Reducing the Dimensionality of Data with Neural Networks

presentation based on a research paper
by G. E. Hinton and R. R. Salakhutdinov

Chris Sipola, s1667278[1] Elias Mistler, s1675946[1]

[1]School of Informatics
University of Edinburgh

Data Mining and Exploration, 2017

Outline

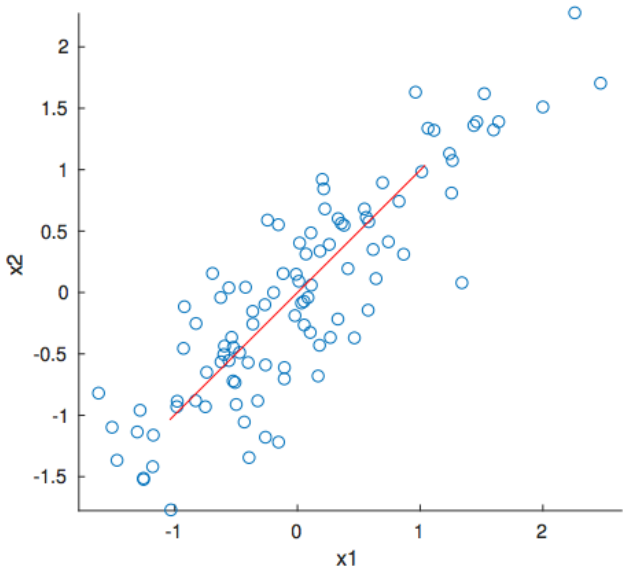
Paper overview

- Neural nets can be used for dimensionality reduction. They're called autoencoder networks
- However, these networks only work well if weights are initialized to be close to correct solution
- This paper describes a way of initializing the weights through pre-training using restricted Boltzmann machines (RBMs)
- It works much better than PCA for dimensionality reduction

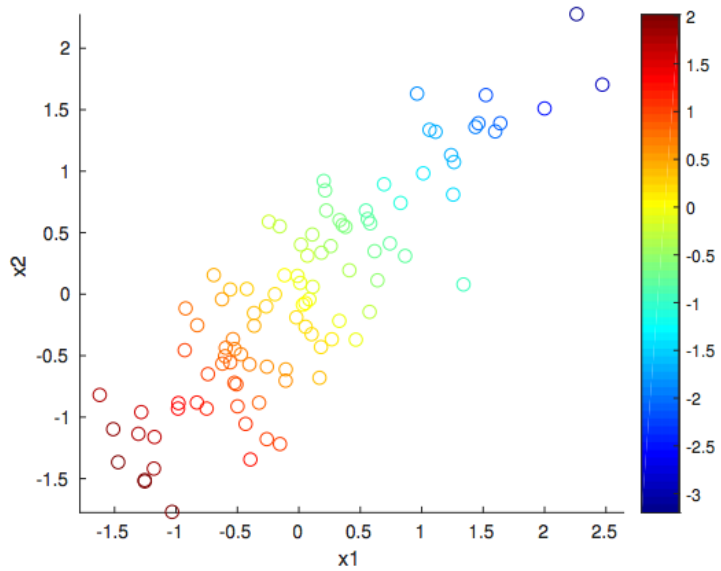
Outline

PCA: linear data[1]

PCA is good at summarizing data where relationships are linear.



(a) Principle component direction

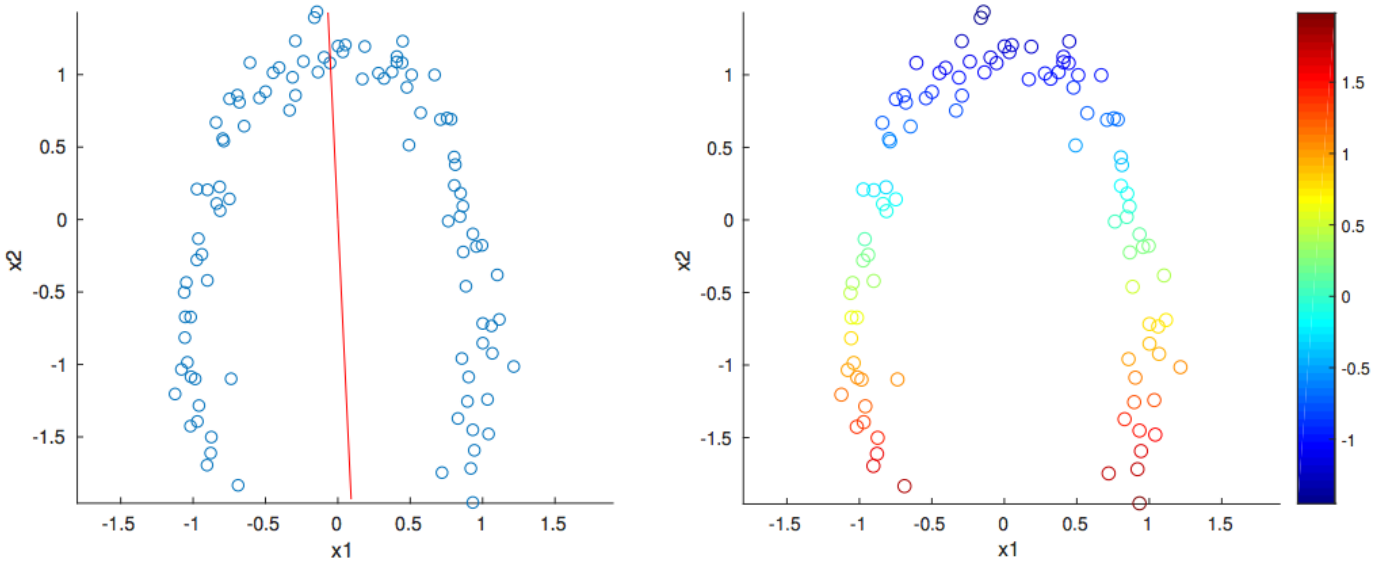


(b) Principle component scores

[1] Image source: http://www.inf.ed.ac.uk/teaching/courses/dme/2017/lecture-notes.pdf

PCA: *non*linear data[2]

However, traditional PCA has trouble summarizing nonlinear data because it can't account for nonlinear relationships.



(a) Principle component direction

(b) Principle component scores

[2]Image source: http://www.inf.ed.ac.uk/teaching/courses/dme/2017/lecture-notes.pdf

Can engineer features for PCA, e.g.,

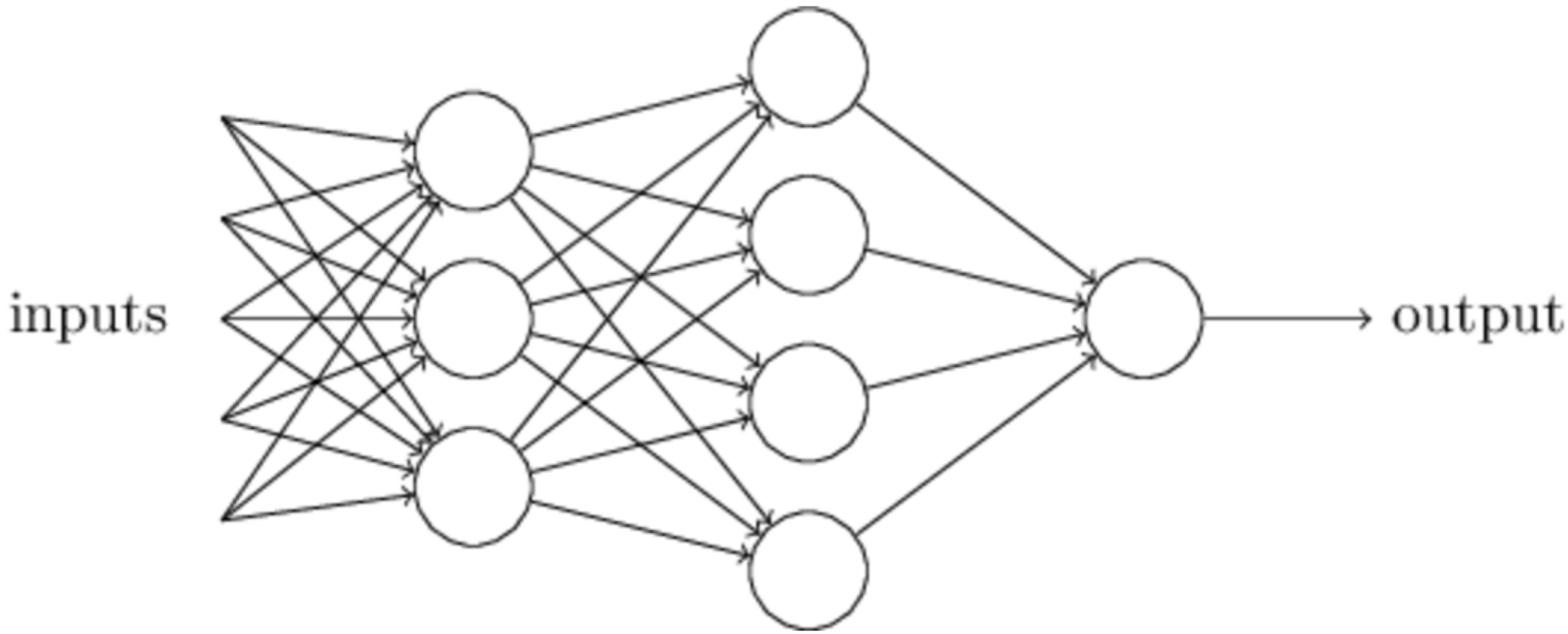
$$\phi(x_1, x_2) = (x_1, x_2, \sqrt{x_1 + x_2}, atan(x_1, x_2))$$

but ideally we would *learn* them using powerful models like neural networks.

Outline

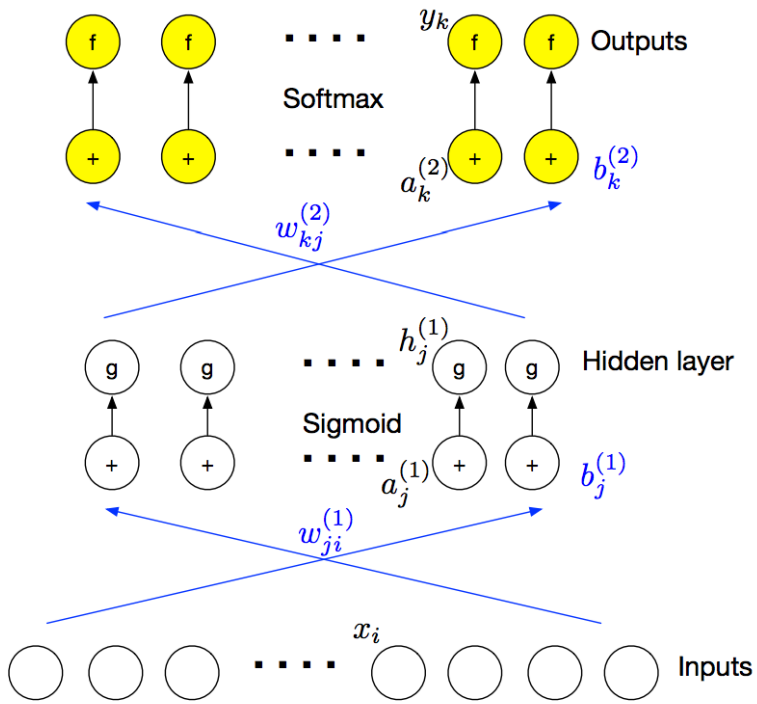
Modeling nonlinear, complex relationships[3]

Activation functions in neural networks—such as the *sigmoid* or *ReLU* functions—allow us to capture complex relationships between input variables.



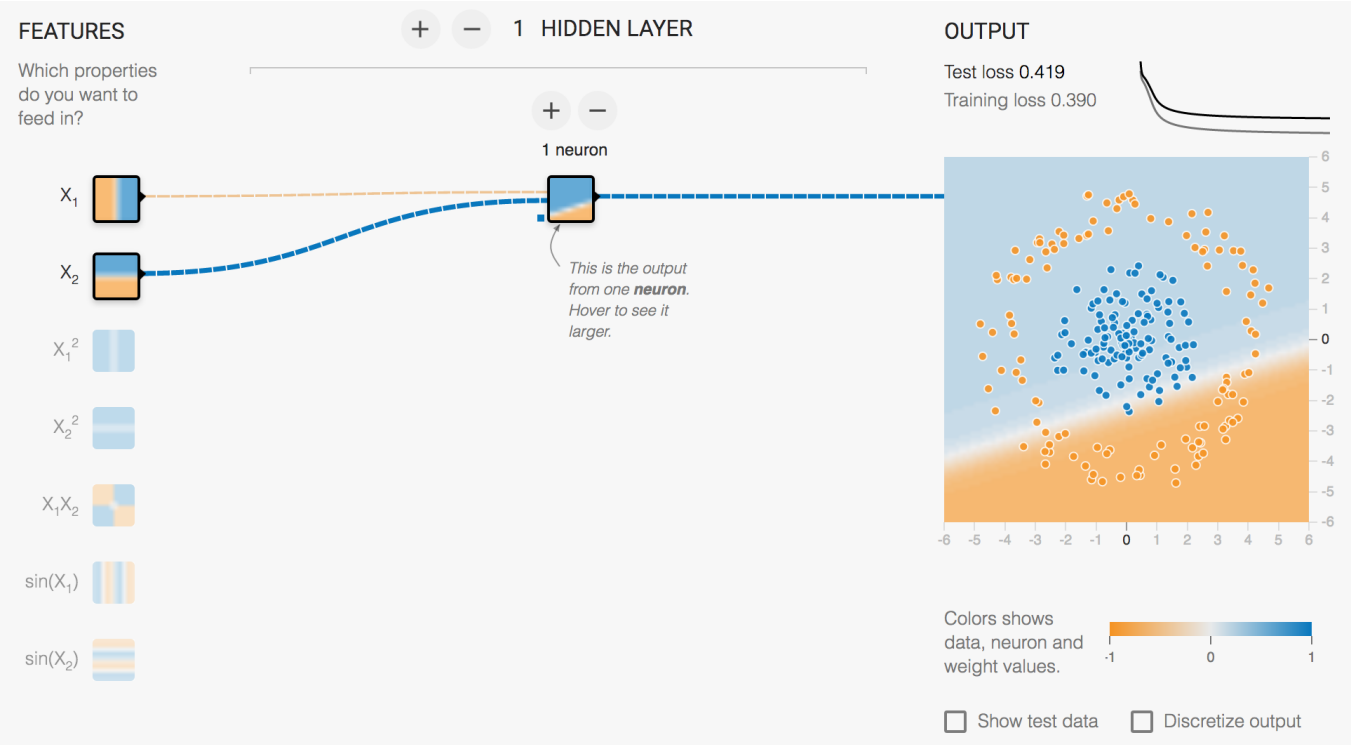
[3] Image source: http://neuralnetworksanddeeplearning.com/chap1.html

Activation function after linear transformation[4]



$$y_k = \text{softmax}\left(\sum_{r=1}^{H} w_{kr}^{(2)} h_r^{(1)} + b_k\right) \qquad h_j^{(1)} = \text{sigmoid}\left(\sum_{s=1}^{d} w_{js}^{(1)} x_s + b_j\right)$$

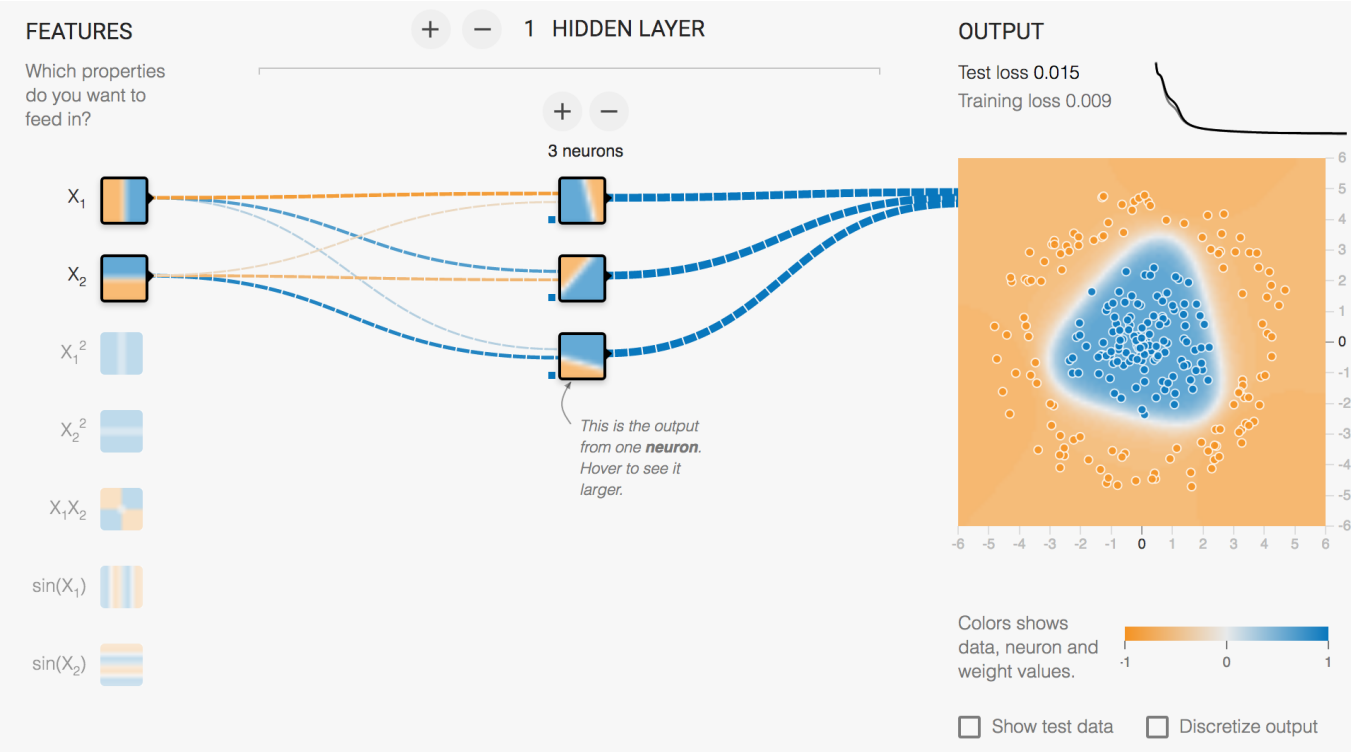
Linear function (no effective hidden layers)[5]



- Data has a nonlinear relationship
- No hidden layers → logistic regression

[5]Source: http://playground.tensorflow.org

Nonlinear function (one hidden layer)[6]



One hidden layer $\rightarrow$ smarter decision boundary.

[6] Source: http://playground.tensorflow.org

Deep networks: training is hard[7]

Why is training deep networks hard?

- Vanishing/exploding gradients: gradients for layers closer to the input layer are computed multiplicatively using backprop
- If sigmoid/tanh hidden units near the output saturate then back-propagated gradients will be very small

[7] Slide material taken from MLP course: http://www.inf.ed.ac.uk/teaching/courses/mlp/2016/mlp06-enc.pdf

Solution: pre-training[8]

Solve by stacked pre-training

- Train the first hidden layer
- Add a new hidden layer, and train only the parameters relating to the new hidden layer. Repeat.
- Then use the pretrained weights to initialise the network and fine-tune the complete network using gradient descent

[8] Slide material taken from MLP course: http://www.inf.ed.ac.uk/teaching/courses/mlp/2016/mlp06-enc.pdf

Solution: pre-training[8]

Solve by stacked pre-training

- Train the first hidden layer
- Add a new hidden layer, and train only the parameters relating to the new hidden layer. Repeat.
- Then use the pretrained weights to initialise the network and fine-tune the complete network using gradient descent

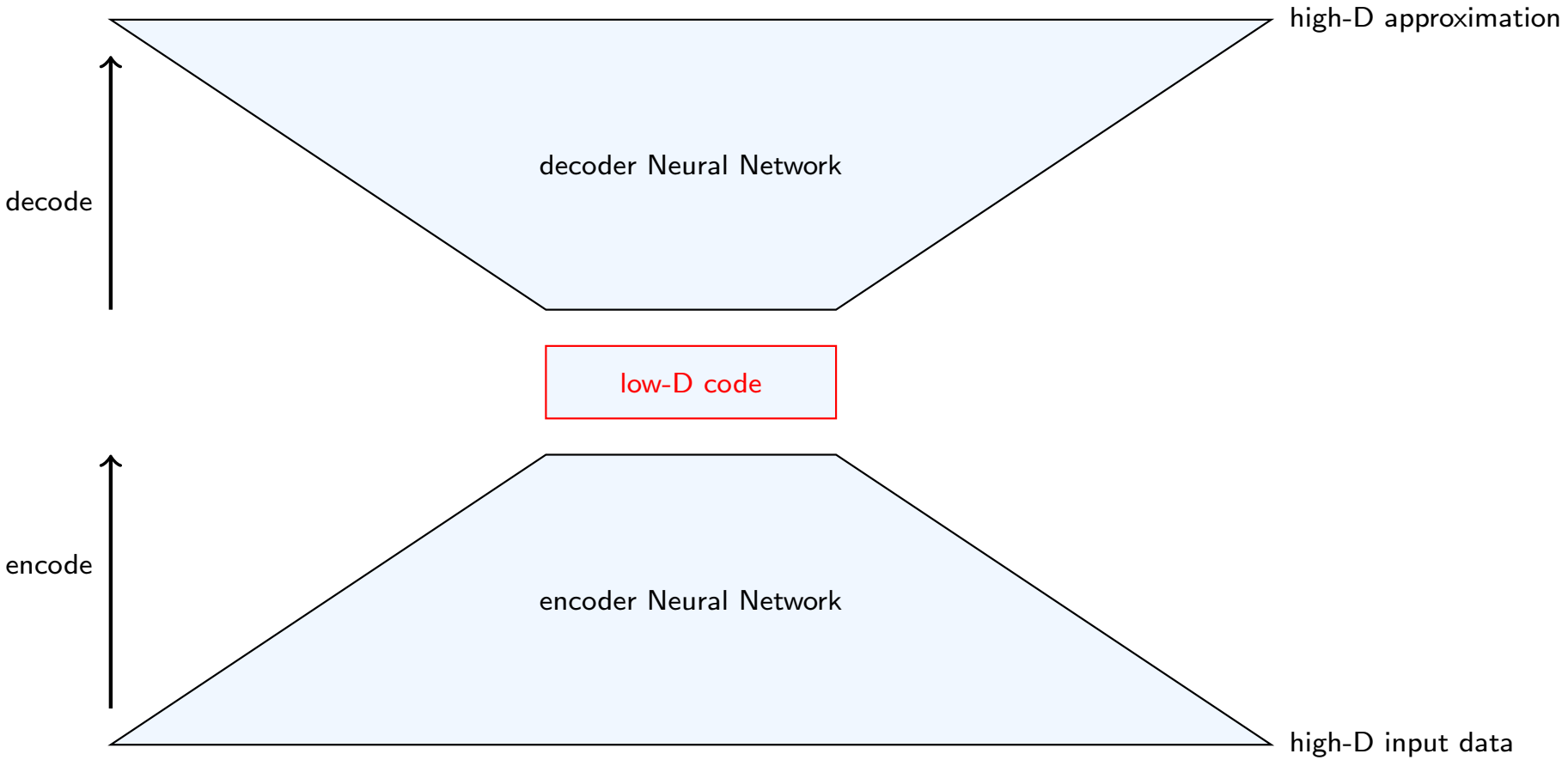
Approaches to pre-training

- Supervised: Layer-by-layer cross-entropy training
- Unsupervised: Restricted Boltzmann machines

[8] Slide material taken from MLP course: http://www.inf.ed.ac.uk/teaching/courses/mlp/2016/mlp06-enc.pdf

Outline

Autoencoder architecture



Autoencoder architecture

- Artificial Neural Network architecture:
  One network with a low-D hidden layer
  Can be split to 2 NNs $\rightarrow$ encoder and decoder
- Unsupervised learning
- learns low-dimensional (higher-level) representation ("code") of the data
- Claim: This code performs much better than PCA for dimensionality reduction if trained properly

Autoencoders: Training

- Unsupervised learning is achieved through minimising the difference between input and output
- Deep architecture $\rightarrow$ hard to train
- Solution:
  - layer-wise pre-training with RBMs
  - gradient descent for fine-tuning

Outline

Restricted Boltzmann Machines (RBM)

- Two-layer Neural Network architecture
- binary nodes
- efficient training through stochasticity
  - logistic squashing / softmax function
  - interpreted as activation probability
  - $\rightarrow$ stochastic binary values
  - $\rightarrow$ training maximises probability of training data

RBM probabilities and energy function

- v: vector of m visible units v_i
- h: vector of n hidden units h_j
- W: $m \times n$ matrix of weights $w_{i,j}$
- a (size m), b (size n): bias vectors for visible / hidden units

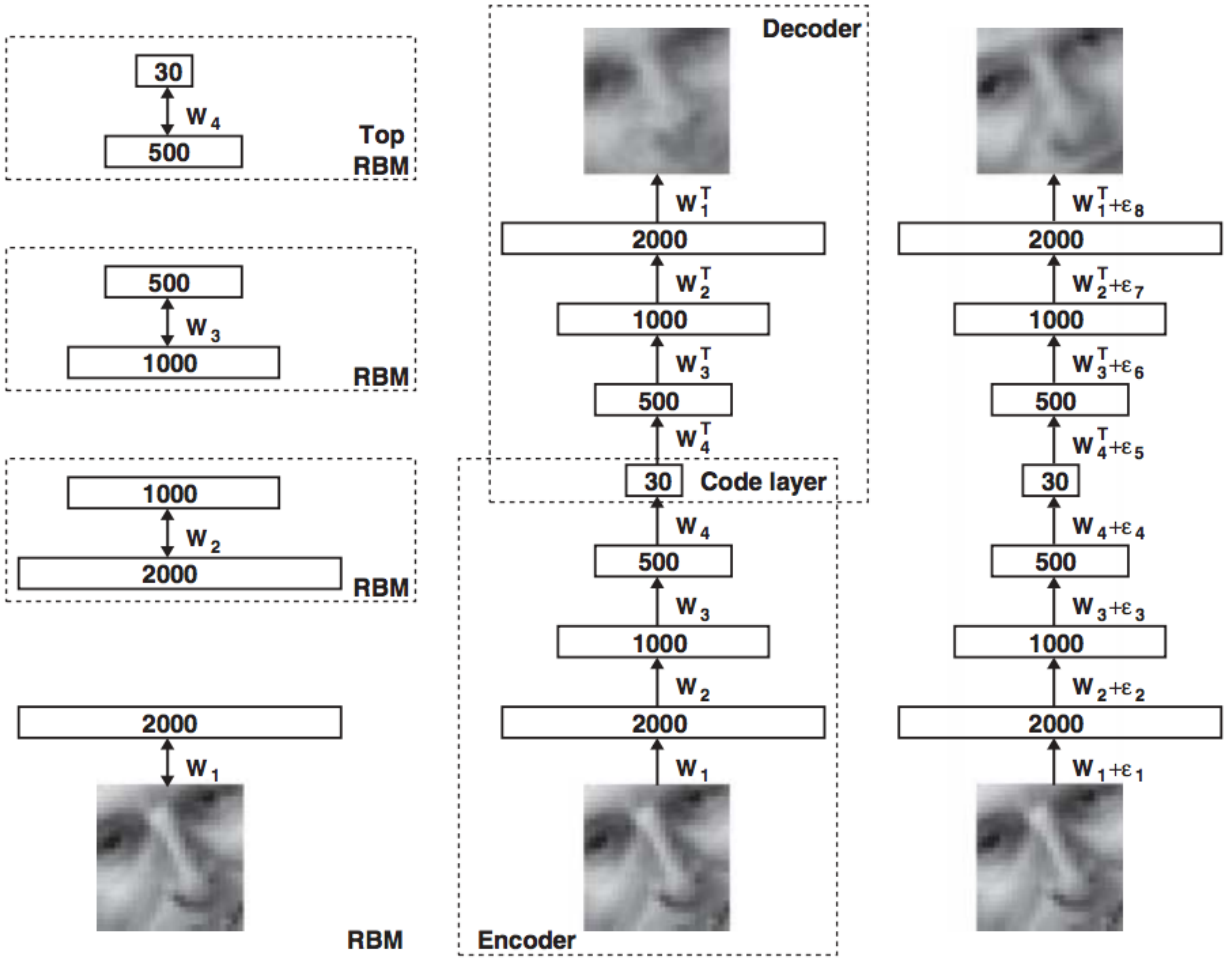
RBM probabilities and energy function

- v: vector of m visible units v_i
- h: vector of n hidden units h_j
- W: $m \times n$ matrix of weights $w_{i,j}$
- a (size m), b (size n): bias vectors for visible / hidden units
- Activation probability: $P(h_j = 1|v) = \sigma(b_j + \sum_{i=1}^{m} w_{i,j} v_i)$

RBM probabilities and energy function

- v: vector of m visible units v_i
- h: vector of n hidden units h_j
- W: $m \times n$ matrix of weights $w_{i,j}$
- a (size m), b (size n): bias vectors for visible / hidden units
- Activation probability: $P(h_j = 1|v) = \sigma(b_j + \sum_{i=1}^{m} w_{i,j} v_i)$
- Energy function: $E(v, h) = -a^T v - b^T h - v^T W h$

Architecture and procedure[9]



Contribution of the paper

- Code layer contains the dimensionality-reduced data
- Note the encoder network and decoder network
- Once reduced, need decoder network to recover data

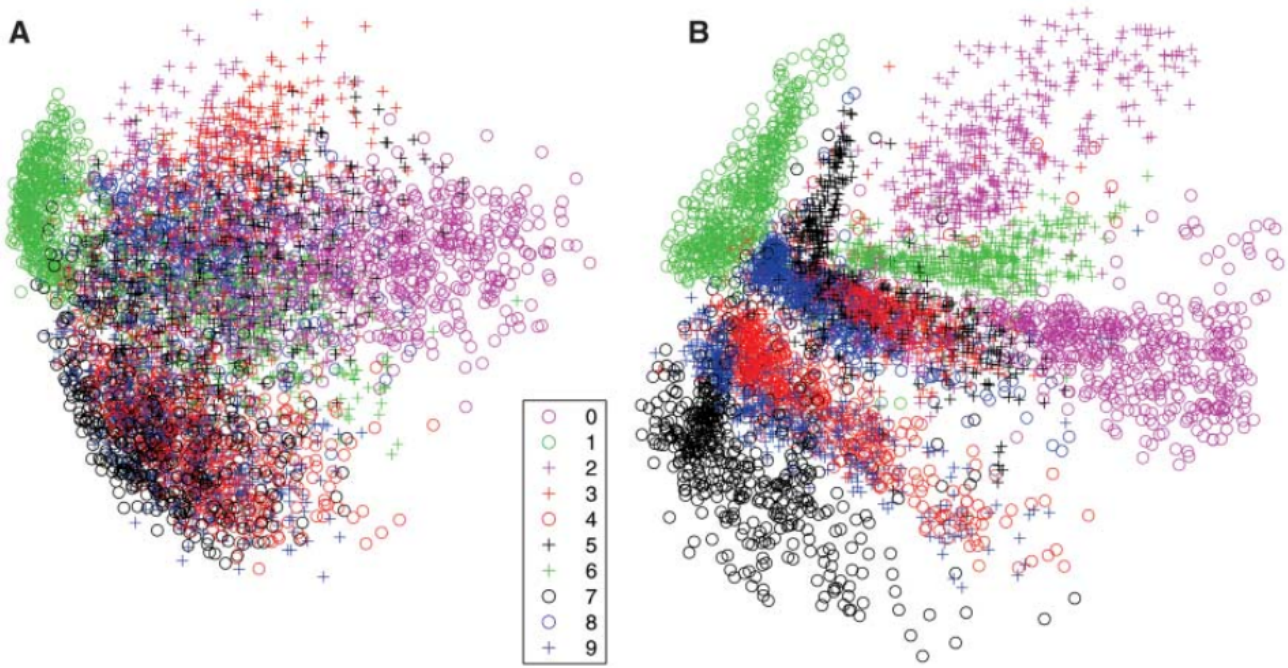
- Deep Neural Network architecture
  $\rightarrow$ need for effective training

$\Rightarrow$ The paper suggests a new approach to autoencoder training by doing layerwise training with restricted Boltzmann machines

Outline

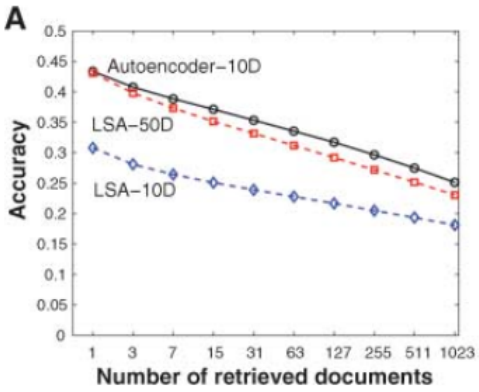
Example 1: image compression

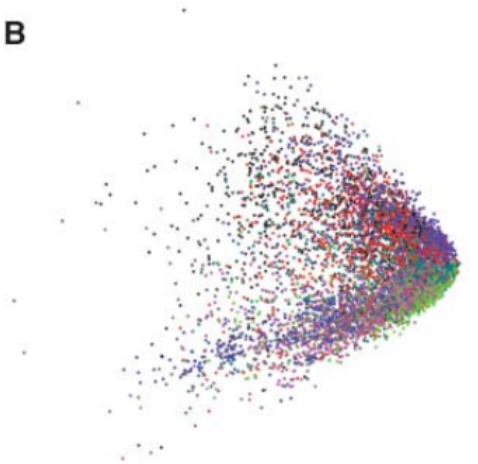
Fig. 3. (**A**) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (**B**) The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (*8*).

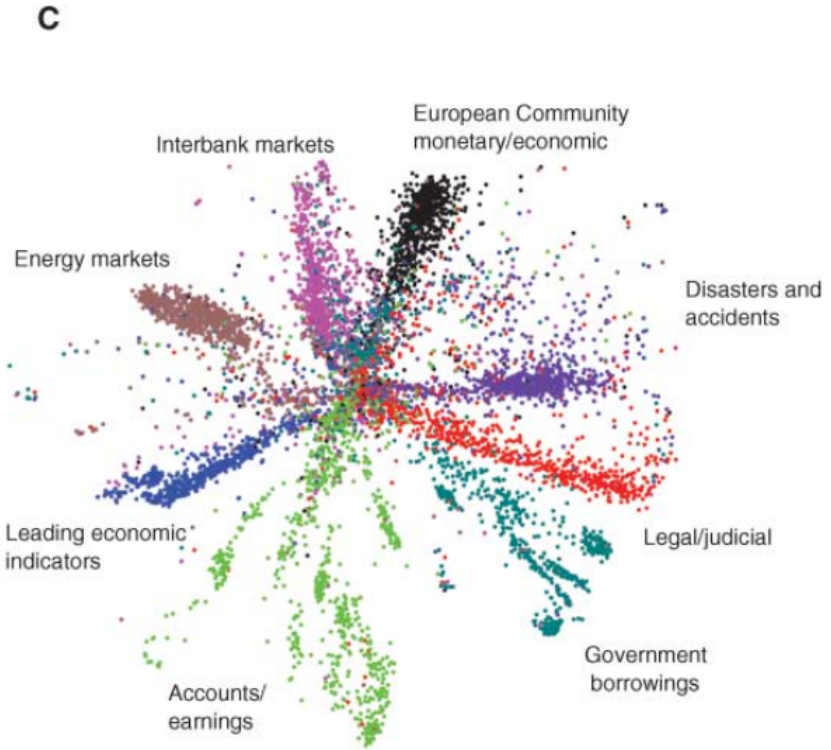


Example 2: document retrieval

Fig. 4. (**A**) The fraction of retrieved documents in the same class as the query when a query document from the test set is used to retrieve other test set documents, averaged over all 402,207 possible queries. (**B**) The codes produced by two-dimensional LSA. (**C**) The codes produced by a 2000-500-250-125-2 autoencoder.



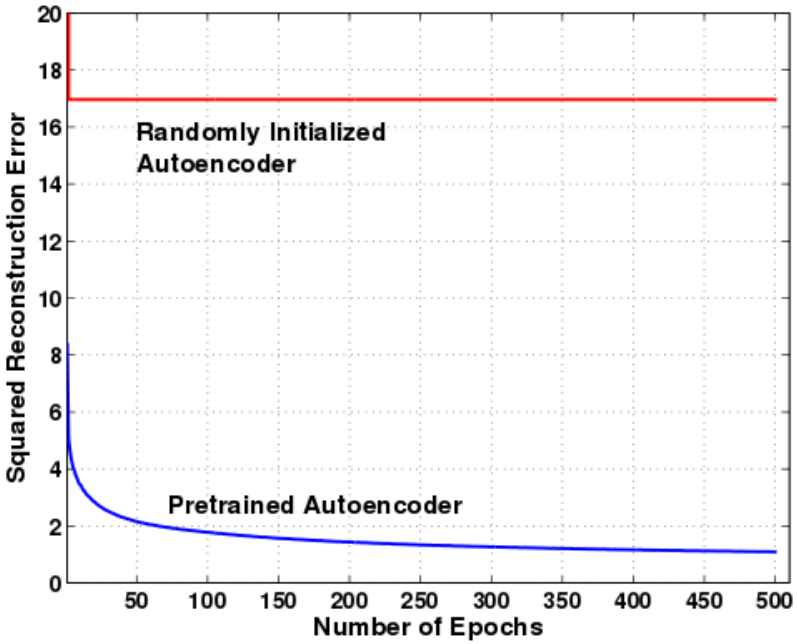


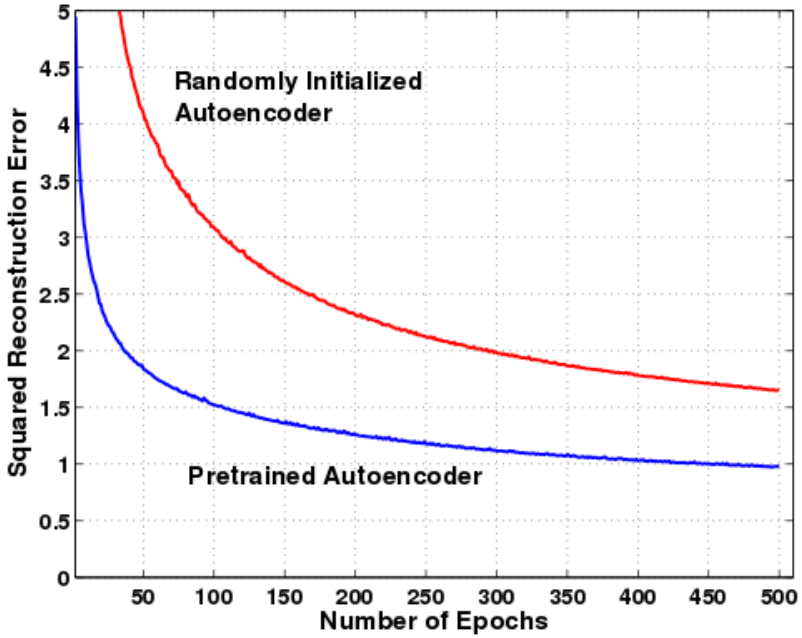


Outline

Effect of the RBM pre-training

Random initialisation vs. layer-wise pre-training in a deep (left) and shallow (right) autoencoder[10]





[10] Image source: Online supplements of the paper

Outline

Why it matters

- Dimensionality reduction is necessary to:
  - handle, visualise and communicate high-D data effectively
  - store data space-efficiently while preserving the inner structure
  - run further Machine Learning algorithms more efficiently

Why it matters

- Dimensionality reduction is necessary to:
  - handle, visualise and communicate high-D data effectively
  - store data space-efficiently while preserving the inner structure
  - run further Machine Learning algorithms more efficiently
- Autoencoders offer dimensionality reduction superior to PCA made feasible by RBM-based, layer-wise pre-training:
  - learns intrinsic high-level features automatically
  - is extremely flexible due to non-linearity
  - can (theoretically) learn arbitrary mappings / structures

Outline

Conclusion

- Autoencoders can reduce dimensionality by better representing nonlinear relationships in the data
- Restricted Boltzmann Machines allow for effective, layer-wise pre-training
- The resulting non-linear representation performs far better than PCA for complex structures in the data