MULTI-DIMENSIONAL DEPENDENCY-TREE HIDDEN MARKOV MODELS

Bernard Merialdo, Joakim Jiten, Benoit Huet

Institut Eurecom, Sophia Antipolis, France

ABSTRACT

In this paper, we propose a new type of multi-dimensional Hidden Markov Model based on the idea of Dependency Tree between positions. This simplification leads to an efficient implementation of the re-estimation algorithms, while keeping a mix of horizontal and vertical dependencies between positions. We explain DT-HMM and we present the formulas for the Maximum Likelihood re-estimation. We illustrate the algorithm by training a 2-dimensional model on a set of coherent images.

1. INTRODUCTION

Hidden Markov Models have proven to be particularly efficient at modeling uni-dimensional signal, in particular in Speech Recognition. In theory, HMM extend easily to multi-dimensional signals, such as image and video, but in practice, the full extension leads to a computational explosion which makes them intractable. Several restrictions have been proposed to simplify the model and reduce the computation, generally by reducing a 2D-model to a 1D-model, or an embedding of 1D-models. Among the first ones is [1] which uses a 1D HMM to model horizontal bands of face images. A more elaborate idea is to extract 1D features out of the image or video, and model these features with one or more 1D models. Another approach is to use a two-level model, called Embeded HMM or Hierarchical HMM, where a first high level model contains super-states associated to a low level HMM, which models the lines of the observed image [14]. The main disadvantage of these approaches is that they greatly reduce the vertical dependencies between states, since it is only achieved through a single super-state.

Finally several attempts have been done to heuristically reduce the complexity of the HMM algorithms by making simplifying assumptions which approximate the real algorithms:

- select a subset of state configurations only [8],
- ignore correlation of distant states [2],
- approximate probabilities by turbo-decoding [7].

The main disadvantage of these approaches is that they only provide approximate computations, so that the probabilistic model is no longer theoretically sound.

2. DEPENDENCY-TREE -HMM

In this section, we briefly recall the basics of 2D HMM and describe our proposed DT-HMM. More details can be found in [3].

The reader is expected to be familiar with 1D-HMM. We denote by $O=\{o_{ij}, i=1,\ldots m, j=1,\ldots,n\}$ a 2-dimensional observation, for example each o_{ij} may be the feature vector of a block (i,j) in the image. We denote by $Q = \{q_{ij}, i=1,\ldots m, j=1,\ldots,n\}$ a state assignment of the HMM, where the HMM is assumed to be in state q_{ij} at position (i,j) and produce the observation vector o_{ij}. If we denote by λ the parameters of the HMM, then, under the Markov assumptions, the joint likelihood of O and Q given λ can be computed as:

$$\begin{aligned} P(O,Q|\lambda) &= P(O|Q,\lambda)\,P(Q|\lambda) \\ &= \prod_{ij} p\left(o_{ij}\middle|q_{ij},\lambda\right)p\left(q_{ij}\middle|q_{i-1,j},q_{i,j-1},\lambda\right) \end{aligned}$$

If the set of possible states of the HMM is $\{s_1, \ldots s_N\}$, then the parameters λ are:

- the output probability distributions $p(o \mid s_i)$
- the transition probability distributions $p(s_i \mid s_j,s_k)$.

Depending on the type of output (discrete or continuous) the output probability distribution are discrete or continuous (typically a mixture of Gaussian distribution).

The problem with 2D-HMM is the double dependency of $q_{i,j}$ on its two neighbors, $q_{i-1,j}$ and $q_{i,j-1}$., which does not allow the factorization of computations as in 1D, and makes the computations intractable in practice.

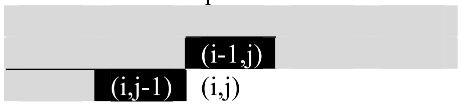


Figure 1. 2D Neighbors

Our idea is to assume that $q_{i,j}$ depends on one neighbor at a time only. This neighbor may be the horizontal or the vertical one, depending on a random variable t(i,j). More precisely, t(i,j) is a random variable with two possible values:

$$t(i,j) = \begin{cases} (i-1,j) & \text{with prob } 0.5 \\ (i,j-1) & \text{with prob } 0.5 \end{cases}$$

For the positions on the first row or the first column, t(i,j) has only one value, the one which leads to a valid position inside the domain. t(0,0) is not defined.

So, our model assumes the following simplification:

$$p(q_{i,j}|q_{i-1,j},q_{i,j-1},t) = \begin{cases} p_V(q_{i,j}|q_{i-1,j}) & \text{if } t(i,j) = (i-1,j) \\ p_H(q_{i,j}|q_{i,j-1}) & \text{if } t(i,j) = (i,j-1) \end{cases}$$

If we further define a "direction" function:

$$D(t) = \begin{cases} V & \text{if } t = (i-1,j) \\ H & \text{if } t = (i,j-1) \end{cases}$$

then we have the simpler formulation:

$$p(q_{i,j}|q_{i-1,j},q_{i,j-1},t) = p_{D(t(i,j))}(q_{i,j}|q_{t(i,j)})$$

Note that the vector **t** of the values t(i,j) for all (i,j) defines a tree structure over all positions, with (0,0) as the root. Figure 2 shows an example of such a Dependency Tree.

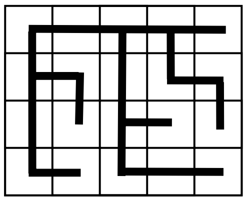


Figure 2. Example of Random Dependency Tree

The DT-HMM replaces the N^3 transition probabilities of the complete 2D-HMM by $2N^2$ transition probabilities. Therefore it is efficient in terms of storage. We will see that it is also efficient in terms of computation.

Position (0,0) has no ancestor. In this paper, we assume for simplicity that the model starts with a predefined initial state s_I in position (0,0). It is straightforward to extend the algorithms to the case where the model starts with an initial probability distribution over all states.

3. MAXIMUM LIKELIHOOD RE-ESTIMATION

If we assume that the Dependency Tree is fixed, then many computations become simple. In particular, we describe in this section the re-estimation of the model using Maximum Likelihood. The organization of the computation is directly inspired from the Inside-Outside algorithm [12] [13], which is the extension of the Baum-Welch algorithm for Probabilistic Context Free Grammars.

3.1. The Inside probabilities

Let us denote by $\beta_{i,j}(s)$ the probability that the portion of the image covered by T(i,j), the subtree with root (i,j), is produced from state s in position (i,j). The $\beta_{i,j}(s)$ can be calculated recursively, in reverse order (starting from the last position), following the relations:

- if (i,j) is a leaf in T(i,j):

$$\beta_{i,j}(s) = p(o_{i,j}|s)$$

- if (i,j) has only an horizontal successor:

$$\beta_{i,j}(s) = p(o_{i,j}|s)\sum_{s'} p_H(s'|s)\beta_{i,j+1}(s')$$

- if (i,j) has only a vertical successor:

$$\beta_{i,j}(s) = p(o_{i,j}|s)\sum_{s'} p_V(s'|s)\beta_{i+1,j}(s')$$

- if (i,j) has both an horizontal and a vertical successors:

$$\beta_{i,j}(s) = p(o_{i,j}|s)\left(\sum_{s'} p_H(s'|s)\beta_{i,j+1}(s')\right)\left(\sum_{s'} p_V(s'|s)\beta_{i+1,j}(s')\right)$$

The probability that the complete observation is produced by the model is then:

$$P(O|\,t) = \beta_{0,0}(s_I)$$

3.2. The Outside Probabilities

Let us denote by O(i,j) the portion of the image which is not covered by the subtrees starting at the successors of (i,j). For example, if (i,j) has two successors, O(i,j) is the portion of the image outside the subtrees T(i+1,j) and T(i,j+1). { O(i,j), T(i+1,j), T(i,j+1)} is a partition of the image, as shown in Figure 3.

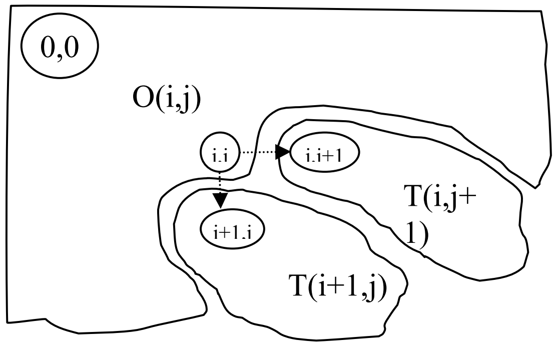


Figure 3. Schema of Outside O(i,j) and Inside T(i,j) areas

If we denote by $\alpha_{i,j}(s)$ the probability of starting from (0,0), generating the output vectors for all the positions in O(i,j), and reaching position (i,j) in state s, then the $\alpha_{i,j}(s)$ probabilities may be computed by the following recursions:

- $\alpha_{0,0}(s_I) = p(o_{0,0}|s_I)$, $\alpha_{0,0}(s) = 0$ for $s \neq s_I$,

In general, (i,j) may have two descendants, so that the outside probability for a descendant has to include the inside probability of the subtree for the other descendant:

- if (i,j+1) has an horizontal ancestor (i,j):

$$\alpha_{i,j+1}(s) = p(o_{i,j+1}|s)\sum_{s'} \alpha_{i,j}(s')p_H(s|s')\sum_{s''} p_V(s''|s')\beta_{i+1,j}(s'')$$

(in this case, O(i,j+1) is the union of O(i,j), (i,j) and T(i+1,j)),

- if (i+1,j) has a vertical ancestor (i,j):

$$\alpha_{i+1,j}(s) = p(o_{i+1,j}|s)\sum_{s'} \alpha_{i,j}(s')p_V(s|s')\sum_{s''} p_V(s''|s')\beta_{i,j+1}(s'')$$

(in this case, O(i+1,j) is the union of O(i,j), (i,j) and T(i,j+1)).

These formulas simplify in the case where the ancestor (i,j) has only one successor:

- if (i,j+1) is the horizontal successor:

$$\alpha_{i,j+1}(s) = p(o_{i,j+1}|s)\sum_{s'}\alpha_{i,j}(s')p_H(s|s')$$

- if (i+1,j) is the vertical successor:

$$\alpha_{i+1,j}(s) = p(o_{i+1,j}|s)\sum_{s'}\alpha_{i,j}(s')p_V(s|s')$$

3.3. Maximum Likelihood training

In Maximum Likelihood training, we need to estimate the number of times that a particular transition, or a particular output, has been used during the generation of the complete observation. Using the inside and outside probabilities defined previously, we can observe that the probability of generating the complete image is:

$$P(O|t) = \sum_s \alpha_{i,j}(s)\left(\sum_{s'} p_H(s'|s)\beta_{i,j+1}(s')\right)\left(\sum_{s''} p_V(s''|s)\beta_{i+1,j}(s'')\right)$$

for any position (i,j) with two successors. Let us define:

$$\gamma^H_{i,j}(s) = \sum_{s'} p_H(s'|s)\beta_{i,j+1}(s')$$

$$\gamma^V_{i,j}(s) = \sum_{s''} p_V(s''|s)\beta_{i+1,j}(s'')$$

Then:

$$P(O|t) = \sum_s \alpha_{i,j}(s)\gamma^H_{i,j}(s)\gamma^V_{i,j}(s)$$

Note that if we define $\gamma^H_{i,j}(s)$ (respectively $\gamma^V_{i,j}(s)$) to be equal to 1 when (i,j) has no horizontal (respectively vertical) successor, then this formula is valid for all positions (i,j), whatever the number of successors.

The probability for being in state s at position (i,j) while generating the complete image is therefore:

$$P(q_{i,j} = s|O,t) = \frac{1}{P(O|t)}\alpha_{i,j}(s)\gamma^H_{i,j}(s)\gamma^V_{i,j}(s)$$

The expected number of times that the system is in state s during the generation of the image is:

$$E(s) = \sum_{i,j} P(q_{i,j} = s|O,t) = \frac{1}{P(O|t)}\sum_{i,j}\alpha_{i,j}(s)\gamma^H_{i,j}(s)\gamma^V_{i,j}(s)$$

The probability for going from state s at position (i,j) to state s' in the successor position (i,j+1) while generating the complete image is:

$$P(q_{i,j} = s, q_{i,j+1} = s'|O,t) = \frac{1}{P(O|t)}\alpha_{i,j}(s)p_H(s'|s)\beta_{i,j+1}(s')\gamma^V_{i,j}(s)$$

The expected number of times that the horizontal transition from s to s' takes place during the generation of the image is:

$$E(s \xrightarrow{H} s') = \sum_{i,j} P(q_{i,j} = s, q_{i,j+1} = s'|O,t) = \frac{1}{P(O|t)}\sum_{i,j}\alpha_{i,j}(s)p_H(s'|s)\beta_{i,j+1}(s')\gamma^V_{i,j}(s)$$

This provides the basis for the reestimation of the horizontal transition probabilities:

$$p'_H(s'|s) = \frac{E(s \xrightarrow{H} s')}{\sum_{s''} E(s \xrightarrow{H} s'')}$$

The vertical transitions are handled in the same way. The reestimation of the output probabilities is similar, as the probability of being in state s at (i,j) is also the probability of emitting the output vector $o_{i,j}$ from state s at position (i,j). When the output probability distribution is discrete, the reestimate is obtained by counting:

$$p'(o|s) = \frac{E(s \to o)}{\sum_{o'} E(s \to o')}$$

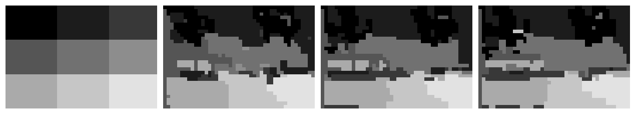
When the distribution is continuous, the expected number of times that the emission is observed can be used to update the parameters of the distribution (in the case of a Mixture of Gaussian distributions, weights, means and variances).

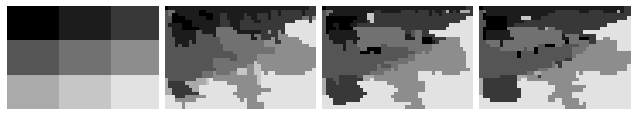
4. EXPERIMENTATION

As an example, we have applied our algorithm for the joint segmentation of a set of images. Each image is split into blocks of 8x8 pixels, and the observation vector for each block is computed as the average and variance of the YIQ coding. We consider a 9 states model, where each output probability is a Gaussian Mixture Model with 2 components. The initial model is constructed with an initial alignment where one state covers one region of a regular 3x3 grid over the image. A Viterbi algorithm allows displaying the best sequence of states at each position, therefore showing how the model matches each image.

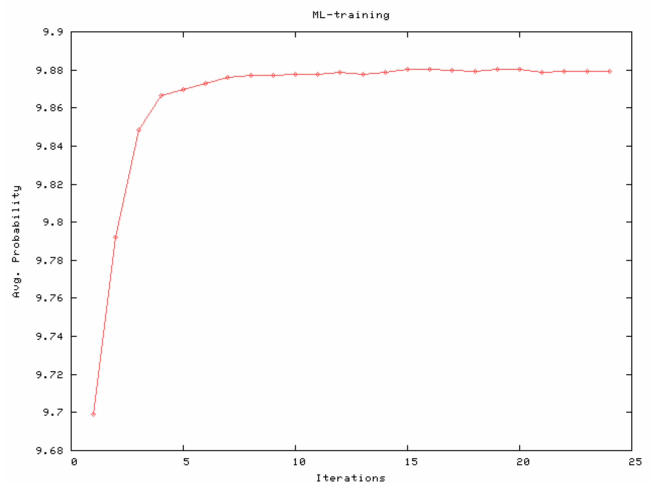
The training set contains two images:

During training, we can observe the state assignment at each iteration, as an indication of the way the model fits the training data. For example, the first four iterations on the training images provide the following assignments:

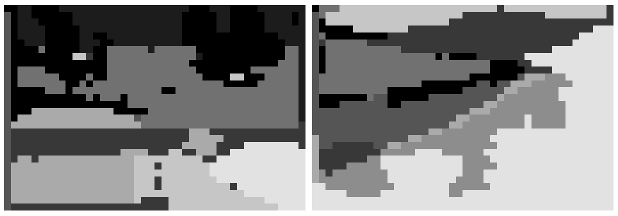




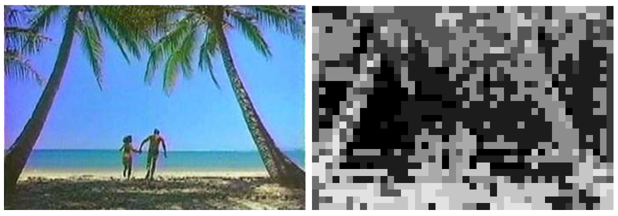
The graph below shows the evolution of likelihood of the training data during the training iterations:



After 25 iterations, we get the following state assignments for the training images:



It should be emphasized that this is not just a simple segmentation of the images, but that each region is also assigned one of the 9 states of the model, and that the definition of those states has been done taking into account all training data simultaneously.
Once the model is trained, we can also apply it on new images. Below is an example of the state assignement for a new image:



Note that those regions are also labeled with the same 9 states as the training data.

6. CONCLUSION

In this paper, we have only presented the basic mechanism for the DT-HMM and some initial experiments. As with regular 1D-HMM, the potential usages are very numerous. Training on a set of images provides a model for a class of data. The probability of emitting a new image can be used as a measure for the probability that it belongs to this class. The Viterbi alignment can be used to identify components within the image. Furthermore, the DT-HMM has several interesting theoretical extensions, for example by using several trees for the same image, considering other ancestors scheme than just horizontal-vertical, and extending to more than 2 dimensions. We plan to explore many of these issues in our further work.

7. REFERENCES

[1] Samaria, Ferdinando and Fallside, *Frank Face Identification and Feature Extraction Using Hidden Markov Models*, Image Processing: Theory and Applications, Elsevier, 1993, pp 295-298

[2] Merialdo, B.; Marchand-Maillet, S.; Huet, B.; Approximate Viterbi decoding for 2D-hidden Markov models, IEEE ICASSP, Volume 6, 5-9 June 2000 Page(s):2147 - 2150 vol.4

[3] Merialdo, B; Dependency Tree Hidden Markov Models, Research Report RR-05-128, Institut Eurecom, Jan 2005

[4] Rabiner, L.R.; A tutorial on hidden Markov models and selected applications in speech recognition Proceedings of the IEEE, Volume 77, Issue 2, Feb. 1989 Page(s):257 - 286

[5] Othman, H.; Aboulnasr, T.; A simplified second-order HMM with application to face recognition. IEEE International Symposium on Circuits and Systems, Volume 2, 6-9 May 2001 Page(s):161 - 164

[6] Wallhoff, F.; Eickeler, S.; Rigoll, G.; A comparison of discrete and continuous output modeling techniques for a pseudo-2D hidden Markov model face recognition system , ICIP, Volume 2, 7-10 Oct. 2001 Page(s):685 - 688

[7] Perronnin, F.; Dugelay, J.-L.; Rose, K.; Deformable face mapping for person identification, International Conference on Image Processing, Volume 1, 14-17 Sept. 2003 Page(s):I - 661-4

[8] J. Li, A. Najmi, and R. M. Gray, Image classification by a two-dimensional hidden markov model, IEEE Trans. Signal Processing, vol. 48, no. 2, pp. 517–533, 2000.

[9] Levin, E.; Pieraccini, R.; Dynamic planar warping for optical character recognition, IEEE ICASSP, , Volume 3, 23-26 March 1992 Page(s):149 - 152

[10] LE. Baum and T. Petrie, Statistical Inference for Probabilistic Functions of Finite State Markov Chains, Annual Math., Stat., 1966, Vol.37, pp. 1554-1563.

[11] M. A. Mohamed and P. Gader, Generalized Hidden Markov Models-Part I: Theoretical Frameworks, IEEE Transaction on Fuzzy Systems, February, 2000, Vol.8, No.1, pp. 67-81.

[12] J.K. Baker, Trainable grammars for speech recognition ,; In Jared J.Wolf and Dennis H. Klatt, editors, Speech communication papers presented at the 97th Meeting of the Acoustical Society of America, MIT, Cambridge, MA, June 1979.

[13] F. Jelinek, *Statistical Methods for Speech Recognition* Cambridge, MA: MIT Press, 1997.

[14] S. Fine, Y. Singer, N. Tishby, "The hierarchical hidden Markov model: Analysis and applications," Machine Learning 32(1998).