c_n, i_0 = I_0, i_1 = I_1, ..., i_n = I_n) = &\underset{c_0,c_1,...,c_n}{\operatorname{argmax}} \Big(p(c_0)\, p(c_1 c_0) ... p(c_n | c_{n-1})\, p(i_0 = I_0 | c_0) \\ &p(i_1 = I_1 | c_1) ... p(i_n = I_n | c_n) \Big)\end{aligned}$$

MAP for bayesian network is found using libpgm python library [2].

5 Appendix

- *all_code.py* does exhaustive reference, finds model accuracy and print outcomes.
- *basian_creator.py* creates JSON file representing bayesian networks.
- *bayesian_solver.py* solves bayesian network and gives probability of finding true word given image ids as evidence.
- *parallel_code.py* is faster implimentation of OCR model using matrix multiplication.
- *furthe_fun.py* uses pyMNS to solve markov net using JTA [1].
- *bayes_net* contains all JSON files of bayesian networks.
- *dataset* contains datasets.
- *docs* contains report files.
- *probs* contains markov net files.

References

[1] Python Markov network solver library, `https://github.com/daveti/pymns`

[2] Python Bayesian network solver library, `http://pythonhosted.org/libpgm/`