Reducing the Dimensionality of Data with Neural Networks

Hinton, Geoffrey E.,
and Ruslan R. Salakhutdinov.

Overview

Dimensionality reduction
- classification, visualization, communication, storage of high-dimensional data

Goal : use a deep neural to reduce the dimensionality of an input.
-> the reduction techniques described are compared to principal component analysis (PCA) and logistic PCA

PCA
: Find the directions of greatest variance in the data set and
Represent each data point by its coordinates along each of these directions.

Overview

Autoencoder

- Nonlinear generalization of PCA
- Encoder transform the high-dimensional data into a low-dimensional code
- Decoder recover the data from the code

- Starts with random weights in the two networks
- Trained by minimizing the discrepancy between the original data and its reconstruction.
- Gradients are obtained by the chain rule to back-propagate error from the decoder network to encoder network.

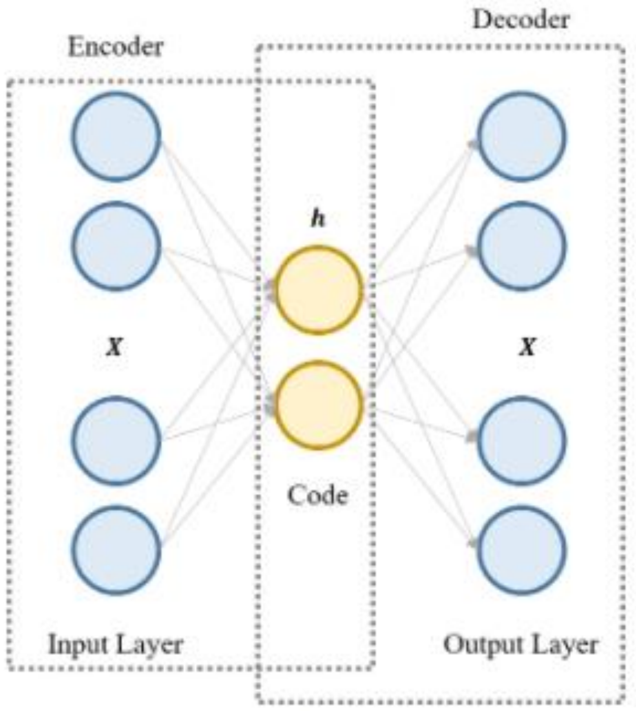


Figure 1: Autoencoder

Overview

It is difficult to optimize multilayer autoencoder

1. With large initial weights
: autoencoders typically find poor local minima

2. With small initial weights
: the gradients in the early layers are tiny, making it infeasible to train autoencoders with many hidden layers

3. If initial weights are close to a good solution
: gradient descent works well
-> But finding such initial weights is very difficult

Pretraining procedure for binary data, generalize it to real-valued data, and show that it works well for a variety of data sets.
-> based on **Restricted Boltzmann Machine (RBM)**

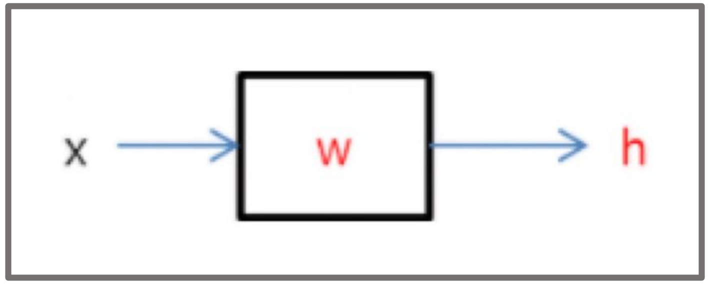
Methods

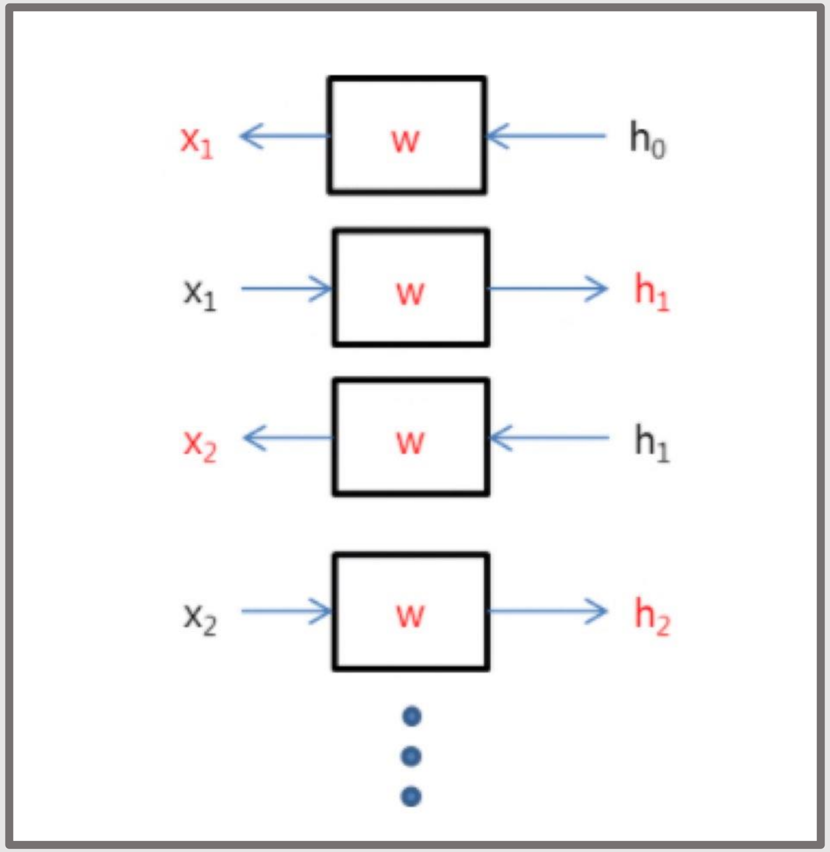
Restricted Boltzmann Machine (RBM)

The input data correspond to **visible** units of the RBM and the feature detectors correspond to hidden units.

A joint configuration (w, h) of the visible and hidden units has an **energy**
The network assigns a probability to every possible data via this energy function

$$E\{h_0 x_0 - h_\infty x_\infty\}$$



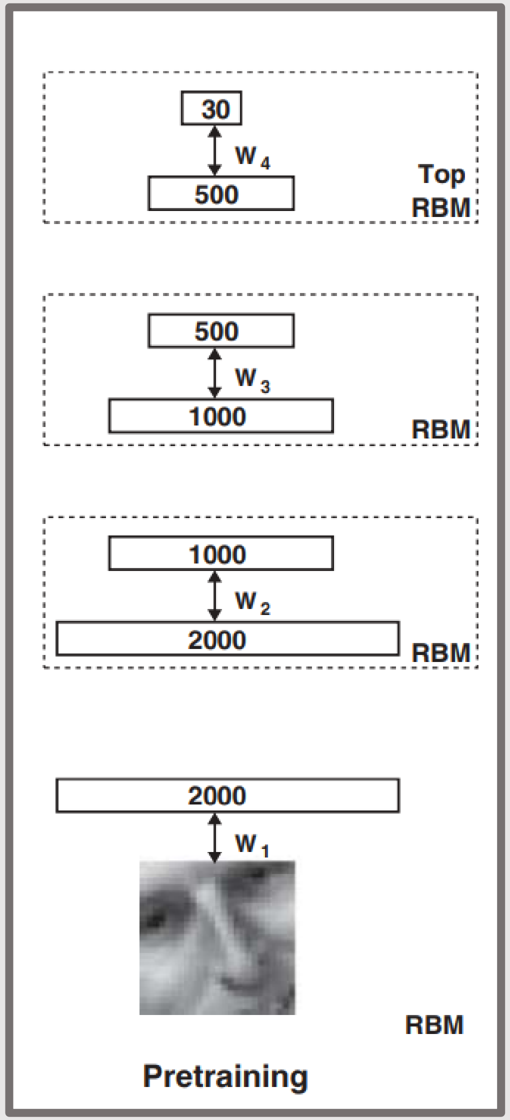


Methods

Training a large neural network through greedy layer-wise pretraining

He took an input (50*50 images), and trained a RBM to reduce the feature space of the image.

Each set of weights are learned individually, so in the first step, W1 is learned and once W1 is learned,
the data is then all restored as the result of the first transfer and W2 is learned, and so on.

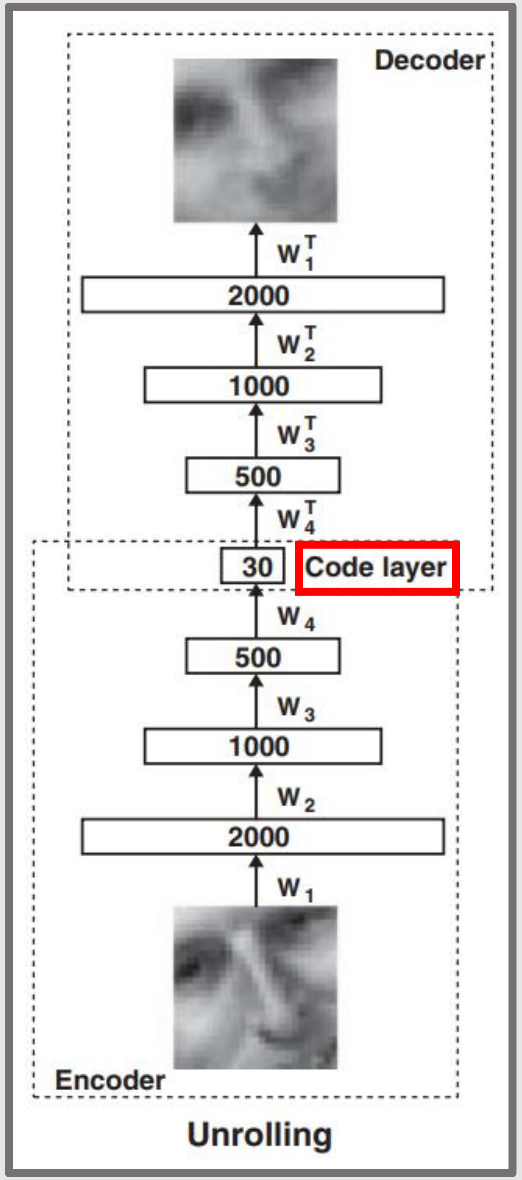


Methods

RBM is generative models, so the weights are bidirectional.
-> activation in layer two can be used to generate a corresponding set of activations in layer one.

The weights between the layers are effectively re-used to map the data back to its original dimensionality.
-> the input weights for a NN that can be trained with backpropagation to fine-tune the weights using the input weights as the outputs to the network

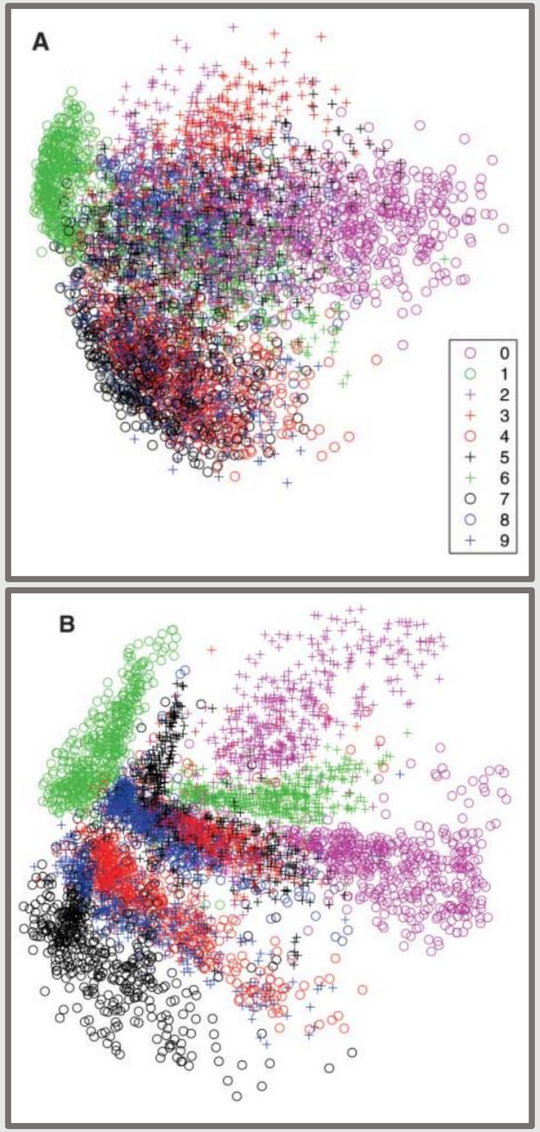
Code layer : used for classification, regression, clustering, etc.



Results

MNIST

A : 2 dimensional results of standard PCA
B : results of a deep neural network
-> autoencoder produces a much more distinct separation of digits when projected into 2 dimensional space than PCA.
-> because the autoencoder has the ability to learn non-linear features since each layers has non-linear activation units and thus can learn a more complex representation of the input.



Analysis

Initialize weights using RBM & gradient descent
-> learning ↑
-> deeplearning ↑

Deep Belief Network (DBN)
: Restricted Boltzmann machine with connection only between layer and layer
-> learning ↑

General NN : Learn weights from upper to lower layers
DBN : Learn weights from lower to upper layers