Modeling the Visual Cortex: Object Recognition with Extended Hierarchical Bayesian Networks

Theresa Q. Vu
Master's Project Report
Department of Computer Science, Brown University
Advisor: Tom Dean

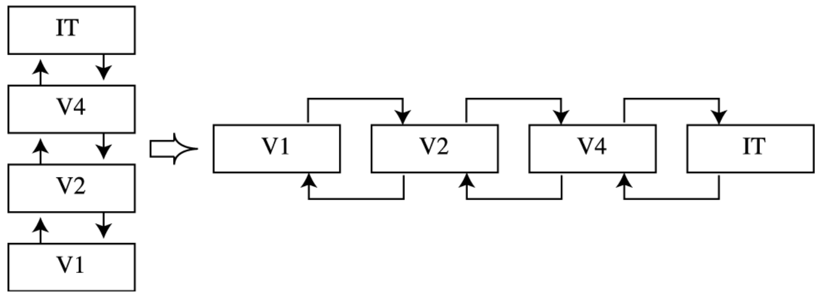
Dec 15, 2006

1 Introduction

1.1 The Visual Cortex

The visual cortex, as the name implies, is the division of the human brain responsible for the processing of vision. It consists of several interconnected visual areas, whose information flow is organized in a rough hierarchy. At the lowest level is the primary visual cortex, V1, which receives light intensity information from the retina via the optic nerve and lateral geniculate nucleus. V1 transmits information through two primary pathways known as the ventral stream and the dorsal stream. The ventral stream, associated with form and object recognition, starts with V1 and transmits information through visual area 2 (V2), then visual area 4 (V4), followed by the inferior-temporal lobe (IT). The dorsal stream also begins at V1, and goes through V2, but diverges afterwards to the dorsomedial area and visual area MT [6].

Below is Tom Dean's graphical representation of information relay in the visual cortex [1].

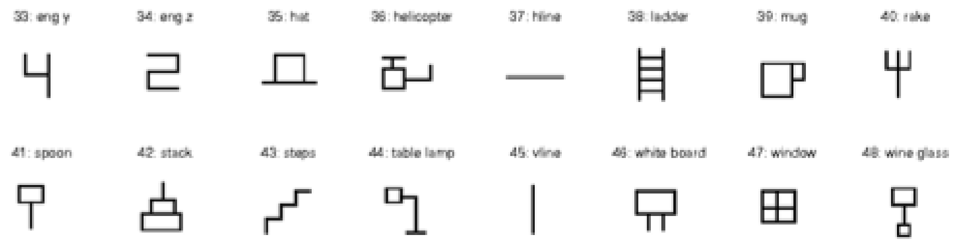


1.2 The Computational Model

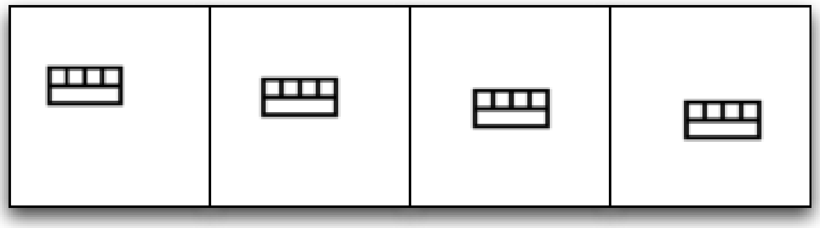
The computational model used in this project is a temporally extended Hierarchical Bayesian Network (HBN). A Hierarchical Bayesian Network is a graphical model whose structure parallels the hierarchical nature of the visual cortex. The nodes in the lowest level of the HBN receive evidence in the form of image pixels, analogous to the ”input” of light intensities to the human retina. At each increasing level in the HBN, a node's receptive field becomes recursively larger making it possible to capture increasingly abstract features [1] until the hierarchy is traversed in full, where upon the output node is used to classify the object. Lee and Mumford [7] first proposed the idea of using a hierarchical Bayesian inference to model the role of feedback and feedforward information between the different visual areas. The temporal extension of the HBN was added as a means of capturing invariant structures in sequence of images [2]. These structures are described in more detail in Section 2.

1.3 Evaluation of the Model

The model will be evaluated using an object recognition task. The dataset is a collection of simple pictographs created by Dileep George [5]. Some examples of the pictograph collection are shown below.



The model will be trained on the original pictographs, and tested with temporal sequences of pictographs to represent movement. The test will elucidate whether the model can learn the invariant structures within the sequences and in turn correctly recognize the object regardless of its position. The temporal sequences are created by specifying the number of elements in the sequence (how many slices of time the sequence captures), the direction of movement (north, south, east, west, northeast, northwest, southeast, southwest), and the velocity of movement (in pixels translated per sequence) Below is an example of a four slice temporal sequence moving 3 pixels per slice in the southeast direction.



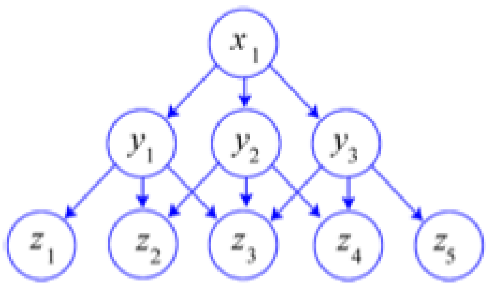
1.4 Collaborative Effort

This project was advised by Professor Tom Dean, and completed in collaboration with Ethan Schreiber. Ethan implemented the infrastructure for distributively processing the model. I implemented the internal structure and the learning of the model. The integration of our modules was done in tandem.

2 Hierarchical Bayesian Networks

2.1 Overall Structure

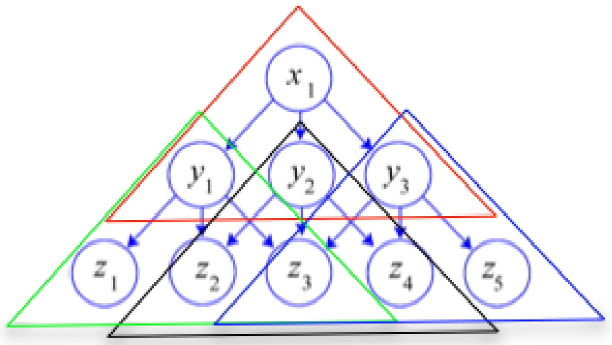
A Hierarchical Bayesian Network is a graphical model arranged into K layers. Each layer in the HBN contains inter-edges, an edge which connects nodes of level K with nodes in the K-1 layer, and intra-edges, edges that connect laterally within the same layer. Figure 2 depicts a simple Hierarchical Bayesian Network [2].



Performing inference on a simple network as the one above is a minor computation, but inference upon a network on a cortical scale poses major algorithmic challenges. In order to mitigate computational intensity, it is necessary to break the network into sub-networks – henceforth referred to as subnets [2].

2.2 Subnet Decomposition

A subnet is a specially grouped set of nodes spanning two levels of an HBN. The group of nodes consists of one node in level k, and one or more of its children in level $k-1$. The k level node is referred to as the hidden node of the subnet; the $k-1$ level nodes is the receptive field of the subnet. Separate subnets may have overlapping receptive fields, but never overlapping hidden nodes. Hidden nodes are unique to their subnet, and are in fact used as subnet IDs in the distributive implementation of the model.



Each subnet outputs a *feature* which encompasses the evidence observed in its receptive fields. This feature is obtained from the subnet's local belief function, which computes the distribution over the subnet's hidden node [2].

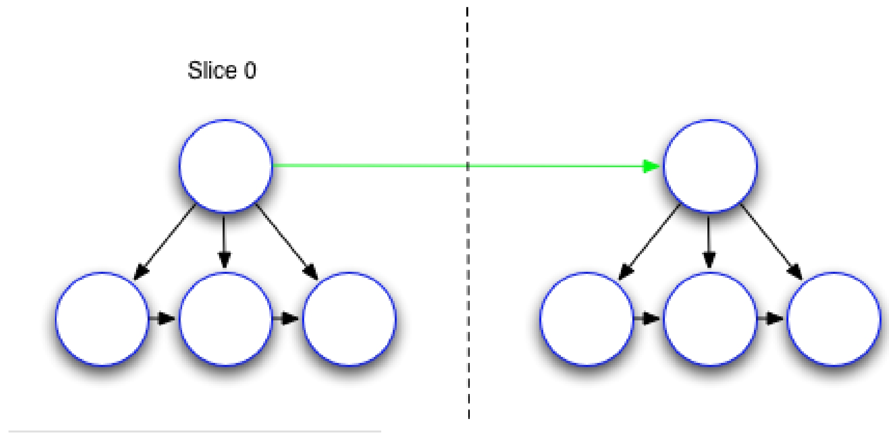
2.3 Temporal Extension

To capture invariant features of temporal sequences, we extend the computational model to account for time. One way to do this is to create one large Markov model where each time slice is represented by an entire HBN, and the state variables its nodes. This implementation can be quite cumbersome computationally, so once again we make the problem more tractable by using subnets. Instead of reproducing the entire HBN for each time slice, we embed a smaller hidden Markov model into each subnet. This is discussed in more detail in the following section.

3 Subnets

3.1 Internal Structure

The internal structure of a subnet is an embedded hidden Markov model. If the number of time slices specified in the model is n, the local subnet graph is replicated $n-1$ times to represent the Markov process. Below is a diagram of a two timeslice subnet, where the dotted line separates the distinct time slices.



3.2 Learning Subnets

Learning a subnet involves learning the structure and parameters of its embedded hidden Markov model. An inference engine is needed (whether performing learning or inference) and we use the exact junction tree algorithm as our engine. Because subnets were designed to be tractable, they contain a manageable number of nodes so an exact inference is perfectly feasible.

The basic algorithms is as follows:

1. Construct and learn a temporary observation model.

   We first build a temporary observation model that emulates the actual model we are trying to learn. The observation model is necessary for generating data crucial to later steps of the subnet learning algorithm. Our observation model takes the form of a tree-augmented naive Bayes net [4] in order to capture the intra-level dependencies among the subnet's receptive field. Using the expectation maximization (EM) algorithm we then learn the parameters of the observation model.

2. Learn a prediction suffix tree from generated observations.

   The observation model allows us to generate the data necessary for learning a Prediction Suffix Tree (PST). Since our computational model of the visual cortex is given picture evidence in sequences, we want to calculate the probability distribution for the next possible symbols in the sequence, given the preceding subsequence. This can be captured using a PST.

3. Construct a probabilistic finite automaton from the prediction suffix tree.

   After learning the PST, we convert it into a probabilistic finite automaton (PFA) using an algorithm proposed by Sage and Buxton. [9]. The purpose of the PFA is to provide the state transition probabilities necessary to create the subnet's embedded variable order hidden Markov model.

4. Build the embedded variable-order hidden Markov model.

   Now that the PFA has been constructed, we build the embedded variable-order hidden Markov model. We use the length of the PFA to determine the state space size of the subnet's hidden node. We then extract the state transition probabilities from the PFA, reshape them to fit the format for a conditional probability table (CPT), and install the new CPT into the subnet's hidden node. Next we map the composite variable duration states in the PFA to atomic states. We extract from the composite-state to primitive-state mapping, and similarly to the method above, install the probabilities into the model.

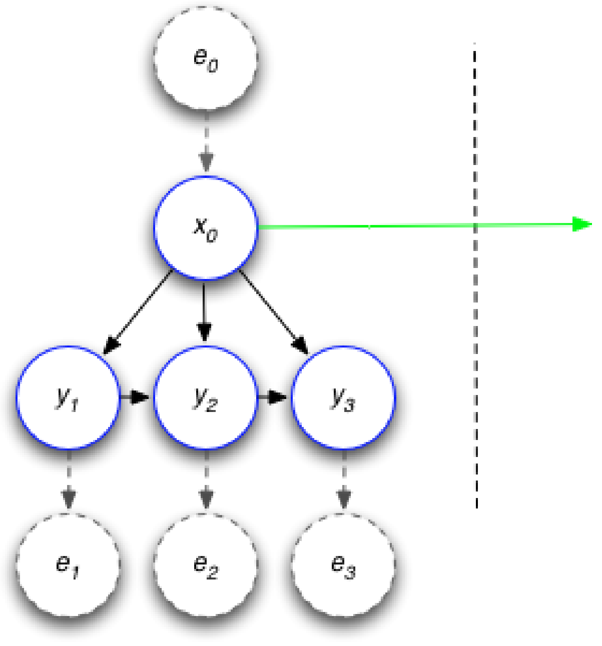
5. Use EM to refine the parameters of the embedded Markov model.

   Finally with the embedded Markov model created, we refine its parameters using a smoother engine for learning dynamic (temporal) Bayesian networks.

3.3 Subnet Potentials

Subnet Potentials are the means in which subnets communicate with each other. These potentials are passed into and out of a subnet from clusters of variables within the larger Hierarchical Bayesian Network. Subnet potentials take the form of likelihood (soft) evidence and are incorporated into the junction tree during belief propagation on the larger HBN.

The diagram below, which shows one slice of a subnet graph, illustrates the passing of subnet potentials.



The nodes in blue are the actual nodes within a subnet, whereas the nodes in gray are not a part of the network – they represent the likelihood evidence passed into the subnet. Since the subnet is a part of a much larger network, we can alternatively think of these gray nodes as lambda and pi messages [2] in the Pearl belief propagation algorithm [8]. In Pearl's belief propagation, the pi term refers to messages sent to a node from its parents, and lambda refers to messages sent to a node from its children.

Further detail regarding the message passing between subnets can be found in Ethan Schreiber's report for the distributed processing of this model.

4 Implementation

The computational model was implemented in C++ using a modified version of Intel's Open-Source Probabilistic Networks Library (PNL) and Chris Chin's adaptation of Kingsley Sage's variable length markov model codebase.

4.1 PNL and Modifications

Intel's PNL [3] is an open source graphical models library. It was used for creating static and dynamic Bayesian networks, as well as performing junction tree inference and expectation maximization (EM) parameter learning within subnets. To implement our computational model however, several modifications to PNL had to be made.

The primary modification centered around soft evidence because PNL itself only handles hard evidence. To pass along soft evidence, the PNL class CEvidence was modified to contain a variable *subPots*, which is an NxM matrix of floats where N is the number of nodes in the Bayes Net, and M is the state space of the nodes. These subnet potentials are passed alongside dummy hard evidence, where they are loaded in lieu of hard evidence during the learning stage of the subnet. This will alter the underlying Bayes net by modifying its factors to reflect the soft evidence (subnet potentials). A modification was made in CTabularDistribFun::ShrinkObservedNodes() which is called during the junction tree algorithm to load the subnet potentials instead of the hard evidence. All subnet potential modifications had to be made for both static and dynamic bayesian net classes and inference engines.

The other modifications to PNL include modifying the inference engines to calculate and return the loglikelihood of its evidence. The loglikelihoods were used to assign priors to a subnet's hidden node.

5 Discussion and Future Work

Object Recognition is a vastly studied field with many successes and many more challenges. Acknowledging that no computer process has yet come close to matching the human brain's adeptness at image recognition, how should we use the intrinsic qualities of the visual cortex to improve upon our models? This particular model focuses on the hierarchy of message passing between the separate areas of the visual cortex by using a Hierarchical Bayesian network as it's structure. The larger HBN is then decomposed into smaller subnets to alleviate computational intensity and also open the door to distributively processing the learning and computation of the network. Currently Tom Dean along with Glenn Caroll, Rich Washington, and Jim Lloyd of Google have taken the code for this computational model and are working to deploy it on the scale of thousands of processors. This should drastically improve the scale and speed of tasks to be performed on the computational model, making it possible to test recognition on more detailed images and more complex sequences. In addition, having this model run on a distributed parallel network would further liken it to the distributed parallel processing that occurs in the human brain.

Acknowledgements

Thanks foremost to Professor Tom Dean for his guidance throughout the completion of this project and the course of my graduate studies. I could not have asked for a more knowledgeable, patient, and personable advisor. Thanks also to Ethan Schreiber, for being my research collaborator and surrogate older brother at Brown. Further thanks to Chris Chin, for his PST/PFA code and his help integrating the code, Dileep George for his pictograph dataset, and Glenn Caroll for his bug finds and fixes.

References

[1] T. Dean, *A computational model of the cerebral cortex*, Proceedings of Twentieth National Conference on Artificial Intelligence, AAAI-05 (2005), pp. 938–943.

[2] T. Dean, *Spatially-extended hierarchical markov models.* Submitted to NIPS, 2006.

[3] V. Eruhimov, D. Dash, G. Bradski, and K. Murhpy. http://www.intel.com/technology/computing/pnl/.

[4] N. Friedman, D. Geiger, and M. Goldszmidt, *Bayesian network classifiers.*, Machine Learning, 29 (1997), pp. 131–163.

[5] D. George and J. Hawkins, *A hierarchical bayesian model of invariant pattern recognition in the visual cortex.*, Proceedings of the International Joint Conference on Neural Networks, 3 (2005), pp. 1812–1817.

[6] M. Goodale and A. Milner, *Separate pathways for perception and action.*, Trends in Neuroscience, 15 (1995), pp. 20–25.

[7] T. S. Lee and D. Mumford, *Hierarchical bayesian inference in the visual cortex*, Journal of the Optical Society of America, 2 (2003), pp. 1434–1448.

[8] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.*, Morgan Kaufmann, 1998.

[9] K. Sage and H. Buxton, *Joint spatial and temporal structure learning for task based control*, International Conference on Pattern Recognition, 2 (2004), pp. 48–51.