

Boltzmann machines

How the Boltzmann Machine was born:

John Hopfield gave a talk, and I was sitting next to Geoff Hinton, and Hopfield was saying, “We could do optimisation here”, . . . and that was something we were interested in. I had just read an article by Scott Kirkpatrick on simulated annealing, and I said, “Geoff, we could heat it up, we could heat up the Hopfield network by adding a temperature”, so instead of always going downhill sometimes you pop up and with that, if you slowly cool it we can find the optimal solution. . . . It turns out something magical happens when you heat up a Hopfield network: it becomes capable of learning the weights to a multi-layer network; which is the first time that had been done.
T Sejnowski, 2023.

From: James Stone, *The Artificial Intelligence Papers: Original Research Papers With Tutorial Commentaries.*

<https://jamesstone.sites.sheffield.ac.uk/books>

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition
Washington, D. C., June, 1983

OPTIMAL PERCEPTUAL INFERENCE

Geoffrey E. Hinton

Computer Science Department
Carnegie-Mellon University

Terrence J. Sejnowski

Biophysics Department
The Johns Hopkins University



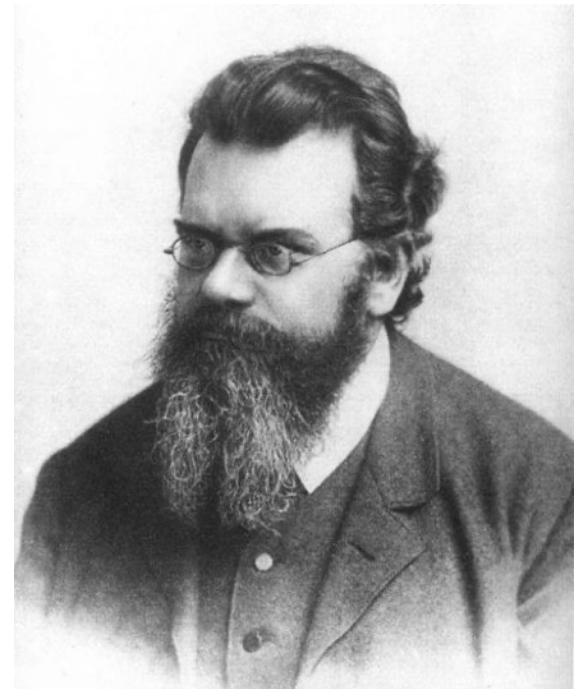
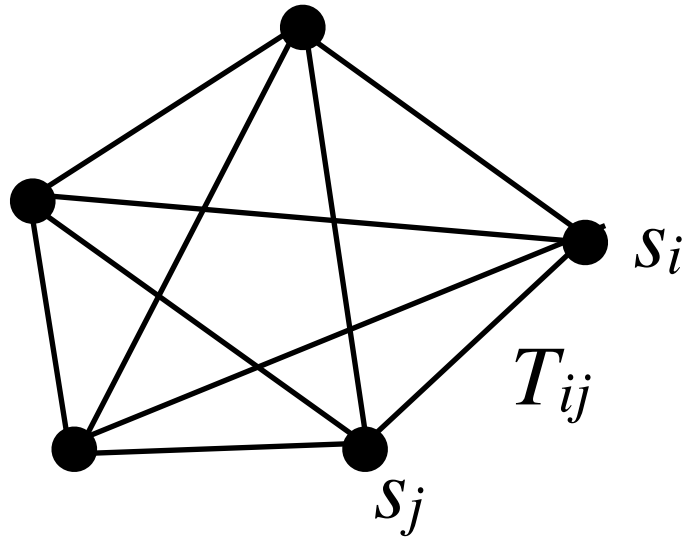
Terry Sejnowski

Geoff Hinton

Introduction

One way of interpreting images is to formulate hypotheses about parts or aspects of the image and then decide which of these hypotheses are likely to be correct. The probability that each hypothesis is correct is determined partly by its fit to the image and partly by its fit to other hypotheses that are taken to be correct, so the truth value of an individual hypothesis cannot be decided in isolation. One method of searching for the most plausible combination of hypotheses is to use a relaxation process in which a probability is associated with each hypothesis, and the probabilities are then iteratively modified on the basis of the fit to the image and the known relationships between hypotheses. An attractive property of relaxation methods is that they can be implemented in parallel hardware where one computational unit is used for each possible hypothesis, and the interactions between hypotheses are implemented by direct hardware connections between the units.

The “Boltzmann machine” (Hinton & Sejnowski, 1983)



Ludwig Boltzmann

$$E(\mathbf{s}) = -\frac{1}{2} \sum_{ij} T_{ij} s_i s_j$$

$$P(\mathbf{s}) = \frac{1}{Z} e^{-\beta E(\mathbf{s})} \quad \beta = \frac{1}{\text{temperature}}$$

Boltzmann-Gibbs distribution

$$P(\mathbf{x}) = \frac{1}{Z} e^{\lambda \phi(\mathbf{x})}$$

$$Z = \int e^{\lambda \phi(\mathbf{x})} d\mathbf{x}$$

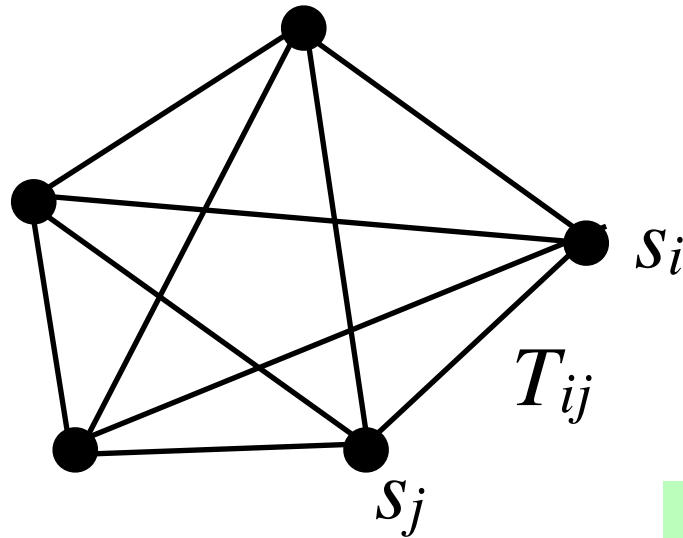
Learning rule:

$$\begin{aligned} \Delta\lambda &\propto \frac{\partial}{\partial\lambda} \langle \log P(\mathbf{x}) \rangle \\ &= \langle \phi(\mathbf{x}) \rangle - \langle \phi(\mathbf{x}) \rangle_{P(\mathbf{x})} \end{aligned}$$

$$\log P(\mathbf{x}) = \lambda \phi(\mathbf{x}) - \log Z$$

$$\begin{aligned} \frac{\partial}{\partial \lambda} \langle \log P(\mathbf{x}) \rangle &= \frac{\partial}{\partial \lambda} \langle \lambda \phi(\mathbf{x}) - \log Z \rangle \\ &= \langle \phi(\mathbf{x}) - \frac{\partial}{\partial \lambda} \log Z \rangle \\ &= \langle \phi(\mathbf{x}) - \frac{1}{Z} \frac{\partial Z}{\partial \lambda} \rangle \\ &= \langle \phi(\mathbf{x}) - \frac{1}{Z} \int \phi(\mathbf{x}) e^{\lambda \phi(\mathbf{x})} d\mathbf{x} \rangle \\ &= \langle \phi(\mathbf{x}) - \int \phi(\mathbf{x}) P(\mathbf{x}) d\mathbf{x} \rangle \\ &= \langle \phi(\mathbf{x}) \rangle - \langle \phi(\mathbf{x}) \rangle_{P(\mathbf{x})} \end{aligned}$$

The “Boltzmann machine” (Hinton & Sejnowski, 1983)



$$E(\mathbf{s}) = -\frac{1}{2} \sum_{ij} T_{ij} s_i s_j$$


λ_{ij} $\phi_{ij}(\mathbf{s})$

$$P(\mathbf{s}) = \frac{1}{Z} e^{-\beta E(\mathbf{s})}$$

The diagram shows a green box containing λ_{ij} and a red box containing $\phi_{ij}(\mathbf{s})$. A green arrow points from the T_{ij} term in the energy equation to the λ_{ij} box. A red arrow points from the $s_i s_j$ term in the energy equation to the $\phi_{ij}(\mathbf{s})$ box.

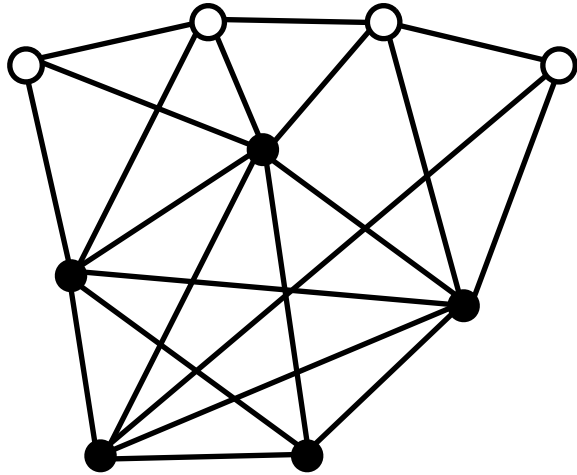
The Boltzmann machine learning rule

$$\begin{aligned}\Delta T_{ij} &\propto \frac{\partial \langle \log P(\mathbf{s}) \rangle}{\partial T_{ij}} \\ &= \beta \left[\langle s_i s_j \rangle_{\text{clamped}} - \langle s_i s_j \rangle_{\text{free}} \right]\end{aligned}$$



data $P(\mathbf{s})$

Boltzmann machine with hidden units (Hinton & Sejnowski)



‘hidden’ units, s^h

‘visible’ units, s^v

$$E(\mathbf{s}^v, \mathbf{s}^h) = - \sum_{i,j} T_{ij}^{vv} s_i^v s_j^v - \sum_{i,j} T_{ij}^{vh} s_i^v s_j^h - \sum_{i,j} T_{ij}^{hh} s_i^h s_j^h$$

$$P(\mathbf{s}^v, \mathbf{s}^h) = \frac{1}{Z} e^{-E(\mathbf{s}^v, \mathbf{s}^h)}$$

$$P(\mathbf{s}^v) = \sum_{\mathbf{s}^h} P(\mathbf{s}^v, \mathbf{s}^h)$$

The Boltzmann machine learning rule

$$\begin{aligned}\Delta T_{ij} &\propto \frac{\partial \log P(\mathbf{s})}{\partial T_{ij}} \\ &= \beta \left[\langle s_i s_j \rangle_{\text{clamped}} - \langle s_i s_j \rangle_{\text{free}} \right]\end{aligned}$$

$$\text{Clamped: } \begin{cases} \mathbf{s}^v = \mathbf{x} \\ \mathbf{s}^h \sim P(\mathbf{s}^h | \mathbf{s}^v) \end{cases}$$

$$\text{Free: } [\mathbf{s}^v, \mathbf{s}^h] \sim P(\mathbf{s}^v, \mathbf{s}^h) \equiv P(\mathbf{s})$$

Gibbs sampling

To sample from $P(\mathbf{x})$:

$$\begin{aligned} x_1 &\sim P(x_1 | x_2, \dots, x_n) \\ &\quad \downarrow \\ x_2 &\sim P(x_2 | x_1, x_3, \dots, x_n) \\ &\quad \downarrow \\ x_3 &\sim P(x_3 | x_1, x_2, x_4, \dots, x_n) \\ &\quad \cdot \\ &\quad \cdot \\ &\quad \cdot \\ x_n &\sim P(x_n | x_1, \dots, x_{n-1}) \end{aligned}$$

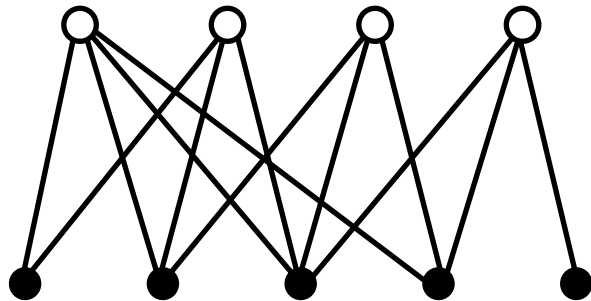
Dynamics

$$P(s_i = 1 | \{s_{\bar{i}}\}) = \frac{P(s_i = 1, \{s_{\bar{i}}\})}{P(s_i = 1, \{s_{\bar{i}}\}) + P(s_i = -1, \{s_{\bar{i}}\})}$$

Thus:

$$P(s_i = 1 | \{s_{\bar{i}}\}) = \sigma(2\beta h_i)$$
$$h_i = \sum_{j \neq i} T_{ij} s_j$$

Restricted Boltzmann machine (RBM)



'hidden' units, s^h

'visible' units, s^v

$$E(\mathbf{s}^v, \mathbf{s}^h) = - \sum_{i,j} T_{ij}^{vh} s_i^v s_j^h$$

$$P(\mathbf{s}^v, \mathbf{s}^h) = \frac{1}{Z} e^{-E(\mathbf{s}^v, \mathbf{s}^h)} \implies$$

$$P(s_i^h | s_i^h, \mathbf{s}^v) = \sigma\left(\sum_j T_{ij}^{hv} s_j^v\right)$$

$$P(s_i^v | s_i^v, \mathbf{s}^h) = \sigma\left(\sum_j T_{ij}^{vh} s_j^h\right)$$

$$P(\mathbf{s}^v) = \sum_{\mathbf{s}^h} P(\mathbf{s}^v, \mathbf{s}^h) = \frac{1}{Z} e^{\sum_i \log(1 + e^{T_i^{vh} \cdot \mathbf{s}^v})}$$

Reducing the Dimensionality of Data with Neural Networks

G. E. Hinton* and R. R. Salakhutdinov

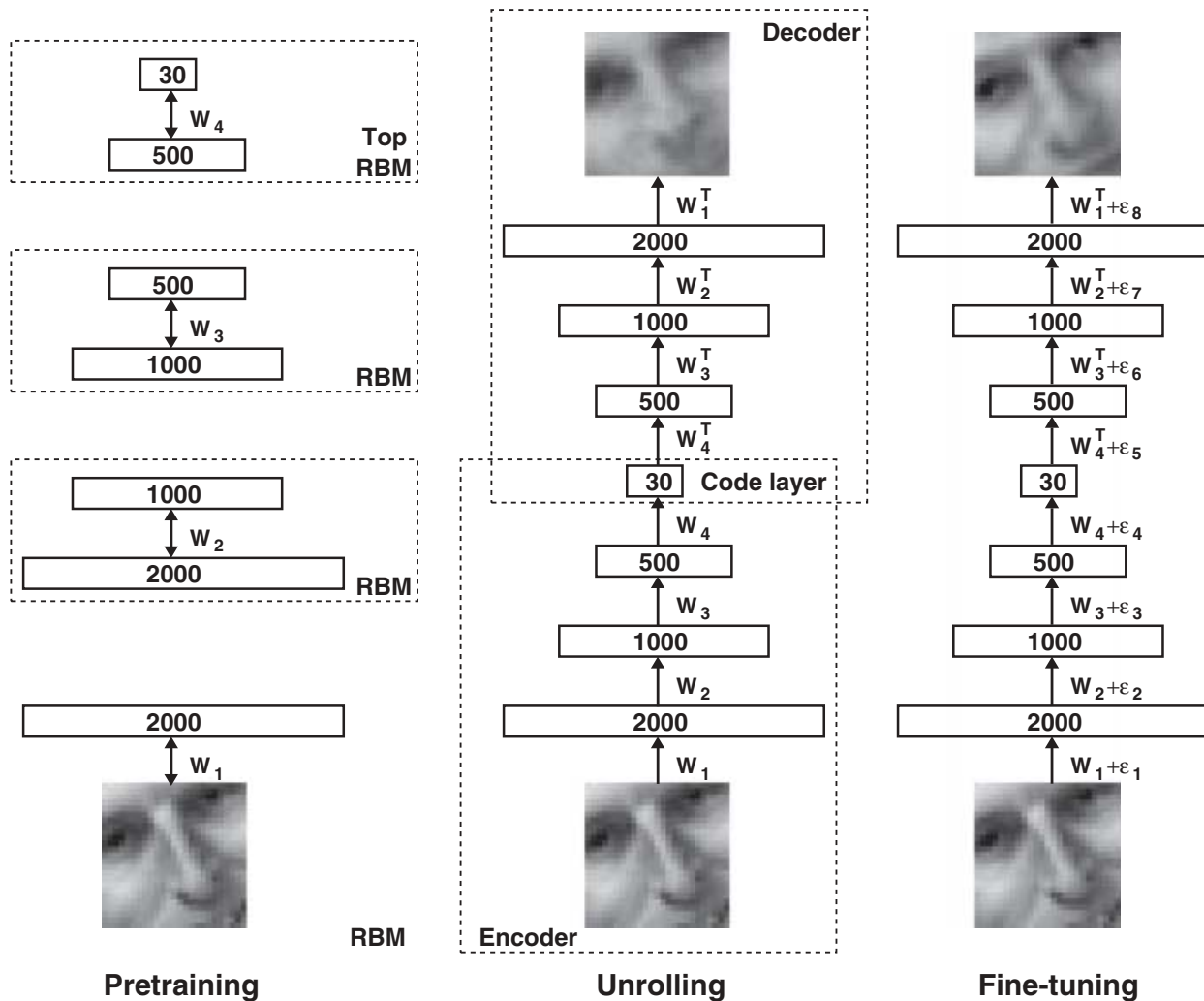


Fig. 3. (A) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (B) The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (8).

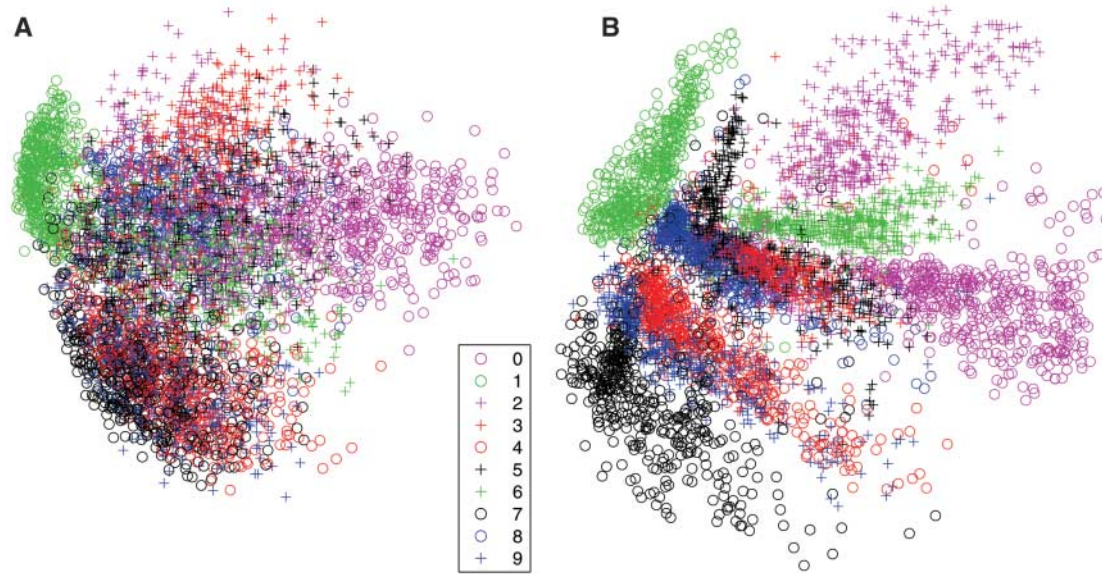
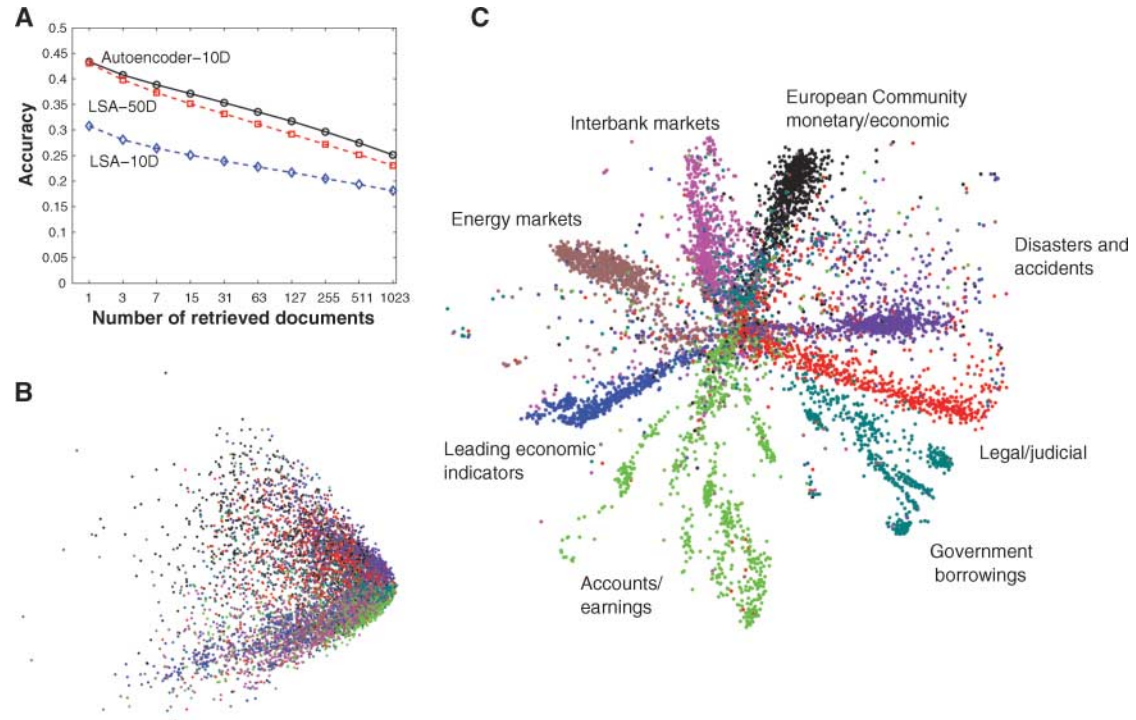
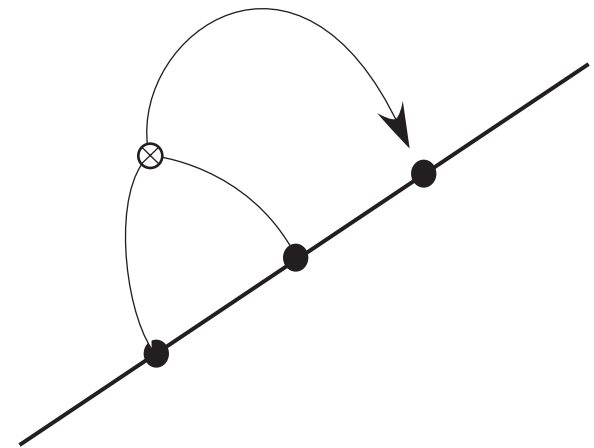
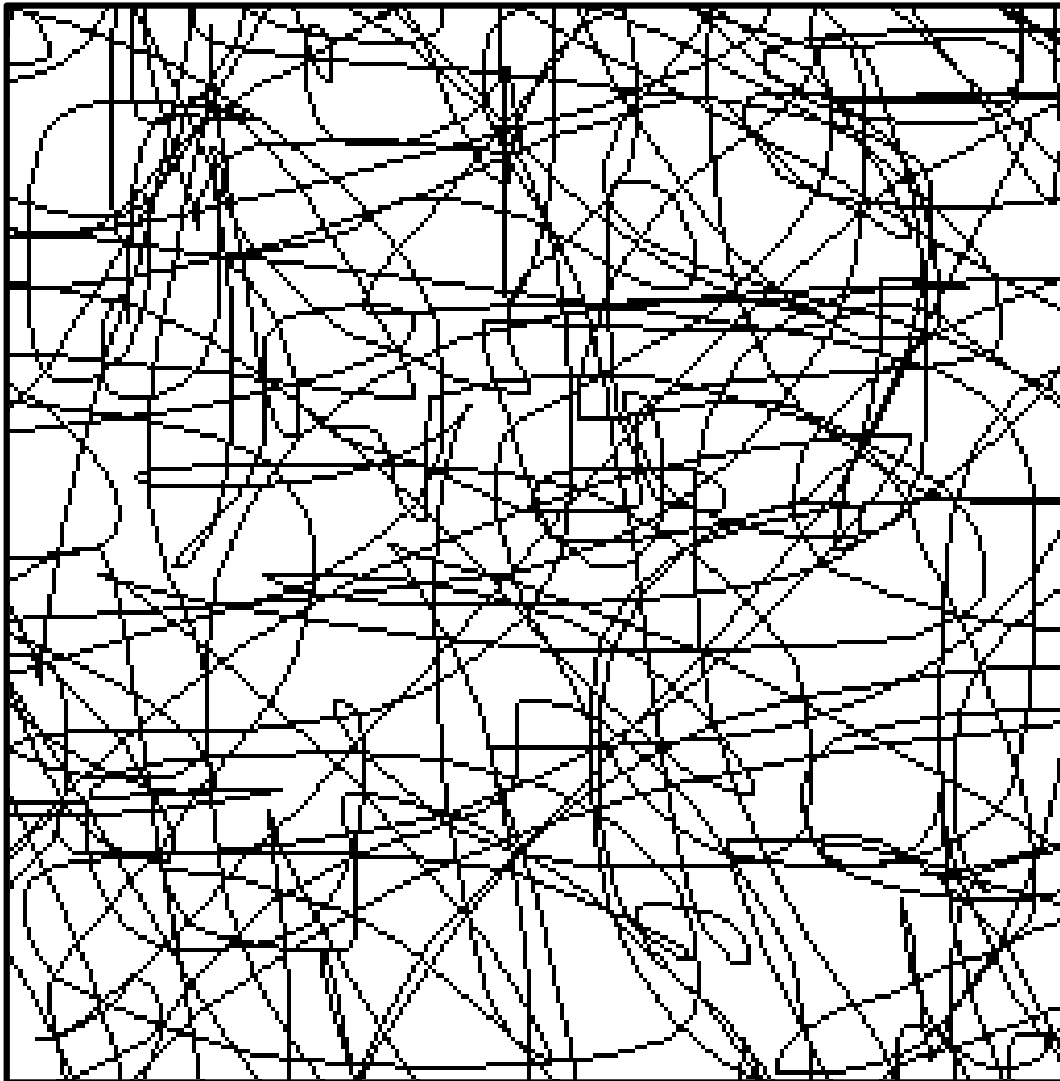
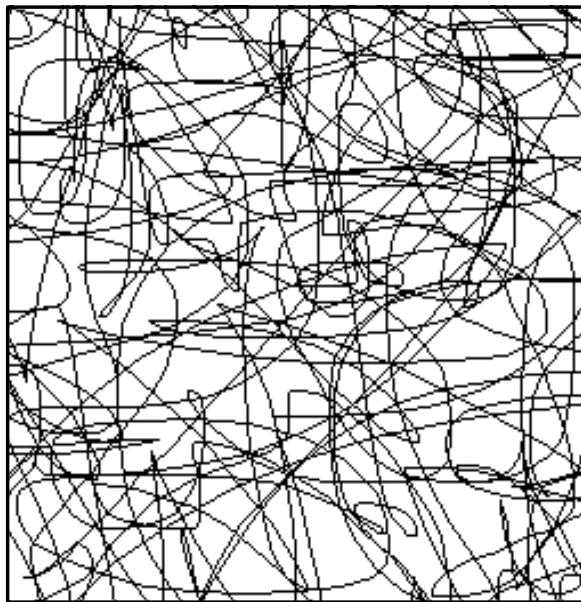


Fig. 4. (A) The fraction of retrieved documents in the same class as the query when a query document from the test set is used to retrieve other test set documents, averaged over all 402,207 possible queries. (B) The codes produced by two-dimensional LSA. (C) The codes produced by a 2000-500-250-125-2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



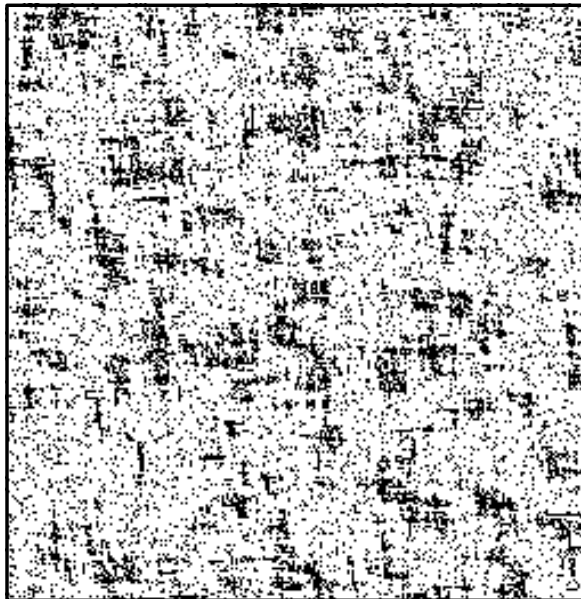
'Lines world'





Collect pairwise
statistics

Synthesize



Collect local
9-dim. pdf (3x3 blocks)

Synthesize

