Deep Belief Networks

Ruslan Salakhutdinov and Geoff Hinton
Department of Computer Science, University of Toronto
6 King's College Rd, M5S 3G4, Canada
rsalakhu,hinton@cs.toronto.edu

Topic: learning algorithms
Preference: oral. Presenter: Ruslan Salakhutdinov

1 Abstract

Recently, Hinton et al.[6] derived a way to perform fast, greedy learning of deep belief networks (DBN) one layer at a time, with the top two layers forming an undirected bipartite graph (associate memory).

The learning procedure consists of training a stack of Restricted Boltzmann Machines (RBM's) each having only one layer of latent (hidden) feature detectors. The learned feature activations of one RBM are used as the "data" for training the next RBM in the stack.

The important aspect of this layer-wise training procedure is that, provided the number of features per layer does not decrease, [6] showed that each extra layer increases a variational lower bound on the log probability of data. So layer-by-layer training can be repeated several times[1] to learn a deep, hierarchical model in which each layer of features captures strong high-order correlations between the activities of features in the layer below. We will discuss three ideas based on greedily learning a hierarchy of features:

Nonlinear Dimensionality Reduction. The DBN framework allows us to make nonlinear autoencoders work considerably better [7] than widely used methods such as PCA, SVD, and LLE.

The standard way to train autoencoders is to use backpropagation to reduce the reconstruction error. It is difficult, however, to optimize the weights in non-linear autoencoders that have multiple hidden layers with many million parameters [3, 5]. We use our greedy learning algorithm to pretrain autoencoders. This *pretraining* stage discovers useful features efficiently.

After the pretraining stage, the model is "unfolded" to produce encoder and decoder networks that initially use the same weights. The global fine-tuning stage then uses backpropagation through the whole autoencoder to fine-tune the weights for optimal reconstruction. The key idea is that the greedy learning algorithm will perform a global search for a good, sensible region in the parameter space. Therefore, with this pretraining, we will already have a good data reconstruction model. Backpropagation is better at local fine-tuning of the model parameters than global search. So further training of the entire autoencoder using backpropagation will result in a good local optimum.

Learning Semantic Address Space (SAS) for Fast Document Retrieval. Most of the existing text retrieval algorithms in one way or another rely on comparing a given query document to all other documents from the large document collection, retrieving the most relevant ones. Typically, the larger the word vocabulary and the size of data collection, the longer it takes to retrieve relevant documents.

The DBN framework allows us to build a model that can learn to map documents into "semantic" binary codes. By using learned binary codes as memory addresses, we can learn *Semantic Address Space*, so a document can be mapped to a memory address in such a way that a small hamming-ball around that memory address contains semantically similar documents. This representation allows to retrieve a short-list of semantically similar documents on very large document sets in time independent of the number of documents. The short-list can then be given to a slower but more precise retrieval method, such as TF-IDF.

Using a simple implementation of a 20-bit Semantic Address Space in C, we performed a set of experiments on Reuters RCV2 dataset, containing 804,414 newswire stories, split randomly into 402,212 training and 402,212 test documents. For a given query, it takes about 0.5 milliseconds to create a short-list of about 3,000 semantically similar documents (hamming-ball of radius 4), and 0.01 seconds to retrieve the top few matches

[1] In fact, one can proceed learning recursively for as many layers as desired

from that short-list using TF-IDF. Locality-Sensitive Hashing (LSH) [1] takes about 0.5 seconds to perform the same search using E^2LSH 0.1 software, provided by Alexandr Andoni and Piotr Indyk. SAS also achieves higher accuracy compared to LSH and TF-IDF alone.

Learning Nonlinear Embeddings. The DBN framework can also be used to efficiently learn a nonlinear transformation from the input space to a low-dimensional feature space in which K-nearest neighbour classification performs well [8]. This can be viewed as a nonlinear extension of NCA [4].

By learning such a transformation, using many layers of nonlinearities, we achieve an error of 1.00% on a widely used version of the MNIST handwritten digit recognition task. This is compared to the best reported error rates (without using any domain-specific knowledge) of 1.6% for randomly initialized backpropagation and 1.4% for Support Vector Machines [2]. Linear methods such as linear NCA, LDA or PCA are much worse than nonlinear NCA (see Fig. 1).

References

[1] Datar, Immorlica, Indyk, and Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *COMPGEOM: Annual ACM Symposium on Computational Geometry*, 2004.

[2] D. Decoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 46(1/3):161, 2002.

[3] D. DeMers and G.W. Cottrell. Nonlinear dimensionality reduction. In Cowan Hanson and Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 580–587, San Mateo, CA, 1993. Morgan Kaufmann.

[4] J. Goldberger, S. T. Roweis, G. E. Hinton, and R. Salakhutdinov. Neighborhood components analysis. In *NIPS*, 2004.

[5] R. Hecht-Nielsen. Replicator neural networks for universal optimal source coding. *Science*, 269:1860–1863, 1995.

[6] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18, 2006.

[7] G. E. Hinton and R. R. Salakhutdinov. Non-linear dimensionality reduction using neural networks. *To appear in Science*, 2006.

[8] Ruslan Salakhutdinov and Geoffrey E. Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *AI and Statistics*, 2007.

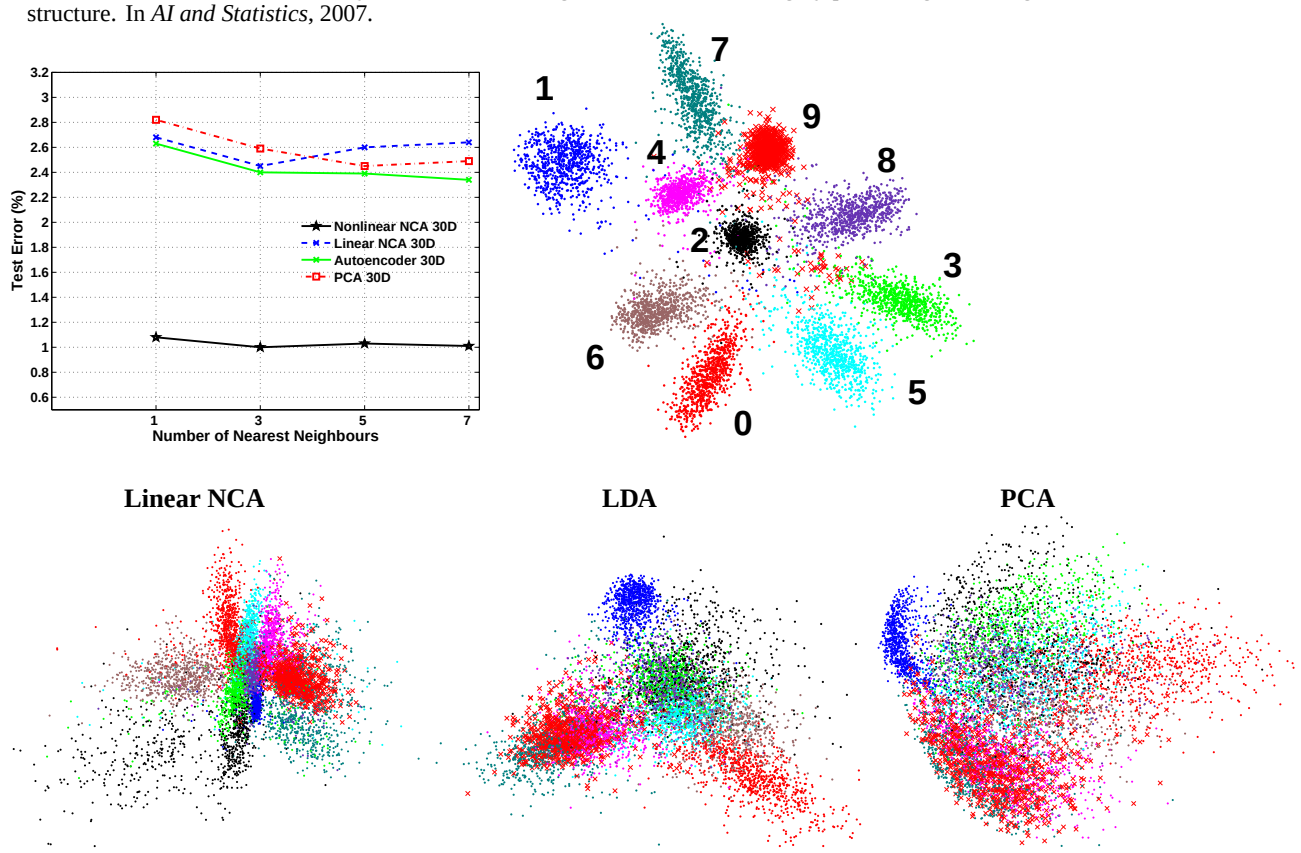

Figure 1: The top left panel shows KNN results on the MNIST test set. The top right panel shows the 2-dimensional codes produced by nonlinear NCA on the test data using a 784-500-500-2000-2 encoder. The bottom panels show the 2-dimensional codes produced by linear NCA, Linear Discriminant Analysis, and PCA.