Non-linear dimensionality reduction using neural networks

Ruslan Salakhutdinov
Joint work with Geoff Hinton

University of Toronto, Machine Learning Group

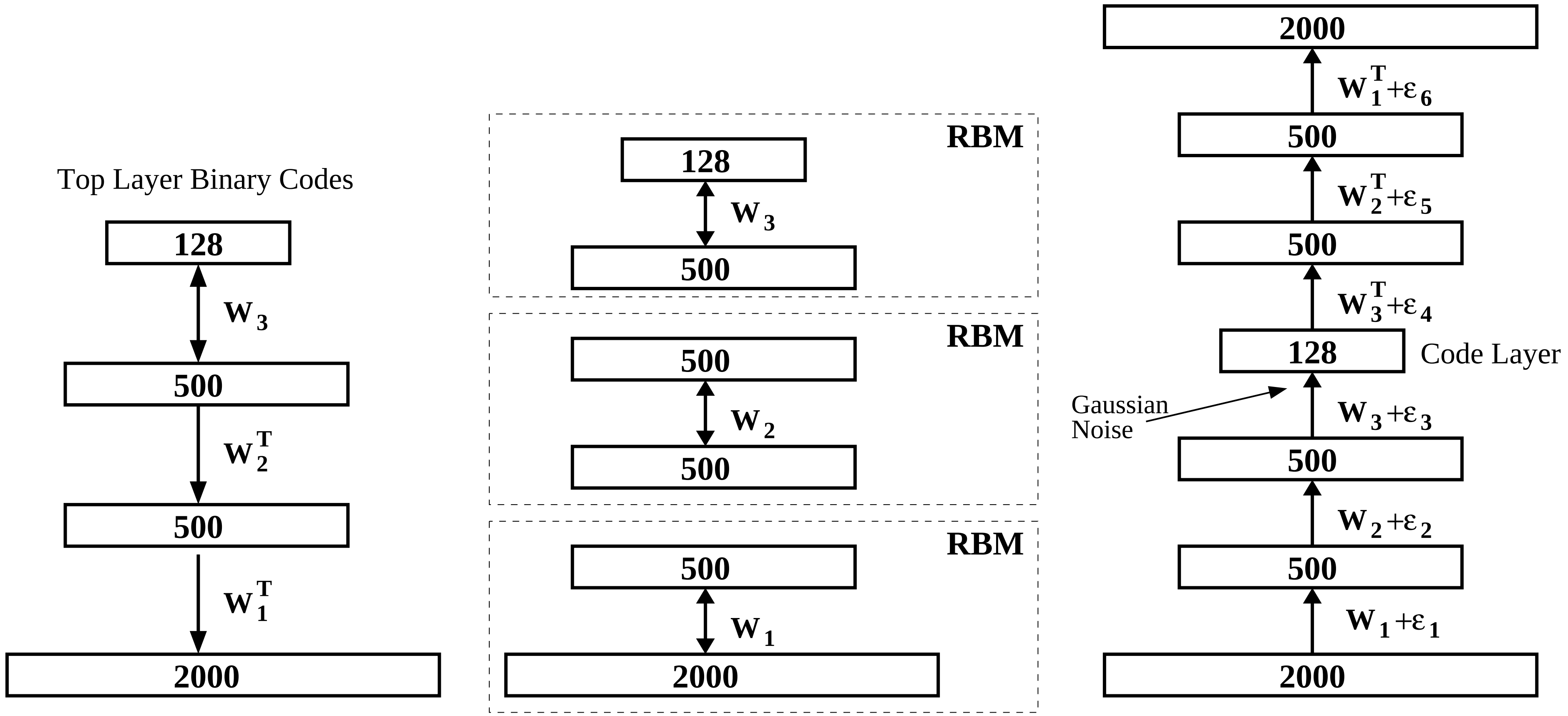
Overview

- Document Retrieval
  - Present layer-by-layer pretraining and the fine-tuning of the multi-layer network that discovers *binary* codes in the top layer. This allows us to significantly speed-up retrieval time.
  - We also show how we can use our model to allow retrieval in constant time (a time independent of the number of documents).
- Show how to perform nonlinear embedding by preserving class neighbouthood structure (supervised, semi-supervised).

Motivation

- For the document retrieval tasks, we want to retrieve a small set of documents, relevant to the given query.
- Popular and widely used in practice text retrieval algorithm is based on TF-IDF (term frequency / inverse document frequency) word-weighting heuristic.
- Drawbacks: it computes document similarity directly in the word-count space, and it does not capture high-order correlations between words in a document.
- We want to extract semantic structure "topics" from documents. Latent Semantic Analysis is a simple and widely-used linear method.
- A network with multiple hidden layers and with many more parameters should be able to discover latent representations that work much better for retrieval.

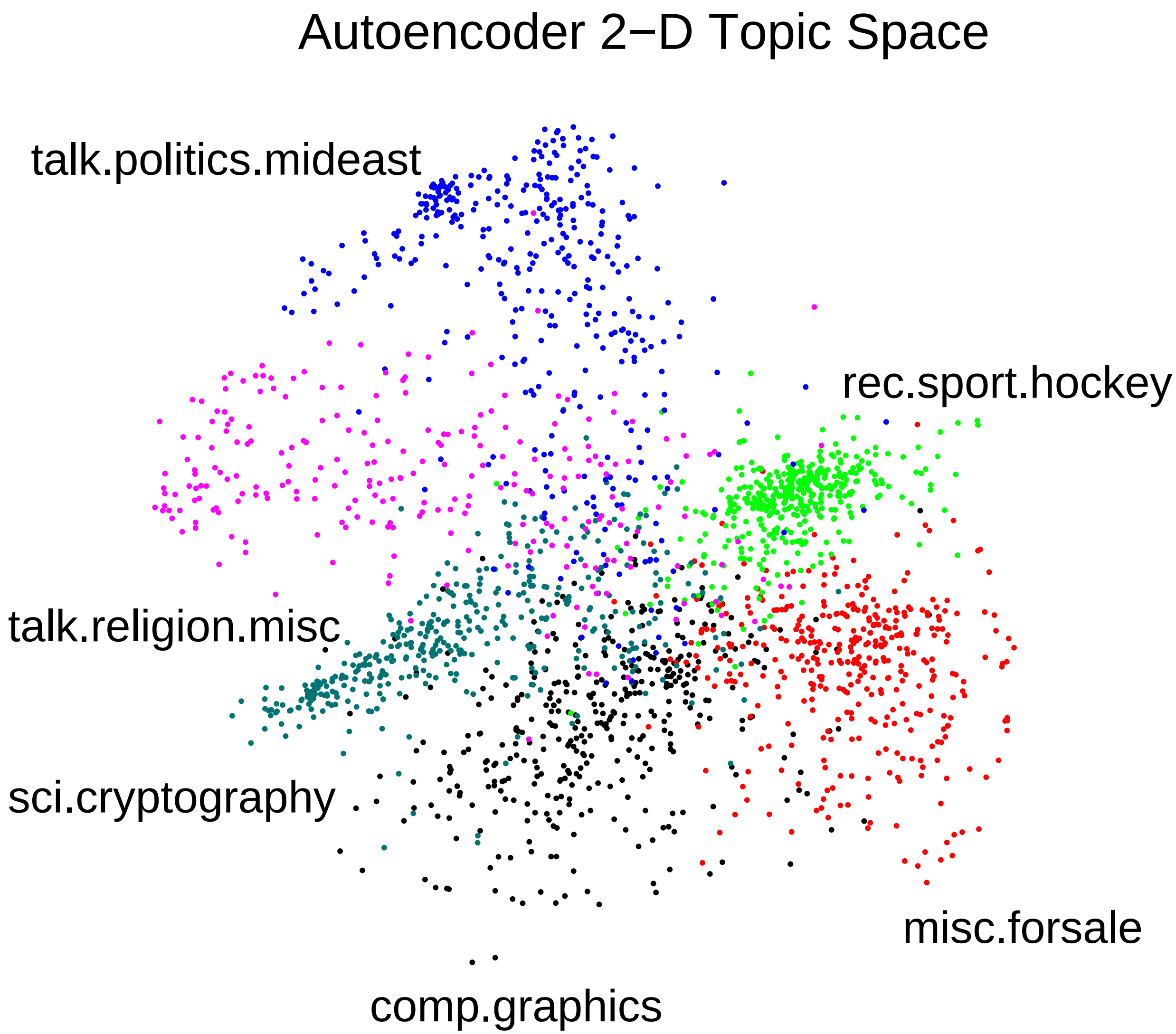
Model



The Deep Generative Model **Recursive Pretraining** **Fine−tuning**

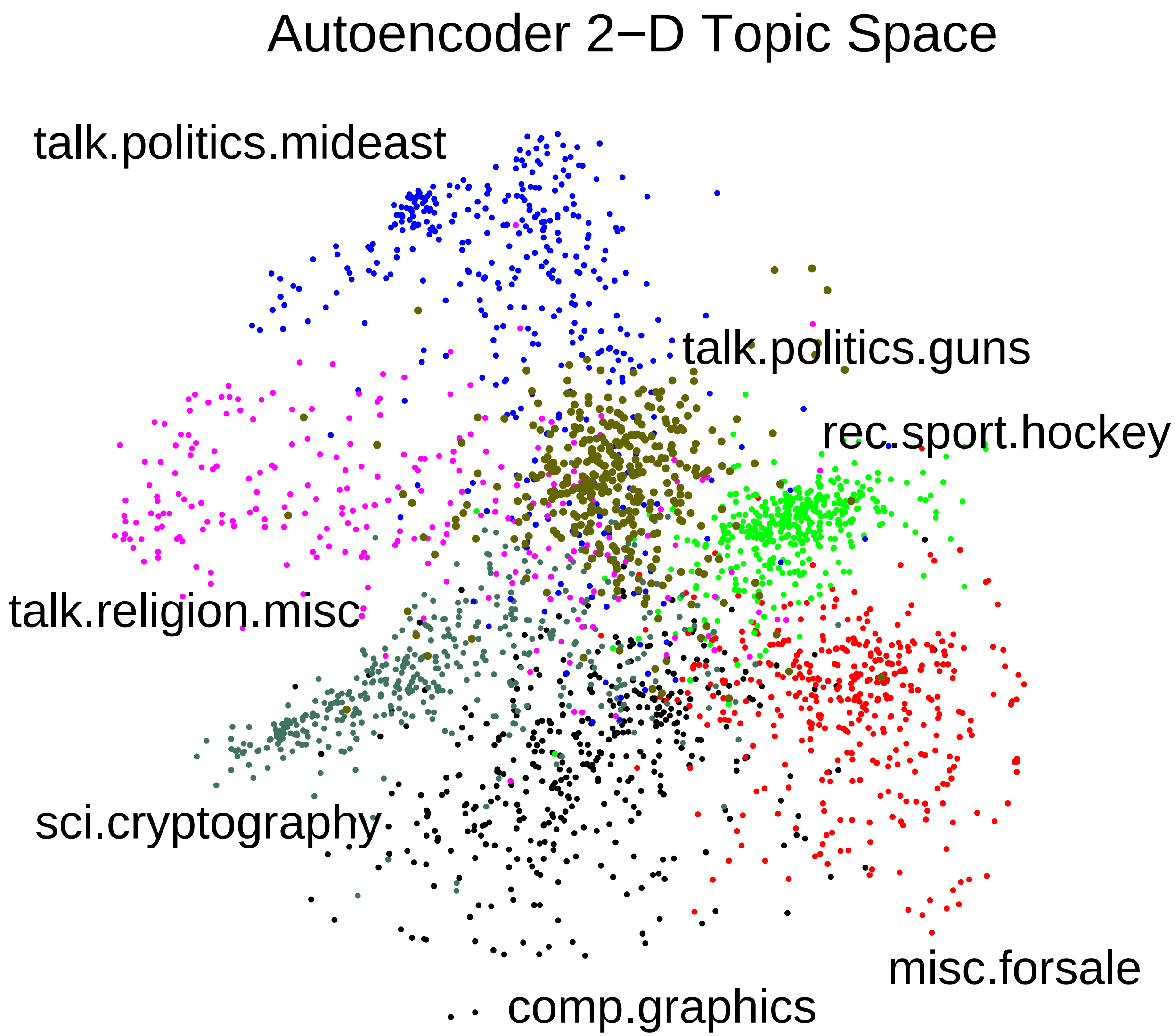
Document Retrieval: 20 newsgroup corpus

- Where should talk.politics.guns go?

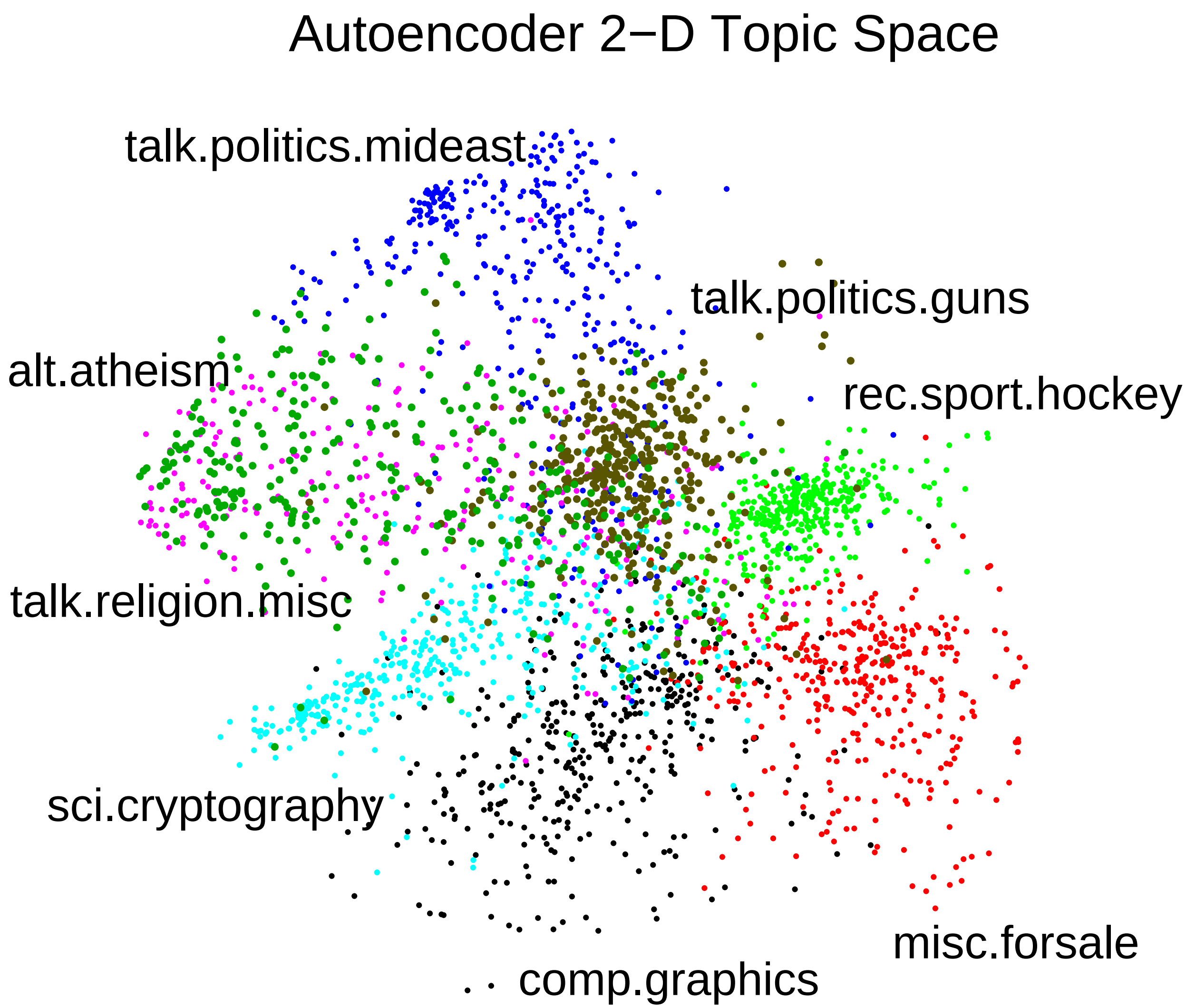


Document Retrieval: 20 newsgroup corpus

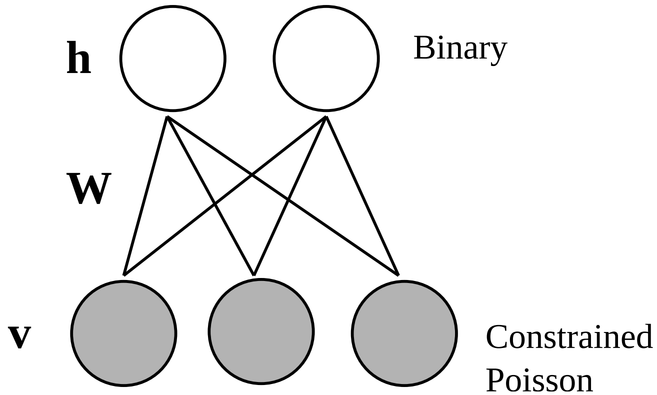
- Where should alt.atheism go?

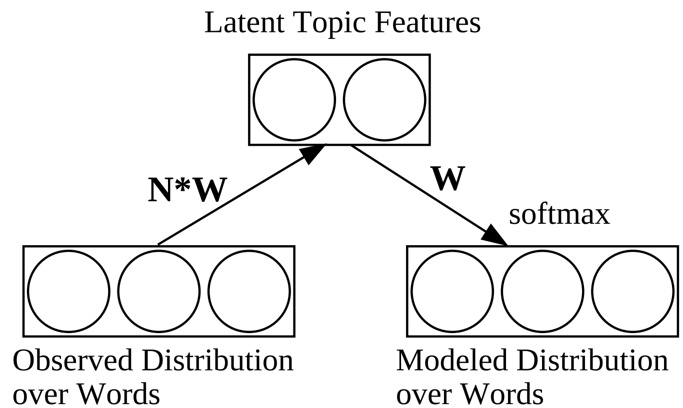


Document Retrieval: 20 newsgroup corpus



Constrained Poisson Model





- Hidden units are binary and the visible word counts are modeled by constrained Poisson model.

- Conditional distributions over hidden and visible units are:

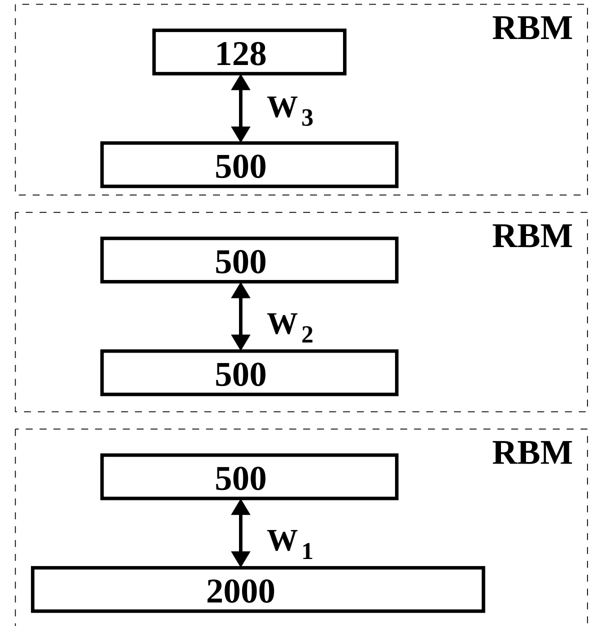
$$p(h_j = 1|\mathbf{v}) \; = \; \frac{1}{1 + \exp(-b_j - \sum_i w_{ij} v_i)}$$

$$p(v_i = n|\mathbf{h}) \; = \; \text{Poisson}\left(\frac{\exp\left(b_i + \sum_j h_j w_{ij}\right)}{Z} \times N \right)$$

where N is the total length of the document and

$$Z = \sum_i \exp\left(b_i + \sum_j h_j w_{ij}\right)$$

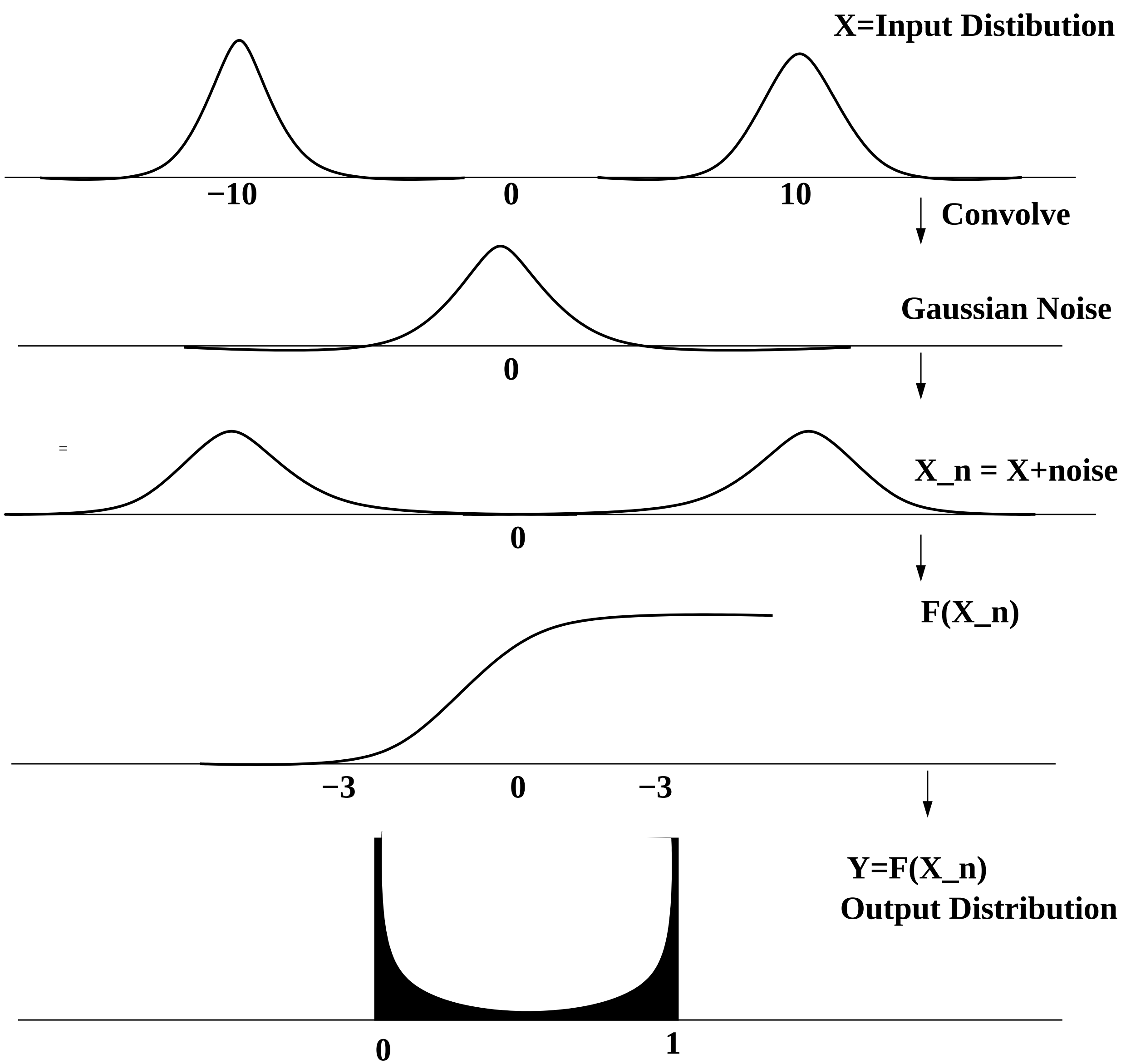
Learning Multiple Layers - Pretraining



Recursive Learning

- A single layer of binary features generally cannot perfectly model the structure in the data.
- Perform greedy, layer-by-layer learning:
  - Learn and Freeze W_1 using Constrained Poisson Model.
  - Treat the existing feature detectors, driven by training data, $W_1^T V$ as if they were data.
  - Learn and Freeze W_2.
  - Proceed recursive greedy learning as many times as desired.
- Under certain conditions adding an extra layer always improves a lower bound on the log probability of data. (In our case, these conditions are violated)
- Each layer of features captures strong high-order correlations between the activities of units in the layer belows.

Learning Binary Representations

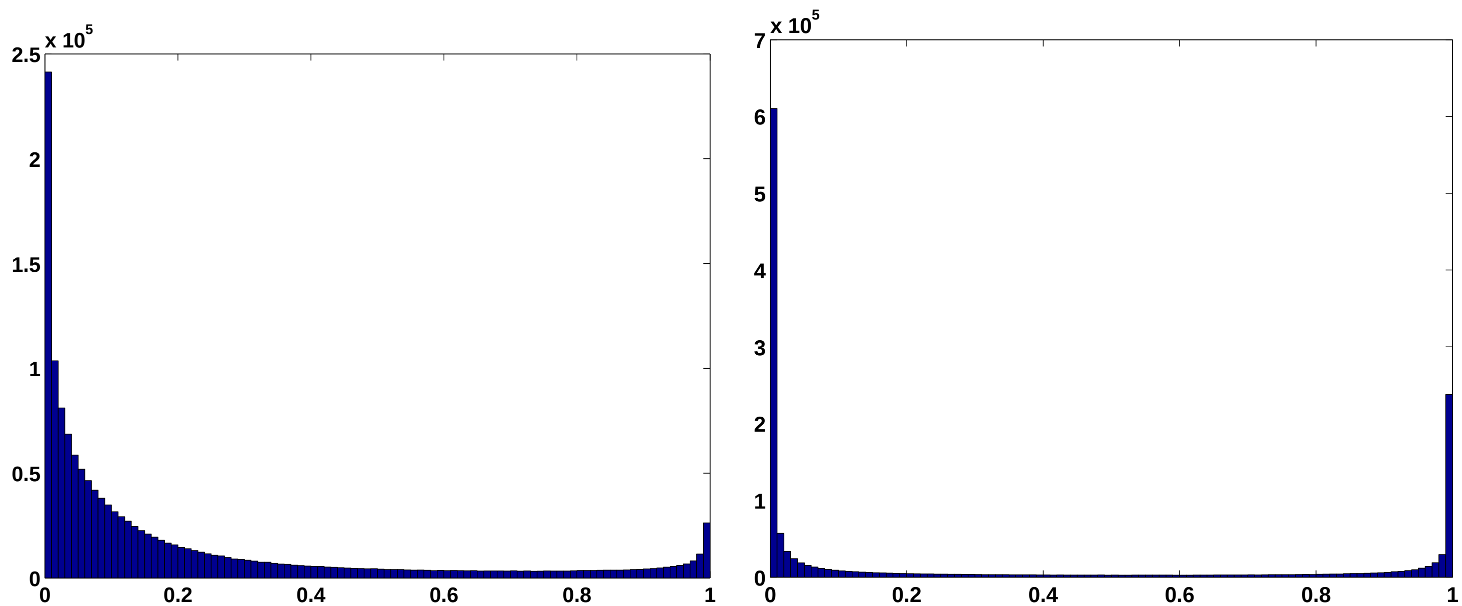


Document Retrieval: 20 Newsgroup Corpus

- We use a 2000-500-500-128 autoencoder to convert a document into a 128-bit vector.

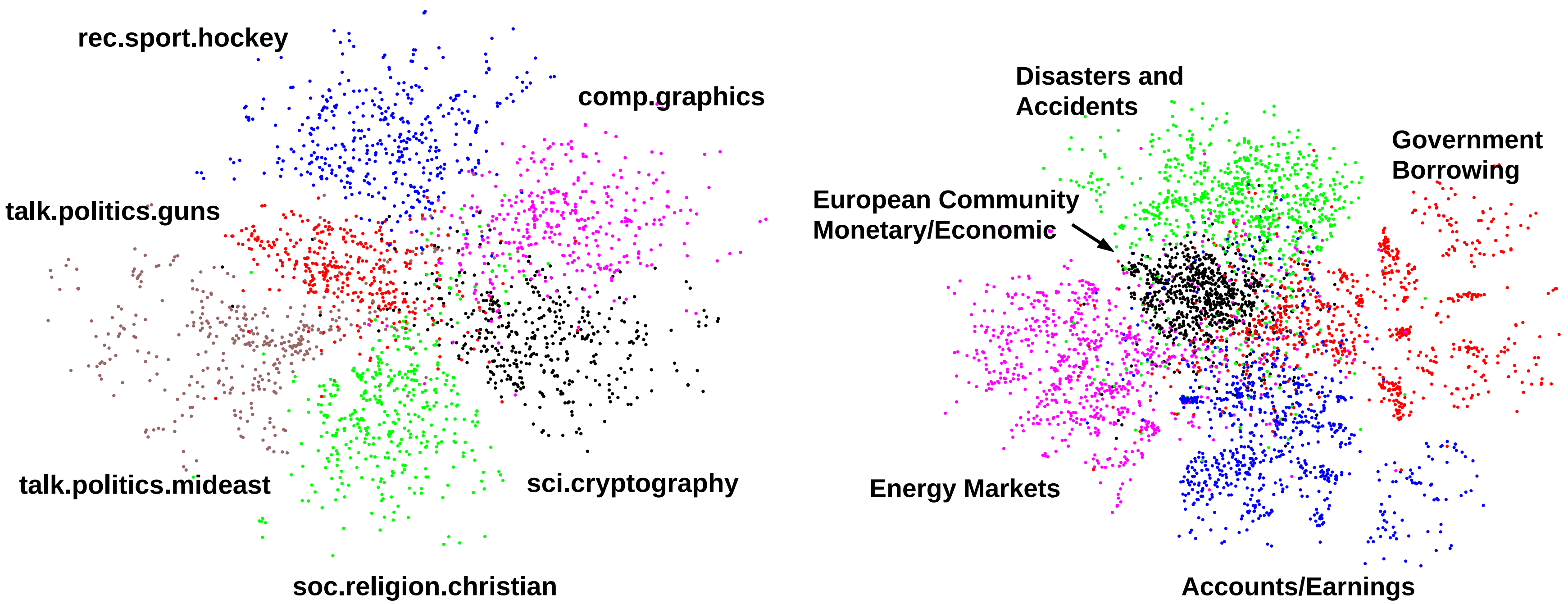
- We corrupted the input signal to the code-layer with Gaussian noise $\sim \mathcal{N}(0, 16)$.

- Empirical distributions of 128 code units before and after fine-tuning:



- After fine-tuning, we binarize codes at 0.1.

Learning Binary Representations

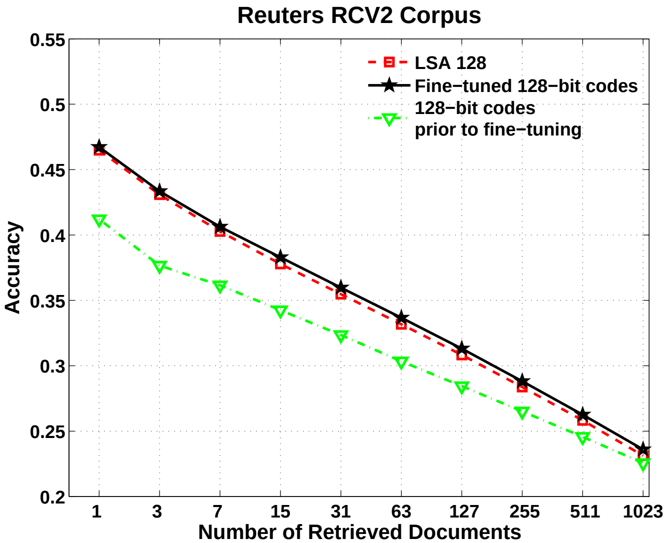


- 2-dimensional embedding of 128-bit codes using SNE for 20 Newsgroup data (left panel) and Reuters RCV2 corpus (right panel).

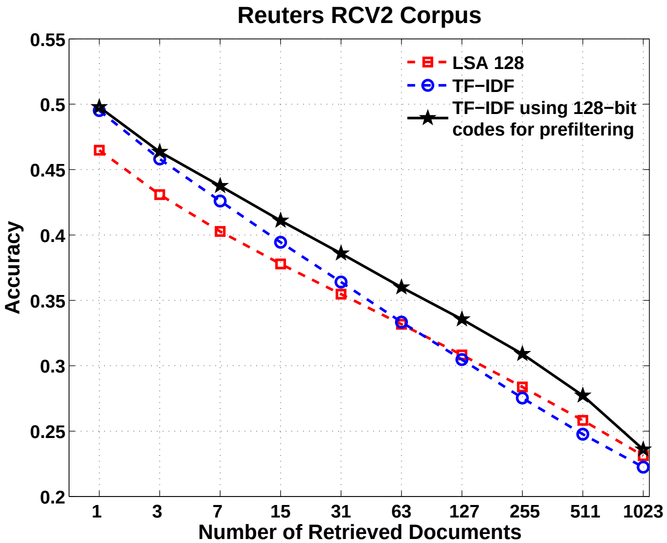
Semantic Address Space

- We could instead learn how to convert a document into a 20-bit code.
- We have ultimate retrieval tool: Given a query document, compute its 20-bit address and retrieve all of the documents stored at similar addresses **with no search at all**.
- Essentially we could learn "semantic" hashing table.
- We could also retrieve similar documents by looking at a hamming-ball of radius, for example, 4.
- The retrieved documents could then be given to a slower but more precise retrieval method, such as TF-IDF.

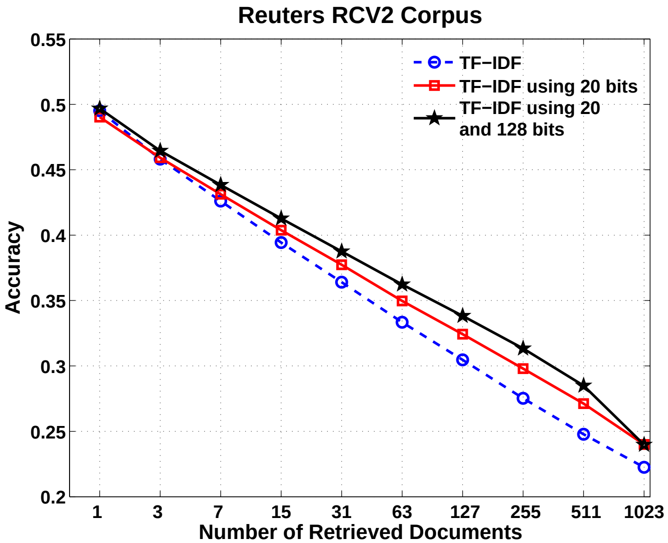
Results



Reuters RCV2 Corpus
LSA 128
Fine−tuned 128−bit codes
128−bit codes
prior to fine−tuning
Accuracy
Number of Retrieved Documents



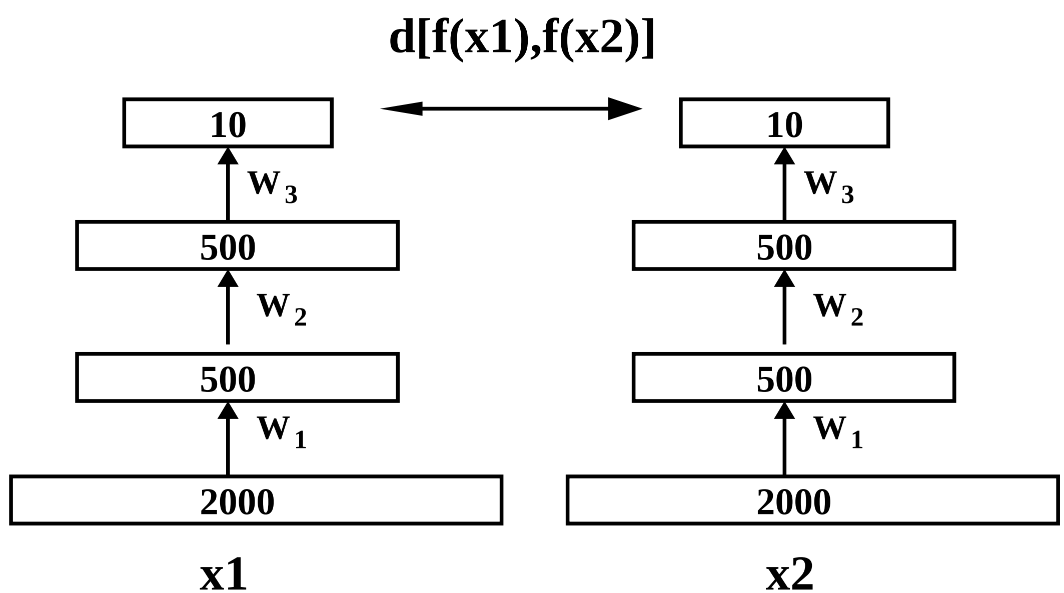
Reuters RCV2 Corpus
LSA 128
TF−IDF
TF−IDF using 128−bit
codes for prefiltering
Accuracy
Number of Retrieved Documents



Reuters RCV2 Corpus
TF−IDF
TF−IDF using 20 bits
TF−IDF using 20
and 128 bits
Accuracy
Number of Retrieved Documents

Learning nonlinear embedding

- We tackle the problem of learning similarity measure or distance metric over the input space X
- Given a distance metric d (f.e. Euclidean) we can measure similarity between two input vectors x_1,$x_2 \in X$ by computing $d[f(x_1|W), f(x_2|W)]$.
- $f(x|W)$ is a function $f : X \to Y$, mapping input vectors in X to a feature space Y and is parametarized by W.



Learning nonlinear embedding

- Most of the previous algorithms studied the case when d is Euclidean measure and $f(x|W)$ is a simple linear projection f(x—W)=Wx.

- The Euclidean distance is then the Mahalanobis distance $d[f(x_1), f(x_2)] = (x_1 - x_2)^T W^T W (x_1 - x_2)$. See Goldberger et. al. 2004, Globerson and Roweis 2005, Kilian et. al. 2005.

- We have a set of N training labeled data vectors (x_i, c_i), where $x_i \in R^d$, and $c_i \in 1, 2, ..., K$.

- For each training vector x_i, define the probability that point i selects one of its neighbours j in the transformed space as:

$$p_{ij} = \frac{\exp(-d_{ij})}{\sum_{k \neq i} \exp(-d_{ik})}, \qquad p_{ii} = 0$$

  where $d_{ij} = \| f(x_i|W) - f(x_j|W) \|^2$, and $f(\cdot|W)$ is a multi-layer perceptron.

Learning nonlinear embedding

- Probability that point i belongs to class k is:

$$p(c_i = k) = \sum_{j|c_j=k} p_{ij}$$

- Mazimize the expected number of correctly classified points on the training data:

$$\text{NCA} = \sum_{i=1}^{N} \sum_{j|c_i=c_j} p_{ij}$$

- One could alternatively minimize KL-divergence:

$$KL(p^0|p) = \sum_{i=1}^{N} \sum_{k=1}^{K} p_{ik}^0 \log \frac{p_{ik}^0}{p(c_i=k)}$$

where

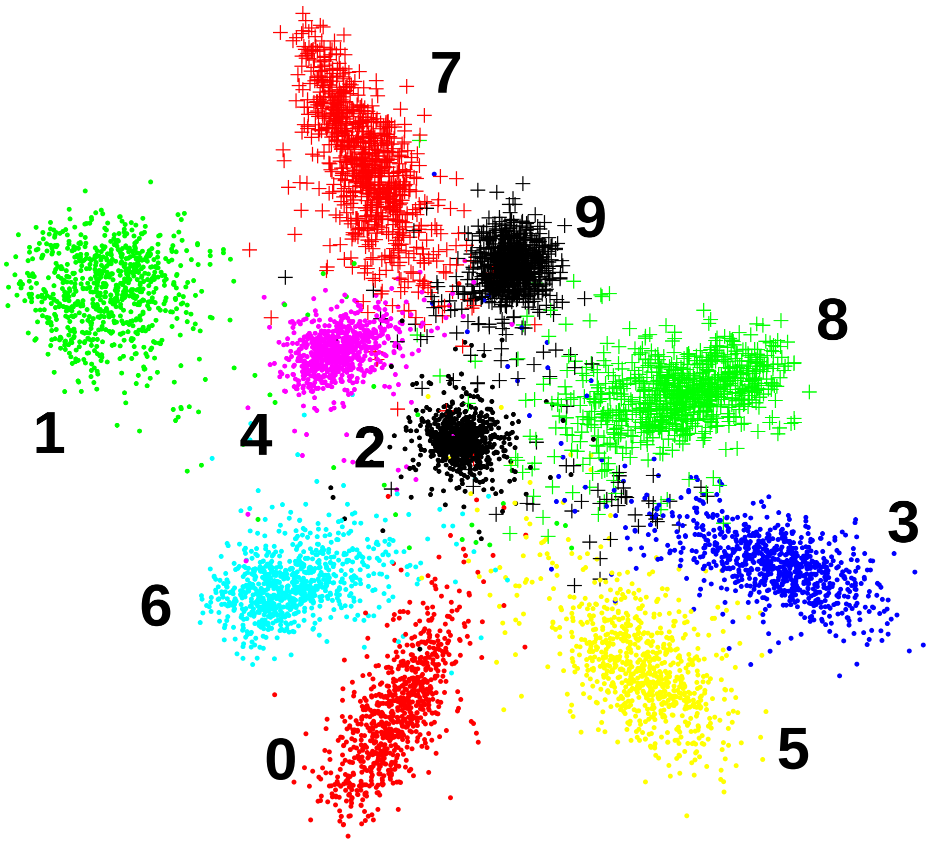
$$p_{ik}^0 = \begin{cases} 1 & \text{if } c_i = k \\ 0 & \text{if } c_i \neq k \end{cases}$$

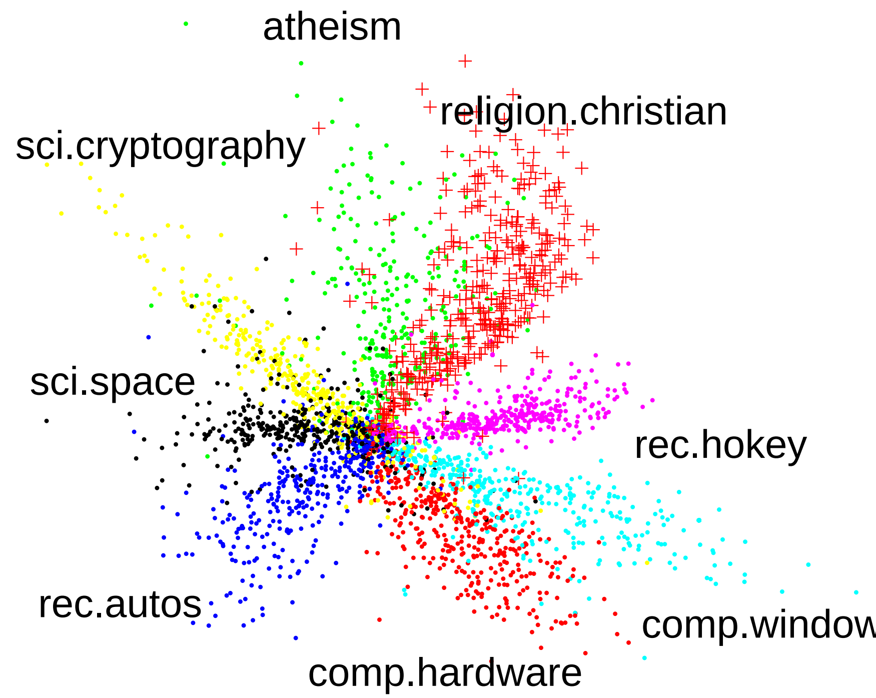
- This induces the following objective function to maximize:

$$\text{NCA}_2 = \sum_{i=1}^{N} \log \left(\sum_{j|c_i=c_j} p_{ij} \right)$$

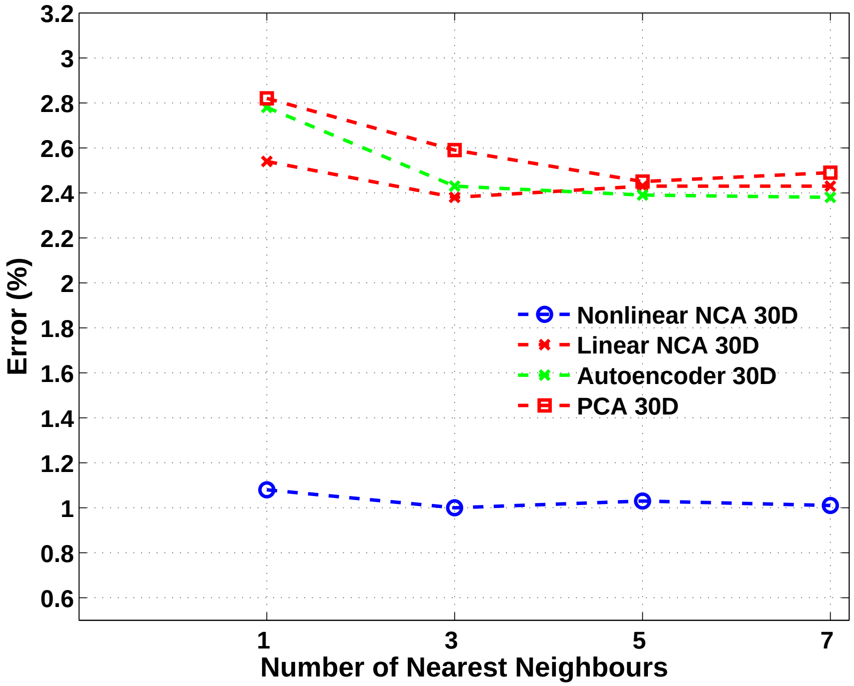
- By considering a linear perceptron we arrive to linear NCA.

Results

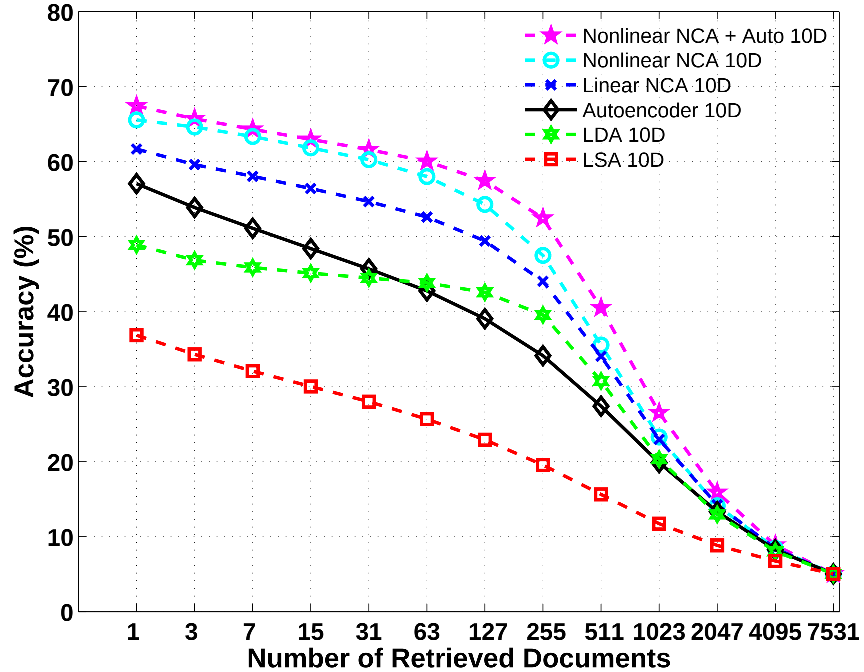

7
9
8
1
4
2
3
6
0
5


atheism
religion.christian
sci.cryptography
sci.space
rec.hokey
rec.autos
comp.window
comp.hardware

Results

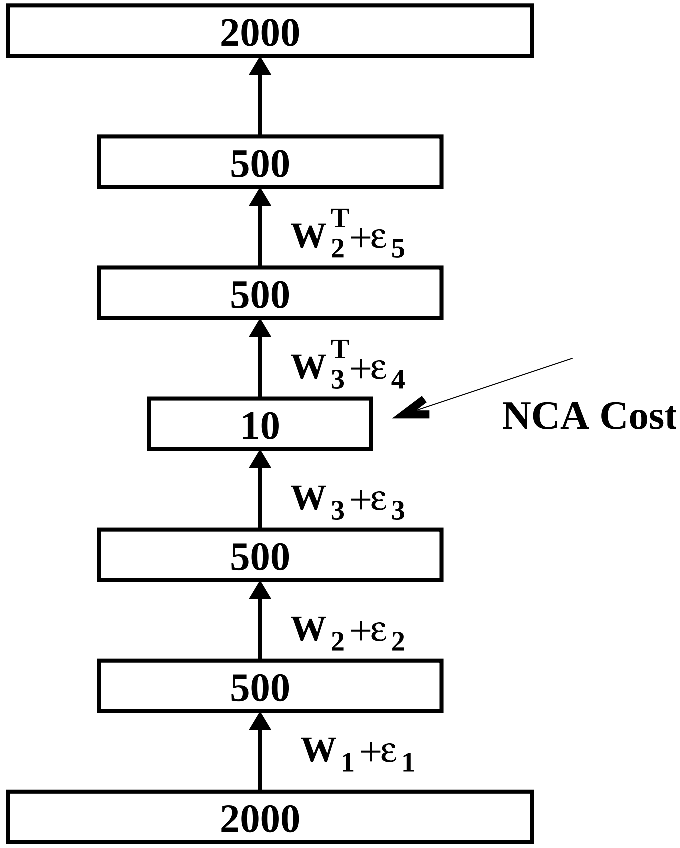


Error (%)
Number of Nearest Neighbours
Nonlinear NCA 30D
Linear NCA 30D
Autoencoder 30D
PCA 30D



Accuracy (%)
Number of Retrieved Documents
Nonlinear NCA + Auto 10D
Nonlinear NCA 10D
Linear NCA 10D
Autoencoder 10D
LDA 10D
LSA 10D

Semi-supervised Extention



- The combined objective we maximize:

$$C = \lambda * \mathrm{NCA} + (1 - \lambda) * (-E)$$

- So the derivative of the reconstruction error E is backpropagated through the autoencoder and is combined with derivatives of NCA at the code level.

Semi-supervised Extention