Deep Woods

Yoshua Bengio, Joseph Turian, and Hugo Larochelle
Dept. IRO, Université de Montréal, C.P. 6128, Montreal, Qc, H3C 3J7, Canada
{*bengioy,turian,larocheh*}@*iro.umontreal.ca*
http://www.iro.umontreal.ca/∼lisa

Abstract

Recent theoretical studies indicate that deep architectures [4, 2] are needed to *efficiently* model complex distributions and to achieve better generalization performance on challenging recognition tasks. The belief that additional levels of functional composition will yield increased representational and modeling power is not new [13, 9, 16]. In practice, learning in deep architectures had proven to be difficult. However, this situation recently changed with the successful approaches of [7, 10, 1, 14, 12] for training Deep Belief Networks and stacked autoencoders (deep neural networks). What these works have in common is several principles for effectively learning deep architectures. In our work we attempt to generalize these learning principles to deep architectures in which each level is represented by an ensemble of decision trees. "Deep woods" are what we call the resulting stacked ensembles of trees. We introduce two new algorithms for learning of deep woods: one is inspired by the deterministic autoencoders which have been used to build deep neural networks, and one is a generalization of Restricted Boltzmann Machines, which have been used to build Deep Belief Networks.

Introduction

Theoretical results suggest that, in order to learn the kind of complicated functions that can represent high-level abstractions (e.g. for vision, language, and other AI-level tasks), one needs *deep architectures* [7, 4, 2]. Deep architectures are composed of multiple levels of non-linear operations, such as neural nets with many hidden layers or complicated propositional formulae re-using many sub-formulae. The depth of an architecture is the depth of the circuit that represents this function [2, 4].

Searching the parameter space of deep architectures is a difficult optimization task, but recently proposed learning algorithms—such as those for Deep Belief Networks [7]—can successfully train deep architectures. The resulting deep machines can beat the shallow state-of-the-art in certain areas. A principle that has been found to help optimizing deep networks is based on the use of unsupervised learning to initialize each layer in the network. Each layer learns a representation of its input that serves as input for the next layer, and training progresses by greedily [7]. The principle of training a deep architecture by greedy layer-wise unsupervised training has been shown to be successful for deep connectionist architectures [7, 10, 1, 14, 12]. Does this deep training principle apply to deep architectures comprising other kinds of primitive units?

We attempt to exploit this principle to develop new deep architectures based on deterministic or stochastic decision trees. Decision trees are local estimators in the sense of relying on a partition of the input space and using separate parameters for each region [3]. For this reason, single decision trees need at least as many training examples as there are variations of interest in the target function, and they cannot generalize to new variations not covered in the training set. On the other hand, ensembles of trees (like boosted trees [6], and forests [11, 5]) are more powerful than a single tree. They add a level to the architecture, which allows the model to discriminate among a number of regions *exponential in the number of parameters* [3]. By analogy, clustering forms a single partition and generally involves a loss of information about the input, whereas a multi-clustering provides a *set* of separate partitions of the input space. If we identify for an input example in which region of each partition it belongs, we have a description of the input pattern which might be very rich, possibly not losing any information. The tuple of symbols specifying to which region of each partition the input belongs can be seen as a transformation of the input into a distributed representation [8], where the statistical structure of the data and the factors of variation in it could be disentangled. This corresponds to the kind of partition of the input space that an ensemble of trees can represent: the distributed representation output at

each level transforms the input into a binary code representing the leaves (and perhaps internal nodes) in the ensemble into which the input falls, with one bit per node.
Motivated by the above considerations, we propose two approaches to train a deep architecture where each level is an ensemble of trees. The first approach is analogous to autoencoders, which compute a deterministic reconstruction and minimize a reconstruction error, while the second approach is analogous to Restricted Boltzmann Machines (RBMs), which build a generative model of the input through a discrete hidden variable. In both cases, a global training criterion is defined which pushes the different trees in the ensemble to take on complementary roles in explaining the input. Consider a binary split $c_k(x)$ for decision node k of the ensemble. c_k can be a deterministic or stochastic function of the input pattern x. A node or leaf j, or equivalently a region associated with one of the decision trees, is defined by a conjunction of splits, i.e. a product of binary decisions of the form $c_k(x)$ or $1 - c_k(x)$. We denote $h_j(x)$ the binary (deterministic or stochastic) variable which indicates the presence of x in the corresponding region (or node of a tree) j. Note that whereas the c_k's can take arbitrary patterns of binary values, the tree structure imposes some constraints on the values of the h_j's, such that h_j can be 1 only if its parent (if any) has the appropriate binary value (depending on whether h_j is child 0 or child 1 of its parent). Let $code(j)_k$ be a binary value that specifies for node j whether node k (which should be an ancestor of j in j's tree) should be 1 or 0 for node j to be allowed to be non-zero, and let $ancestors(j)$ be the set of ancestors of j (i.e. nodes whose value constrains the value of h_j).

AutoEncoding Woods

In autoencoding forests, we associate with each leaf a reconstruction vector r_j, and we optimize both the reconstruction patterns and the node decision conditions so as to minimize a reconstruction error. The reconstruction for the whole forest is obtained by summing the reconstructions from each tree:

$$r(x) = \sum_i \sum_{j \in leaves(i)} r_j \prod_{k \in ancestors(j)} c_k(x)^{code(j)_k} (1 - c_k(x))^{1-code(j)_k}$$

For binary or binomial-like inputs we use as reconstruction error the cross-entropy between $sigmoid(r(x))$ and the actual input pattern x. For continuous-valued inputs we use the squared error $||r(x) - x||^2$. In both cases these can be seen as $-\log P(x|r(x))$ under some model.
Adding decision nodes and trees is achieved using a construction heuristic that chooses splits greedily to maximize the gradient of the objective, similarly to a method used for supervised decision trees [15]. These heuristics are based on the traditional axis-aligned decision nodes, where only one of the input variables is used for each particular decision function $c_k(x)$, i.e. $c_k(x) = 1_{x_{d(k)} < \theta_k}$, where $d(k)$ selects an input dimension for decision k.

RBM Woods

We construct a joint distribution for h and the input vector x through an energy function similar to the one proposed for RBMs:

$$E_\lambda(x, h) = -b'x - c'h - h'Wx + \lambda \sum_i \sum_{j \in leaves(i)} h_j (1 - \prod_{k \in ancestors(j)} h_k^{code(j)_k} (1 - h_k)^{1-code(j)_k})$$

As $\lambda \to \infty$ the λ-weighted sum just imposes the constraint that only one leaf per tree is 1 and that it can be 1 only if its ancestors have the appropriate value. One can then decompose $P(h|x)$ as a product $\prod_i P(h^i|x)$ where h^i only contains the bits for tree i, and each $P(h^i|x)$ can be expanded according to the tree structure as a product of conditionals of the form $P(h_j|\{h_k : k \in ancestors(j)\}, x)$. Each of these conditionals turn out to be of the usual affine-sigmoid form $sigmoid(b_j + W_j' x)$. We also obtain that $P(h_j|x)$ has the form of a product of $sigmoid(b_k + W_k' x)$ or $(1 - sigmoid(b_k + W_k' x))$ with k an ancestor decision of j, which is a stochastic version of a decision tree with decisions $c_k \sim sigmoid(b_k + W_k' x)$ on the nodes. We can therefore easily sample from $P(h|x)$, as well as compute these probabilities for any given h vector, at a cost proportional to the sum of the depths of the trees. $P(x|h)$ does not depend on the tree constraints and is computed as usual in RBMs. The contrastive divergence algorithm or other approximation of the log-likelihood gradient can therefore be applied to this particular model. All the parameters of the RBM Woods can be trained by stochastic gradient descent, given a particular tree structure; unlike with the above autoencoding forest, stacked RBM Woods can be fine-tuned globally just like Deep Belief Networks, giving rise to Stochastic Deep Woods. Note that when a tree is a chain we obtain a form of shunting inhibition as the resulting "activation function" for the deepest leaf.

References

[1] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 153–160. MIT Press, 2007.

[2] Yoshua Bengio. Learning deep architectures for AI. Technical Report 1312, Université de Montréal, dept. IRO, 2007.

[3] Yoshua Bengio, Olivier Delalleau, and Clarence Simard. Decision trees do not generalize to new variations. Technical Report 1304, Universite de Montreal, Dept. IRO, 2007.

[4] Yoshua Bengio and Yann Le Cun. Scaling learning algorithms towards AI. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large Scale Kernel Machines*. MIT Press, 2007.

[5] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[6] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of Thirteenth International Conference*, pages 148–156, USA, 1996. ACM.

[7] G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

[8] G.E. Hinton. Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 1–12, Amherst 1986, 1986. Lawrence Erlbaum, Hillsdale.

[9] G.E. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40:185–234, 1989.

[10] G.E. Hinton and R.R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[11] Tin Kam Ho. Random decision forest. In *3rd International Conference on Document Analysis and Recognition*, pages 278–282, Montreal, Canada, 1995.

[12] Honglak Lee, Chaitanya Ekanadham, and Andrew Ng. Sparse deep belief net model for visual area V2. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA, 2008.

[13] J.L. McClelland, D.E. Rumelhart, and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 2. MIT Press, Cambridge, 1986.

[14] Marc'Aurelio Ranzato, Christopher Poultney, Sumit Chopra, and Yann LeCun. Efficient learning of sparse representations with an energy-based model. In J. Platt et al., editor, *Advances in Neural Information Processing Systems (NIPS 2006)*. MIT Press, 2007.

[15] Joseph Turian. *Constituent Parsing by Classification*. PhD thesis, New York University, 2007.

[16] P.E. Utgoff and D.J. Stracuzzi. Many-layered learning. *Neural Computation*, 14:2497–2539, 2002.