Deep Belief Networks with applications to Fast Document Retrieval

Ruslan Salakhutdinov

joint work with Geoffrey Hinton

University of Toronto, Machine Learning Group

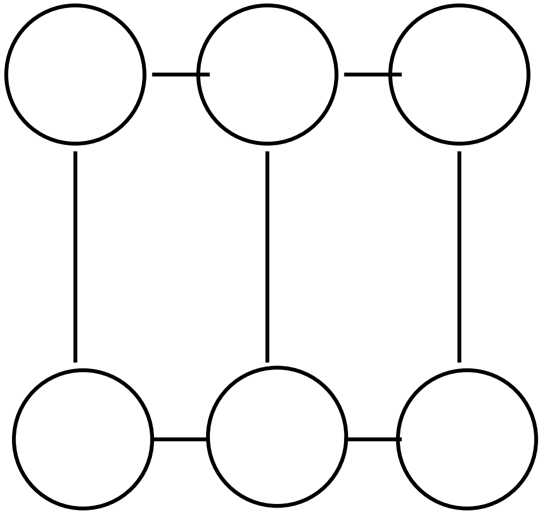
IPAM workshop

October 24, 2007

Talk outline

- Markov Random Fields (MRF's) and Restricted Boltzmann Machines (RBM's)
- Deep Belief Nets as stacks of Restricted Boltzmann Machines.
  - Nonlinear Dimensionality Reduction.
  - Discriminative Fine-tuning for Regression and Classification.
  - Deep Belief Nets as Generative Models.
- Semantic Hashing for Fast Document Retrieval.

Markov Random Fields



- A Markov Random Field (undirected graphical model) is a model of the joint probability distribution over a set of random variables.

- It contains a set of nodes (vertices) that represent random variables, and a set of undirected links (edges) that represent dependencies between those random variables.

- The joint distribution takes the form of the product of non-negative potential functions ψ_C over the maximal cliques (connected subsets of nodes):

$$p(X = x) = \frac{1}{Z} \prod_C \psi_C(x_C) \qquad Z = \sum_x \prod_C \psi_C(x_C)$$

- The normalizing constant Z is called a partition function.

- Computing Z is often very hard. This represents a major limitation of undirected graphical models.

Markov Random Fields

$$p(X = x) = \frac{1}{Z} \prod_C \psi_C(x_C) \qquad Z = \sum_x \prod_C \psi_C(x_C)$$

- Each ψ_C is a mapping from joint configurations of random variables in clique C to non-negative real numbers.

- The choice of potential functions is not restricted to having specific probabilistic interpretations.

- Potential functions are often represented as exponentials:

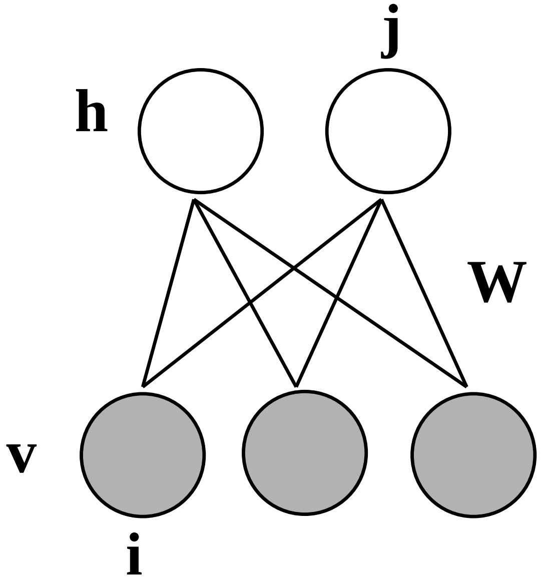
  $$\psi_C(x_C) = \exp\left(-E_C(x_C)\right)$$

  where real-valued function $E_C(x_C)$ is called an energy function. Thus:

  $$p(X = x) = \frac{1}{Z} \prod_C \psi_C(x_C) = \frac{1}{Z} \exp\left(-\sum_C E_C(x_C)\right) = \frac{1}{Z} \exp\left(-E(x)\right)$$

Restricted Boltzmann Machines

- We can model an ensemble of binary images using Restricted Boltzmann Machines (RBM's).



- An RBM is a type of MRF with hidden units.

- RBM's have a two-layer architecture in which visible, binary stochastic pixels **v** are connected to hidden binary stochastic feature detectors **h**.

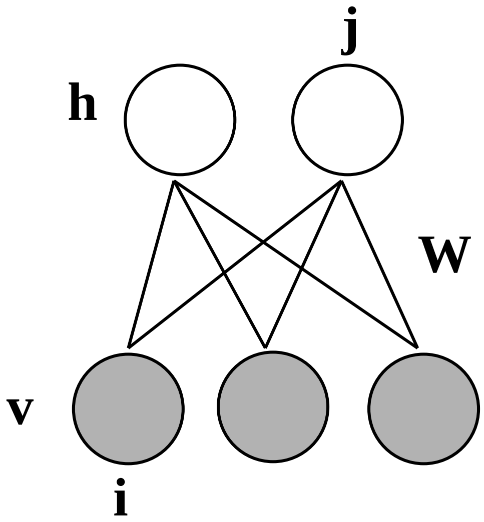
- A joint configuration $(\mathbf{v}, \mathbf{h})$ has an energy:

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i \in \text{pixels}} b_i v_i - \sum_{j \in \text{features}} b_j h_j - \sum_{i,j} v_i h_j W_{ij}$$

- The probability that the model assigns to **v** is

$$p(\mathbf{v}) = \sum_{\mathbf{h} \in \mathcal{H}} p(\mathbf{v},\mathbf{h}) = \sum_{\mathbf{h} \in \mathcal{H}} \frac{\exp(-E(\mathbf{v},\mathbf{h}))}{\sum_{\mathbf{u},\mathbf{g}} \exp(-E(\mathbf{u},\mathbf{g}))}$$

Inference



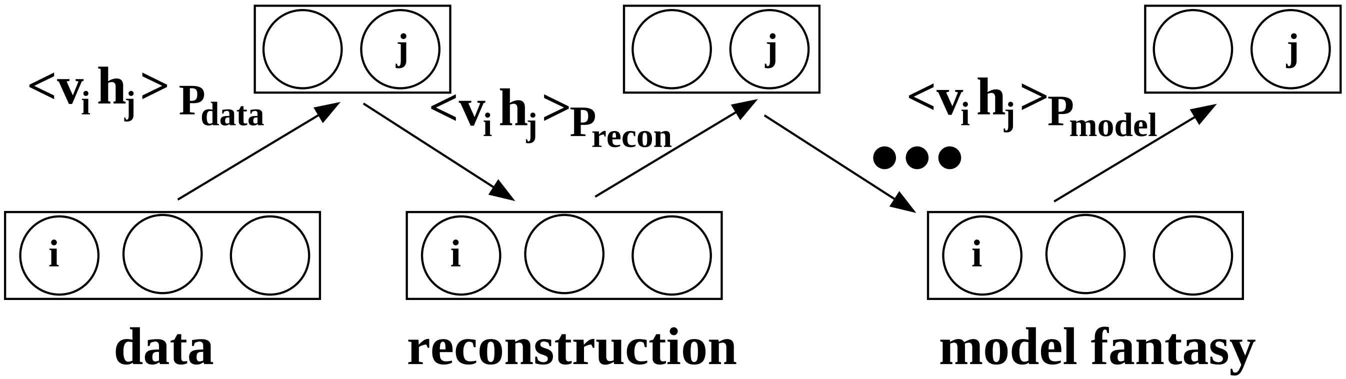
- Conditional distributions over hidden and visible units are given by logistic functions:

$$p(h_j = 1|\mathbf{v}) = \frac{1}{1 + \exp(-b_j - \sum_i v_i W_{ij})}$$
$$p(v_i = 1|\mathbf{h}) = \frac{1}{1 + \exp(-b_i - \sum_j h_j W_{ji})}$$

- Note that due to RBM's special architecture, for a given data vector **v**, posterior over **h** factors:

$$p(\mathbf{h}|\mathbf{v}) = \prod_j p(h_j|\mathbf{v})$$

Learning



j
j
j
<v_i h_j> P_{data}
<v_i h_j>P_{recon}
<v_i h_j>P_{model}
i
i
i
data
reconstruction
model fantasy

- Maximum Likelihood learning is hard:

$$\frac{\partial \log p(\mathbf{v})}{\partial W_{ij}} = \mathrm{E}_{P_{data}}[v_i h_j] - \mathrm{E}_{P_{model}}[v_i h_j]$$

due to the second term. One way of estimating this expectation is via MCMC → slow.

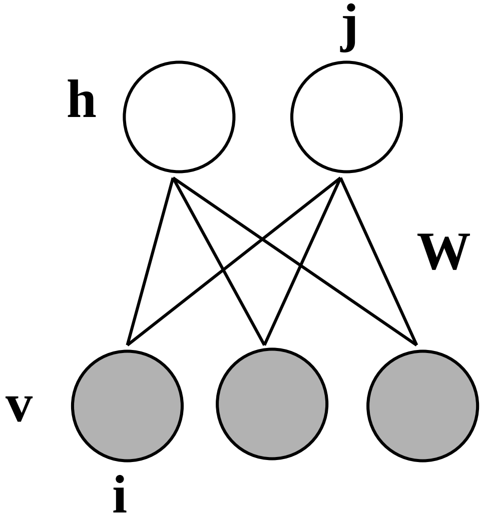
- Instead, we will use Contrastive Divergence (1-step) learning:

$$\Delta W_{ij} = \mathrm{E}_{P_{data}}[v_i h_j] - \mathrm{E}_{P_{recon}}[v_i h_j]$$

RBM's for continuous data

- Hidden units remain binary.

- The visible units are replaced by linear stochastic units that have Gaussian noise.



- The energy becomes:

$$E(\mathbf{v},\mathbf{h}) = \sum_{i \in \text{pixels}} \frac{(v_i - b_i)^2}{2\sigma_i^2} - \sum_{j \in \text{features}} b_j h_j - \sum_{i,j} \frac{v_i}{\sigma_i} h_j W_{ij}$$

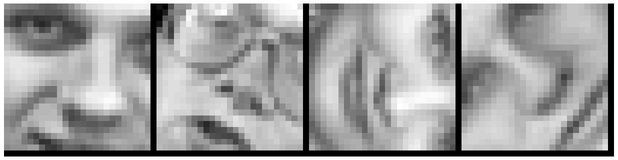
- Conditional distributions over hidden and visible units are:

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

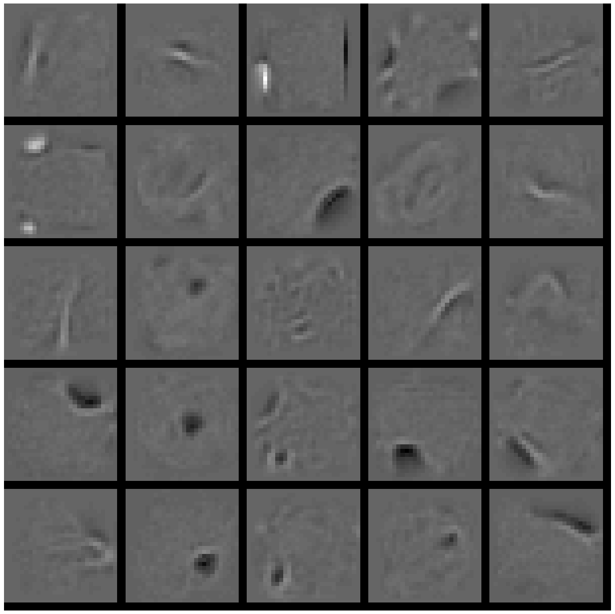
$$p(v_i = v|\mathbf{h}) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\Big(- \frac{(v - b_i - \sigma_i \sum_j h_j W_{ij})^2}{2\sigma_i^2}\Big)$$

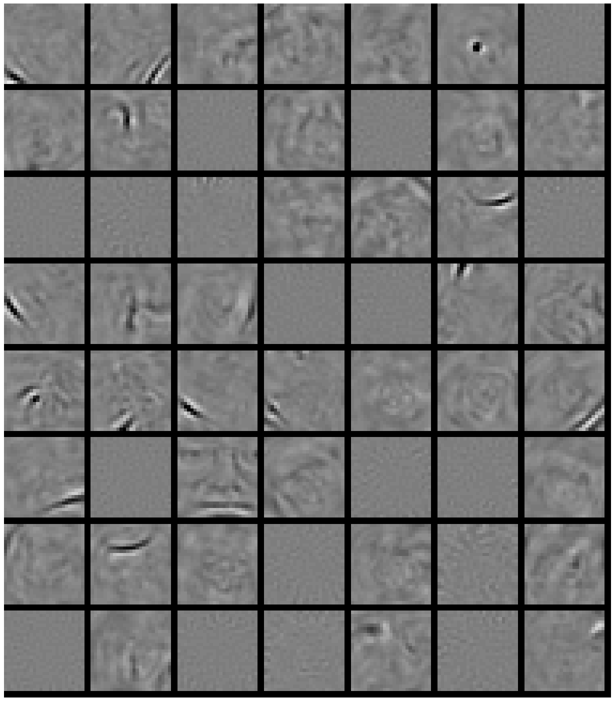
What a single RBM learns

- Random sample of the RBM's receptive fields (W) for MNIST (left) and Olivetti (right).
- Input data



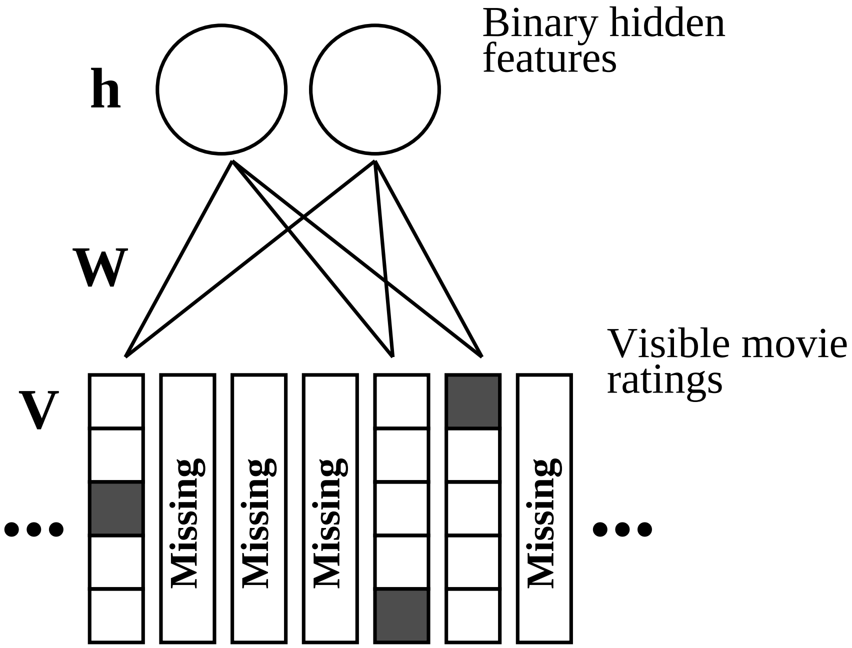
- Learned W





RBM's for Collaborative Filtering

- A restricted Boltzmann machine can also be applied to collaborative filtering. In particular, it performs quite well on the Netflix movie rating problem.



- A restricted Boltzmann machine with binary hidden units and softmax visible units.

- For each user, the RBM includes softmax units for the movies that a user has rated.

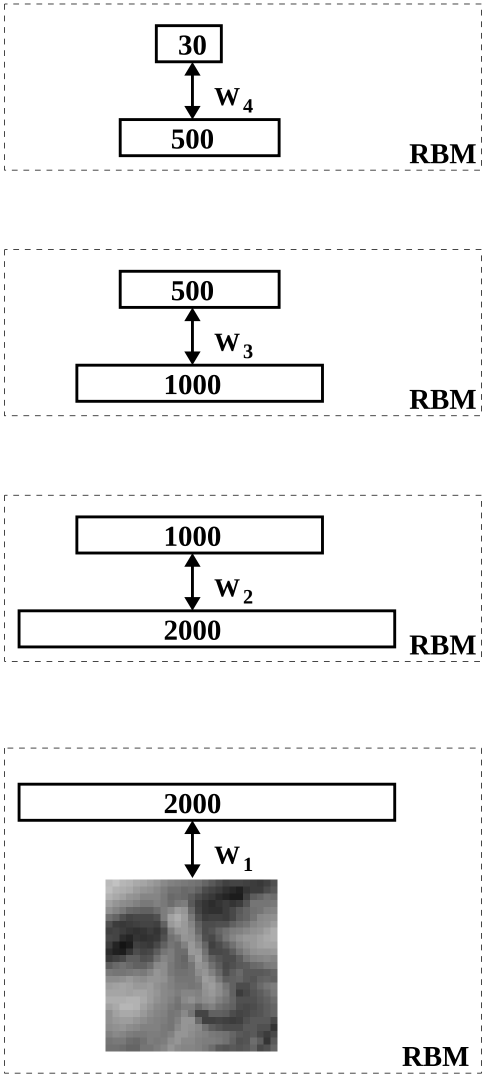
- We let **h** represent stochastic binary hidden features that have different values for different users.

(See Salakhutdinov, Mnih, and Hinton, ICML 2007)

Learning Stacks of RBM's

- 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.
  - Treat the existing feature detectors, driven by training data, $\sigma(W_1^T V)$ as if they were data.
  - Learn and Freeze W_2.
  - Greedily learn as many layers of features as desired.



- Under certain conditions adding an extra layer always improves a lower bound on the log probability of data (explained later).

- Each layer of features captures strong high-order correlations between the activities of units in the layer below.

Deep Belief Nets with applications to Nonlinear Dimensionality Reduction

Dimensionality Reduction

- Problem: How to discover low-dimensional structure from high-dimensional observations.

- The compact representation can be used for exploratory data analysis, preprocessing, data visualization and compression.

- Variety of dimensionality reduction techniques:
  - Linear methods (such as Principal Component Analysis)
  - Non-linear mappings (such as autoencoders)
  - Proximity based methods (such as Local Linear Embedding)

Drawbacks of Existing Methods

- Linear Methods:
  - If the data lie on an embedded low-dimensional nonlinear manifold then linear methods cannot recover this structure.

- Proximity based methods are more powerful, BUT
  - computational cost scales quadratically with the number of observations.
  - cannot be applied to very large high-dimensional data sets.

- Nonlinear mapping algorithms, such as autoencoders:
  - painfully slow to train.
  - prone to getting stuck in local minima.

Nonlinear Dimensionality Reduction

- Perform greedy, layer-by-layer pretraining.

- After pretraining multiple layers, the model is unrolled to create a deep autoencoder.

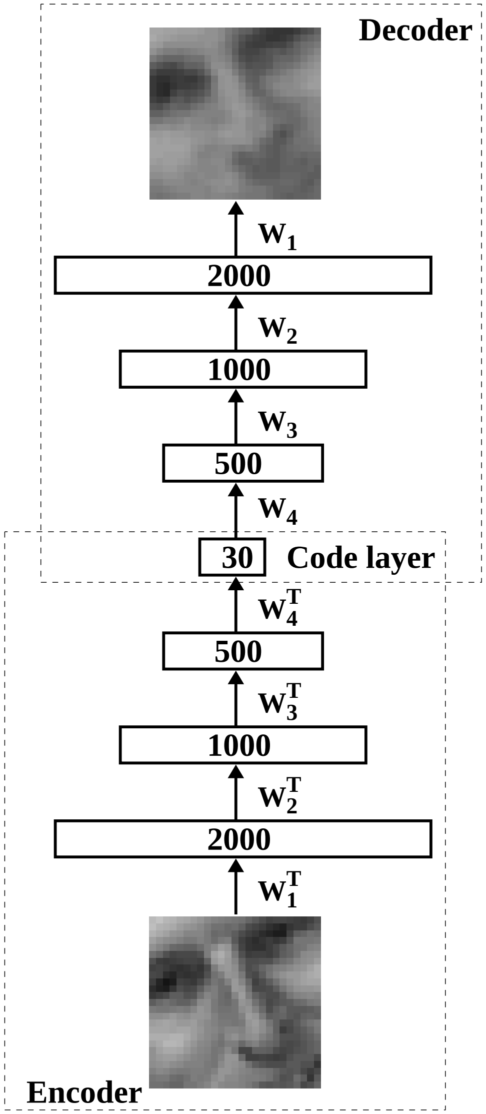
- Initially encoder and decoder networks use the same weights.

- The global fine-tuning uses backpropagation through the whole autoencoder to fine-tune the weights for optimal reconstruction.

- Backpropagation only has to do local search.

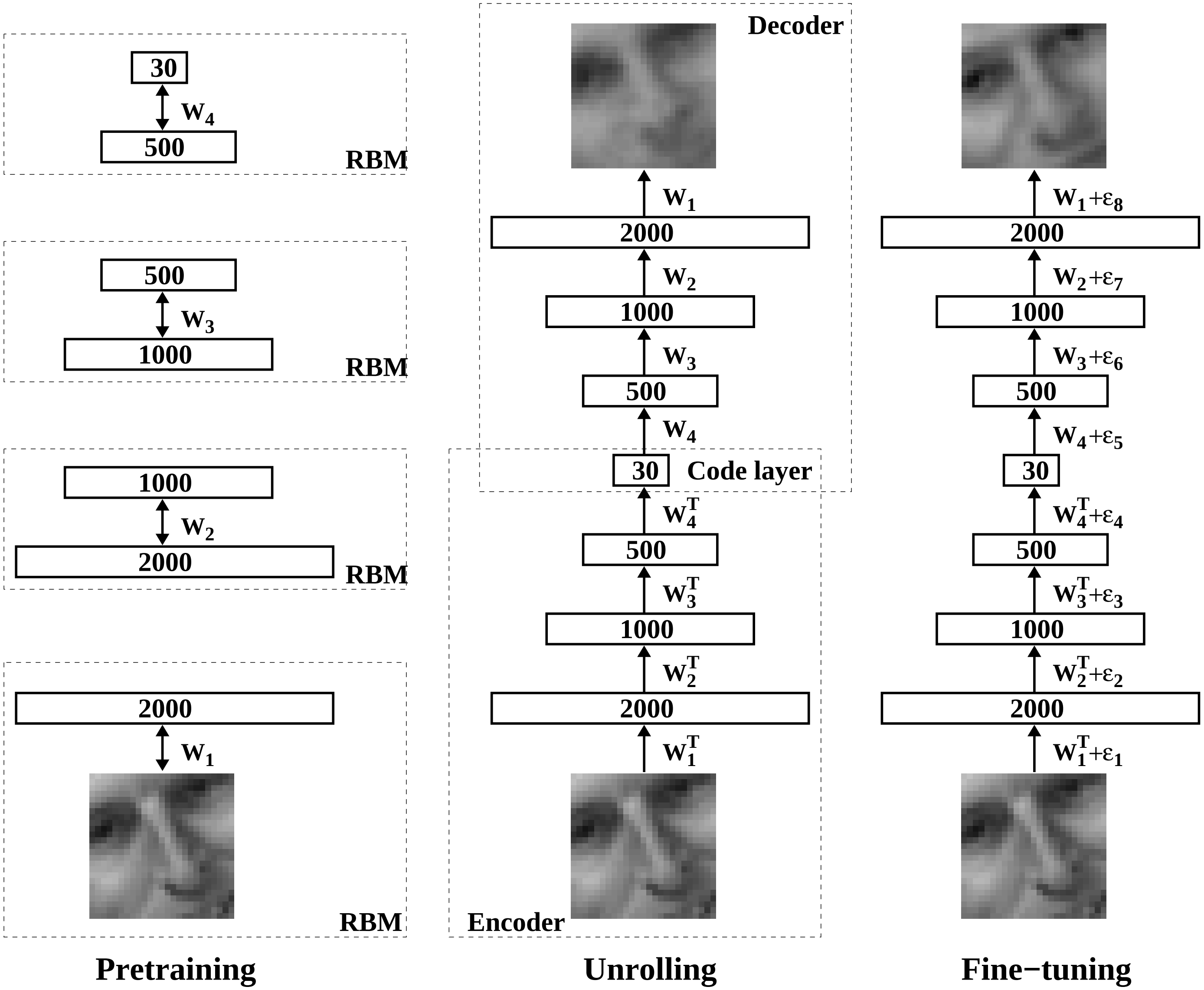
- We used a 625-2000-1000-500-30 autoencoder to extract 30-D real-valued codes for Olivetti face patches (7 hidden layers is usually hard to train).

- We used a 784-1000-500-250-30 autoencoder to extract 30-D real-valued codes for MNIST images.



Unrolling

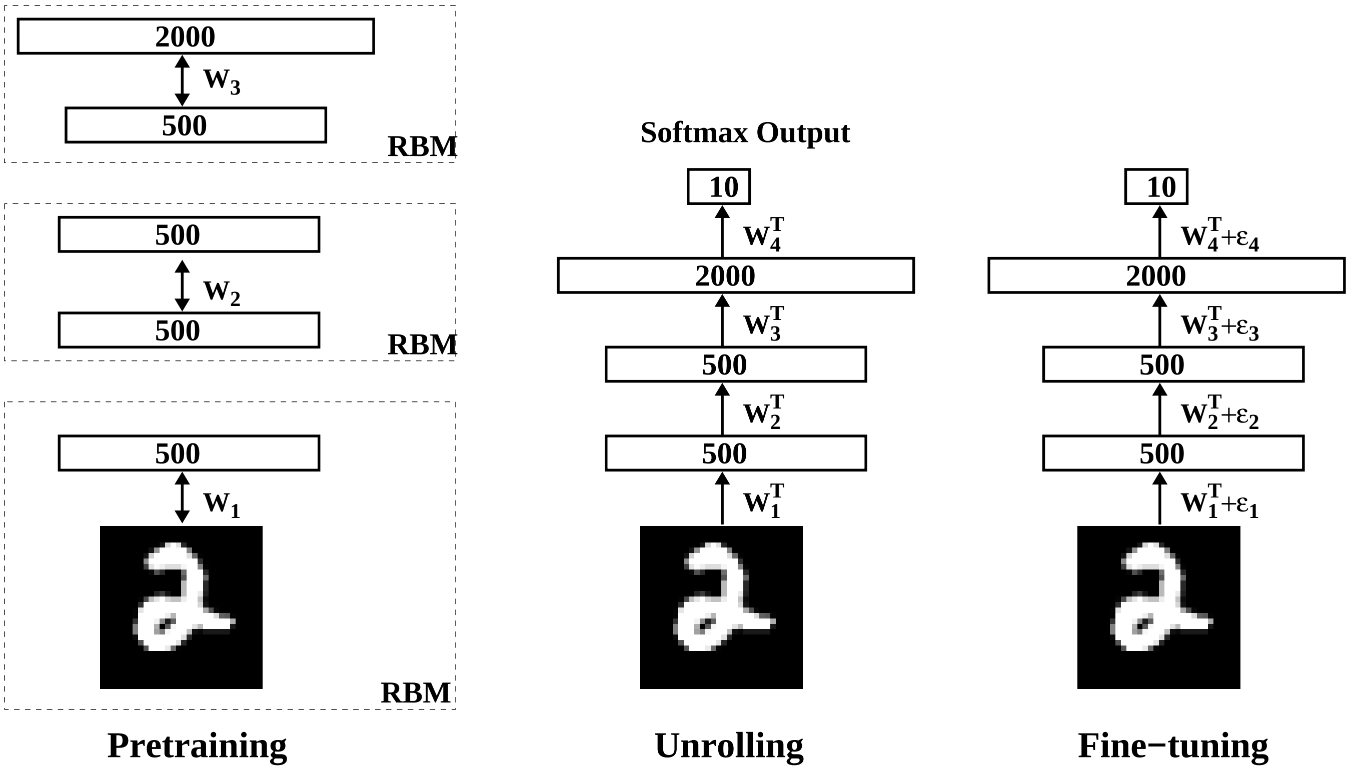
The Big Picture



Show Demo.
(See Hinton and Salakhutdinov, Science 2006)

Deep Belief Nets for Classification and Regression

Deep Belief Nets for Classification



Pretraining

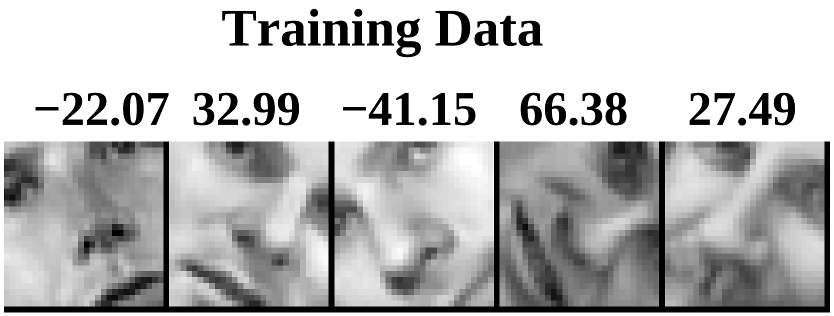
Unrolling

Fine−tuning

- After layer-by-layer pretraining of a 784-500-500-2000-10 network, discriminative fine-tuning achieves an error rate of 1.2% on MNIST. SVM's get 1.4% and randomly initialized backprop gets 1.6%.
- Clearly pretraining helps generalization. It ensures that most of the information in the weights comes from modeling the input data.
- The very limited information in the labels is used only to slightly adjust the final weights.

A Regression Task

- Predicting the orientation of a face patch.

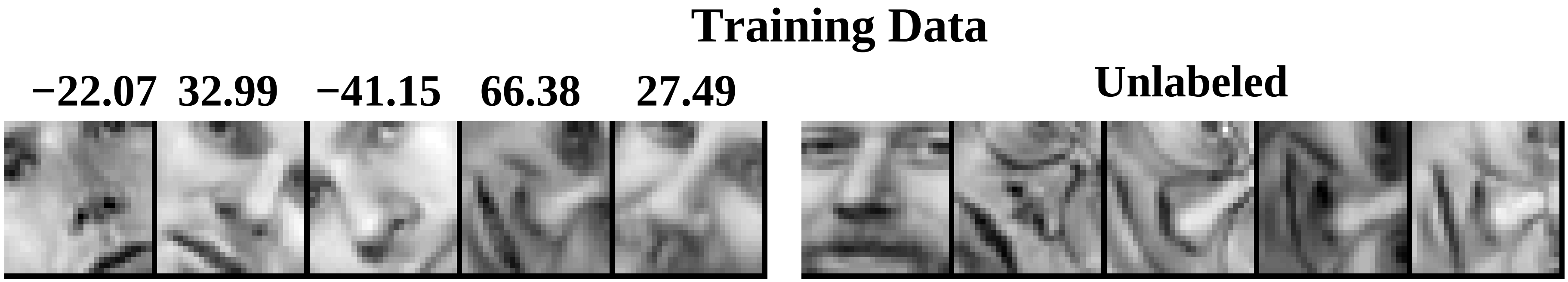


- Labeled Training Data:
  Input: 1000 labeled training patches from Olivetti faces of 30 training people.
  Output: orientation

- Labeled Test Data:
  Input: 1000 labeled test patches from Olivetti faces of 10 new people.
  Predict: orientation

- Gaussian Processes with spherical Gaussian kernel achieves a RMSE of 16.33°.

Deep Belief Nets for Regression



- Additional Unlabeled Training Data: 12000 face patches from 30 training people.

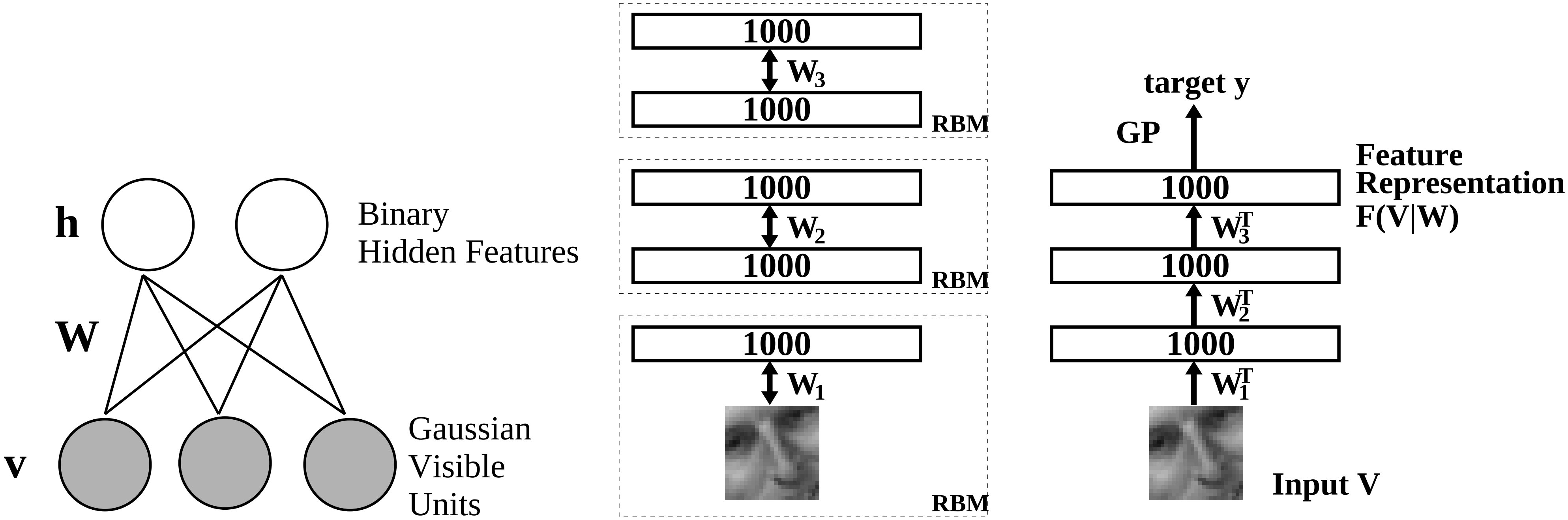
- Pretrain a stack of RBM's: 784-1000-1000-1000.

- **Features were extracted with no idea of the final task.**

The same GP on the top-level features: RMSE 11.22°.

GP with fine-tuned covariance Gaussian kernel: RMSE 6.42°.

Learning Covariance Kernel using Deep Belief Nets



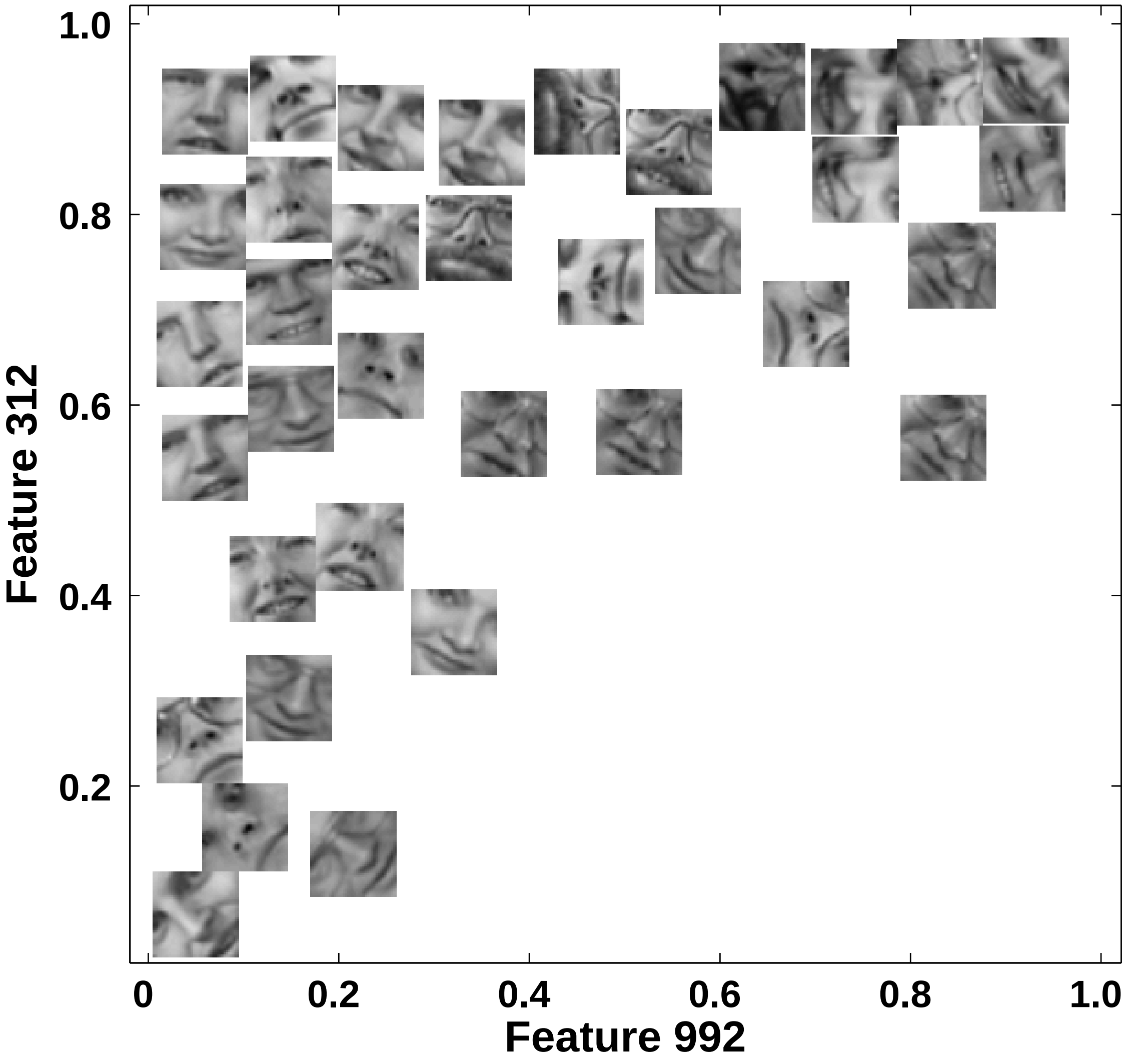
- Using DBN, we can initialize covariance function of the Gaussian Process parametrized by $\theta = \{\alpha, \beta\}$ and W as:

$$K_{nm} = \alpha \exp\Big(- \frac{1}{2\beta} ||F(\mathbf{v}^{\mathrm{n}}|\mathbf{W}) - \mathbf{F}(\mathbf{v}^{\mathrm{m}}|\mathbf{W})||^{\mathbf{2}}\Big)$$

And learn the parameters of this covariance function by maximizing the marginal likelihood.

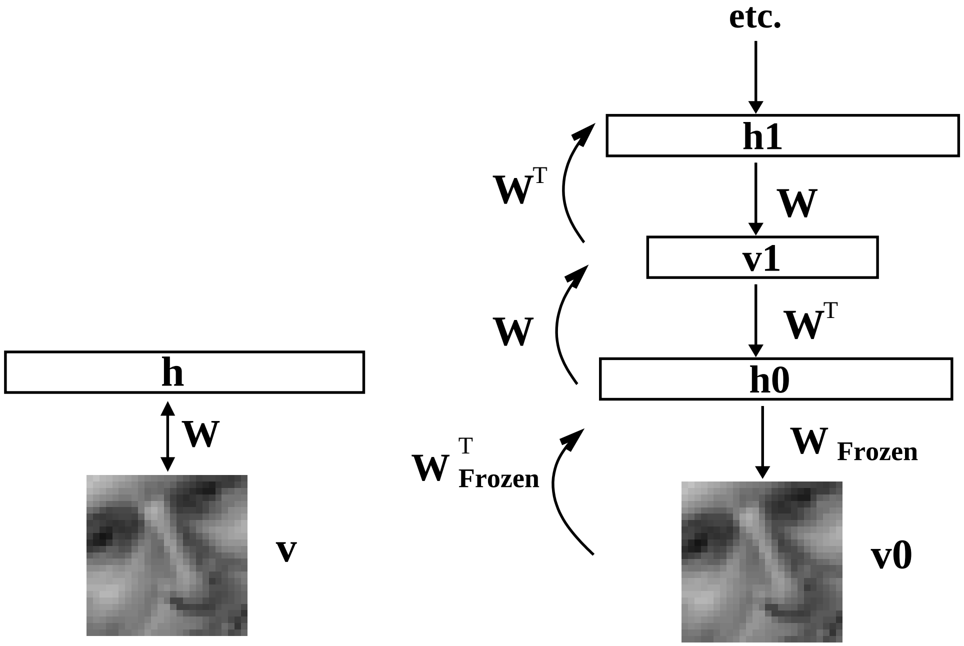
(See Salakhutdinov and Hinton, NIPS 2007)

Learning Covariance Kernel using Deep Belief Nets



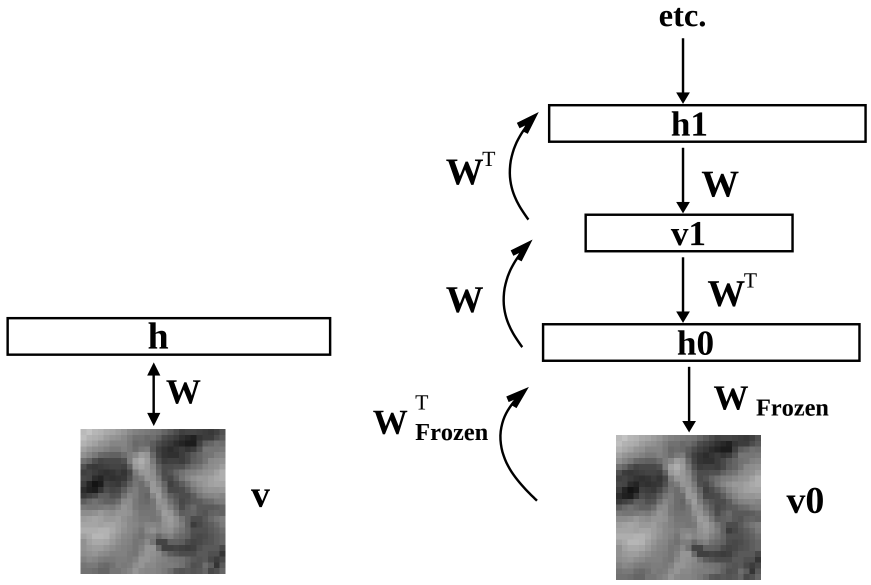
- A scatter plot of the two most relevant features, with each point replaced by the corresponding input test image.

The Generative View of Stacks of RBM's



- When $\mathbf{W}_{\mathbf{frozen}} = \mathbf{W}$, the two models are the same.
- The weights W_{frozen} define $p(\mathbf{v_0}|\mathbf{h_0}, \mathbf{W}_{\mathbf{frozen}})$ but also indirectly define $p(\mathbf{h_0})$.
- Idea: Freeze bottom layer of weights at $\mathbf{W}_{\mathbf{frozen}}$ and change higher layers to build a better model for $p(\mathbf{h_0})$, that is closer to the **posterior** hidden features produced by $\mathbf{W}_{\mathbf{frozen}}$ applied to the data $p(\mathbf{h_0}|\mathbf{v_0}, \mathbf{W}^{\mathbf{T}}_{\mathbf{frozen}})$.
- As we learn a new layer, the inference becomes incorrect, but the bound on the log probability of the data increases (see Hinton et.al.).

The Generative View of Stacks of RBM's



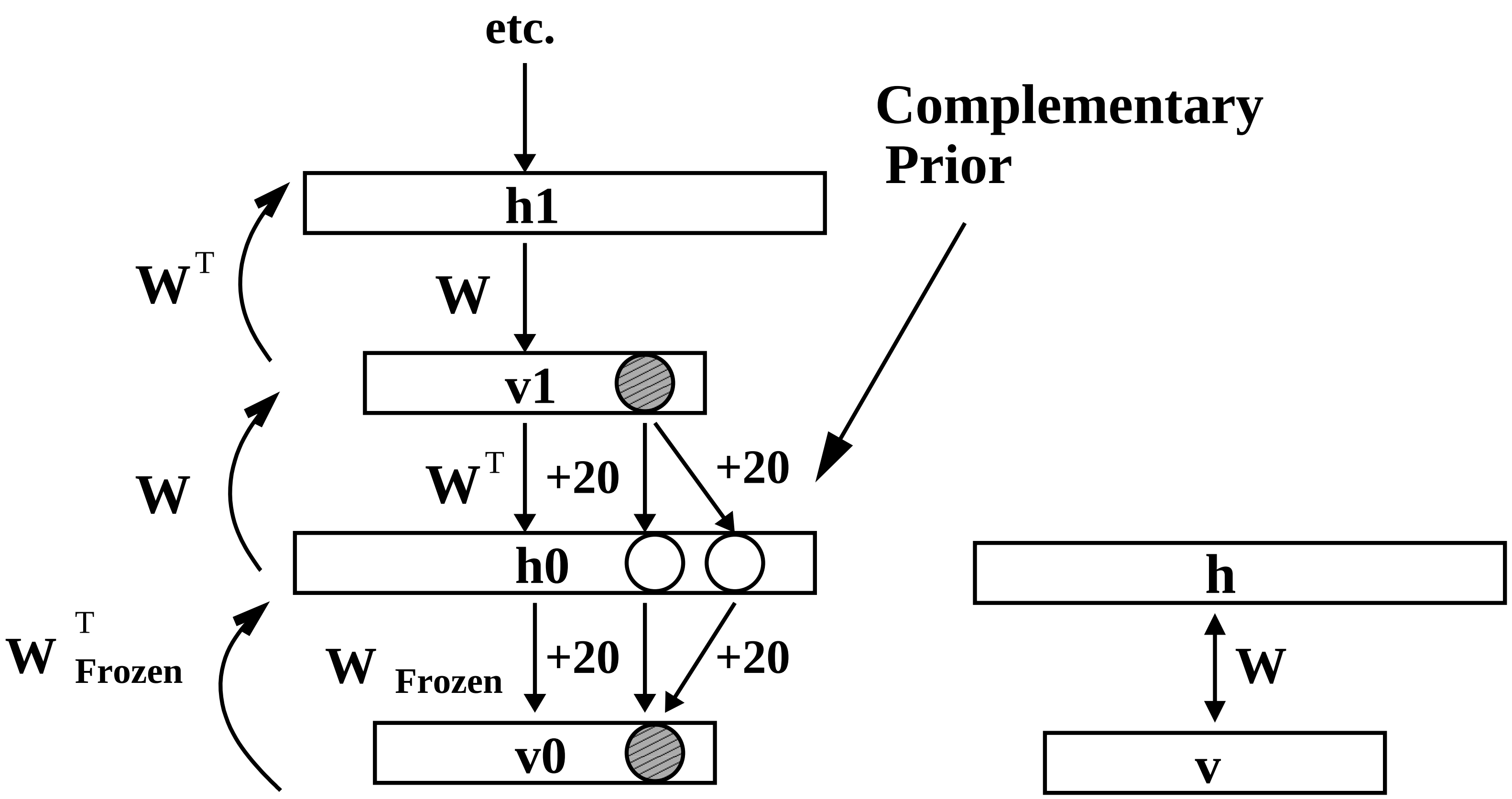
- For any approximating distribution $Q(\mathbf{h}|\mathbf{v_0}, \mathbf{W})$ we can write:

$$\log p(\mathbf{v_0}) \geq \sum_{\mathbf{h_0}} Q(\mathbf{h_0}|\mathbf{v_0}, \mathbf{W}) \Big[\log \mathbf{p}(\mathbf{h_0}|\mathbf{W}) + \log \mathbf{p}(\mathbf{v_0}|\mathbf{h_0}, \mathbf{W}) \Big]$$
$$- \sum_{\mathbf{h}} Q(\mathbf{h_0}|\mathbf{v_0}, \mathbf{W}) \log \mathbf{Q}(\mathbf{h_0}|\mathbf{v_0}, \mathbf{W})$$

- Initially $Q(\mathbf{h_0}|\mathbf{v_0}, \mathbf{W}) = \mathbf{p}(\mathbf{h_0}|\mathbf{v_0}, \mathbf{W})$.
- Freezing W at W_{frozen} will freeze $Q(\mathbf{h_0}|\mathbf{v_0}, \mathbf{W_{frozen}})$ and $p(\mathbf{v_0}|\mathbf{h_0}, \mathbf{W_{frozen}})$.
- Maximizing the above bound is equivalent to maximizing:

$$\sum_{\mathbf{h_0}} Q(\mathbf{h_0}|\mathbf{v_0}, \mathbf{W_{frozen}}) \log \mathbf{p}(\mathbf{h_0}|\mathbf{W})$$

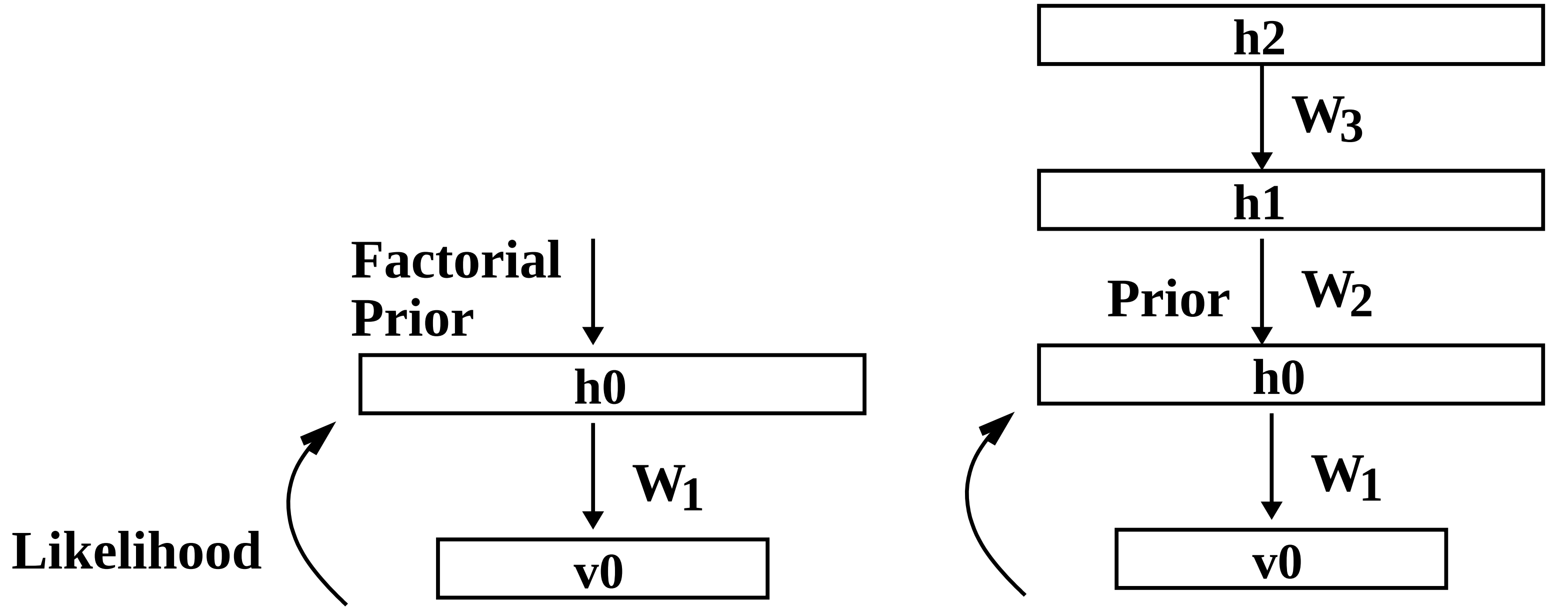
The Generative View of Stacks of RBM’s



$$\text{Likelihood} \times \text{Prior} = \sigma(W^T_{\text{Frozen}}\ v0)$$

- What about explaining away?
- A complementary prior exactly cancels out correlations created by explaining away! So the posterior factors.

Two Alternatives to Our Method



- Alternative 1:
  - Without complementary prior, learning one layer at a time is hard because of explaining away.
- Alternative 2:
  - If we start with different weights in each layer and try to learn them all at once, we have major problems.
  - Just to calculate the prior for h_0 requires integration over all higher-level hidden configurations!

The Generative View of Stacks of RBM's

- Demo of Digits and Faces

Semantic Hashing for Fast Information Retrieval

Existing Methods

- One of the most popular and widely used in practice algorithms for document retrieval tasks is TF-IDF. However:
  - It computes document similarity directly in the word-count space, which can be slow for large vocabularies.
  - It assumes that the counts of different words provide independent evidence of similarity.
  - It makes no use of semantic similarities between words.

- To overcome these drawbacks, models for capturing low-dimensional, latent representations have been proposed and successfully applied in the domain of information retrieval.

- One such simple and widely-used method is Latent Semantic Analysis (LSA), which extracts low-dimensional semantic structure using SVD to get a low-rank approximation of the word-document co-occurrence matrix.

Drawbacks of Existing Methods

- LSA is a linear method so it can only capture pairwise correlations between words. We need something more powerful.

- Numerous methods, in particular probabilistic versions of LSA were introduced in the machine learning community.

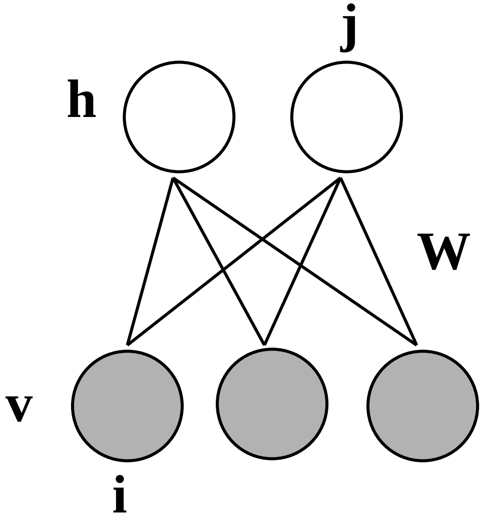
- These models can be viewed as graphical models in which a single layer of hidden topic variables have directed connections to variables that represent word-counts.

- There are limitations on the types of structure that can be represented efficiently by a single layer of hidden variables.

- We will build a network with multiple hidden layers and with millions of parameters and show that it can discover latent representations that work much better.

RBM's for count data

- Hidden units remain binary and the visible word counts are modeled by the Constrained Poisson Model.



- The energy is defined as:

$$E(\mathbf{v},\mathbf{h}) = -\sum_i b_i v_i - \sum_j b_j h_j - \sum_{i,j} v_i h_j W_{ij} + \sum_i \log v!$$

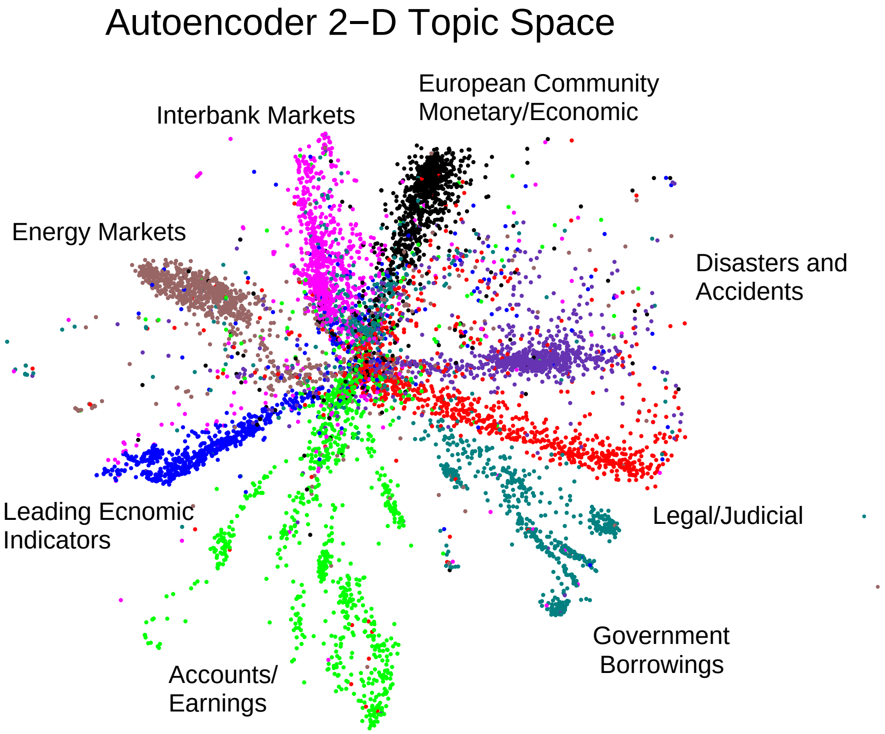
- Conditional distributions over hidden and visible units are:

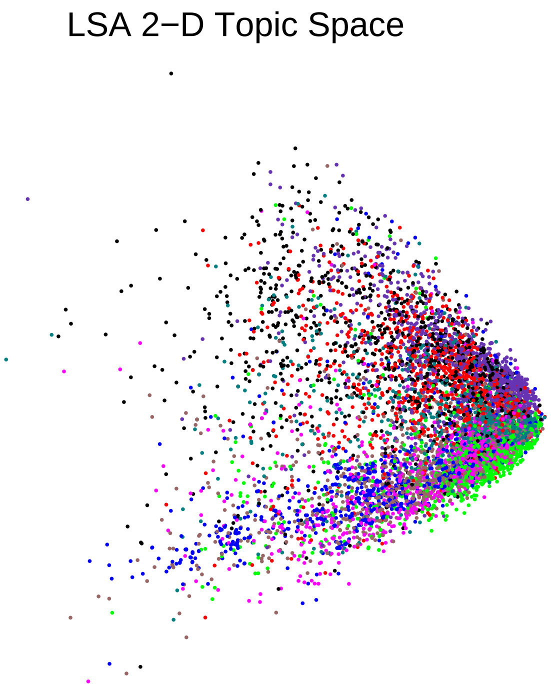
$$p(h_j = 1|\mathbf{v}) = \frac{1}{1+\exp(-b_j - \sum_i W_{ij} v_i)}$$

$$p(v_i = n|\mathbf{h}) = \text{Poisson}\left(\frac{\exp\left(\lambda_i + \sum_j h_j W_{ij}\right)}{\sum_k \exp\left(\lambda_k + \sum_j h_j W_{kj}\right)} N\right)$$

- where N is the total length of the document.

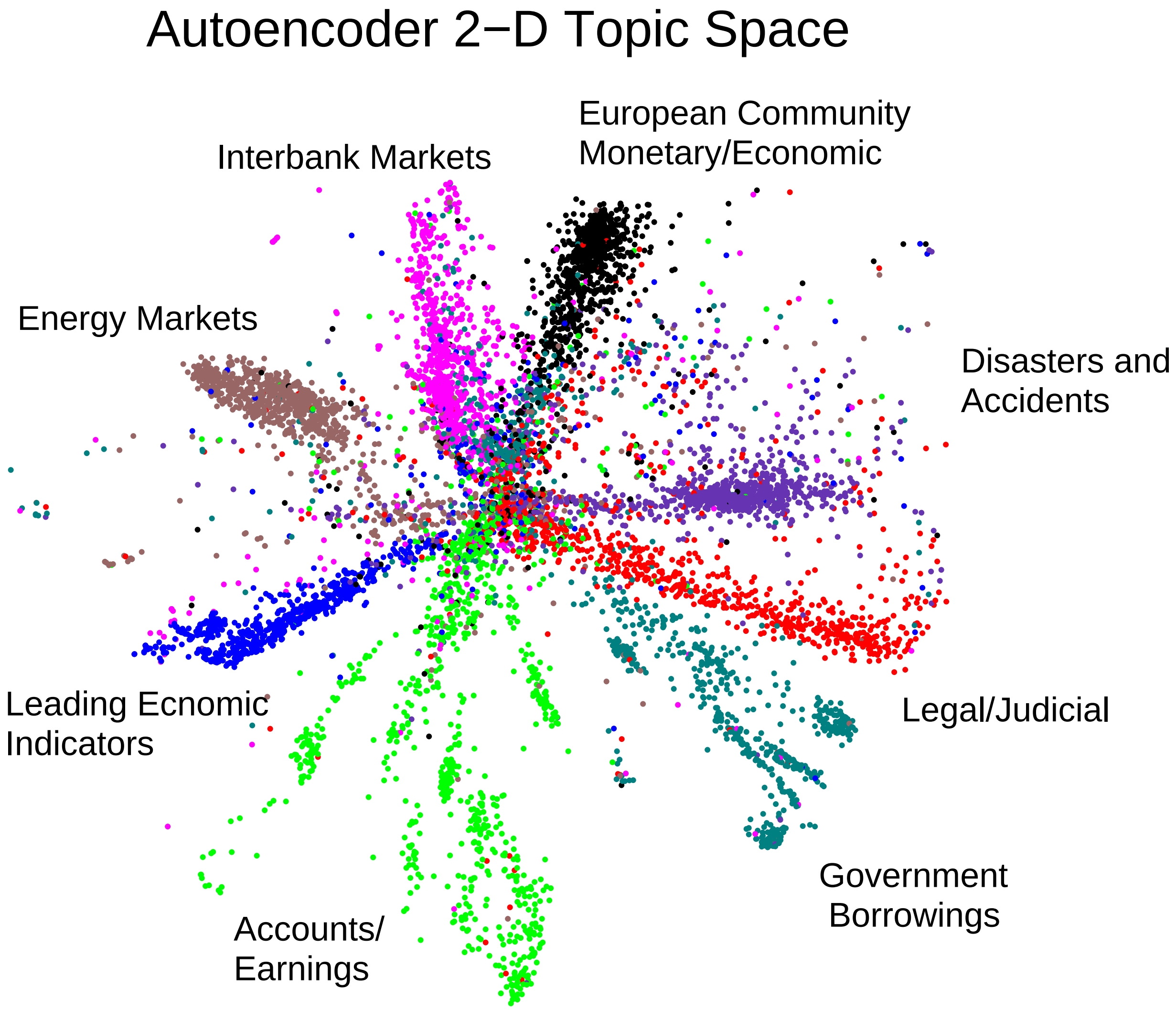
Reuters Corpus: Learning 2-D code space


Autoencoder 2−D Topic Space


LSA 2−D Topic Space

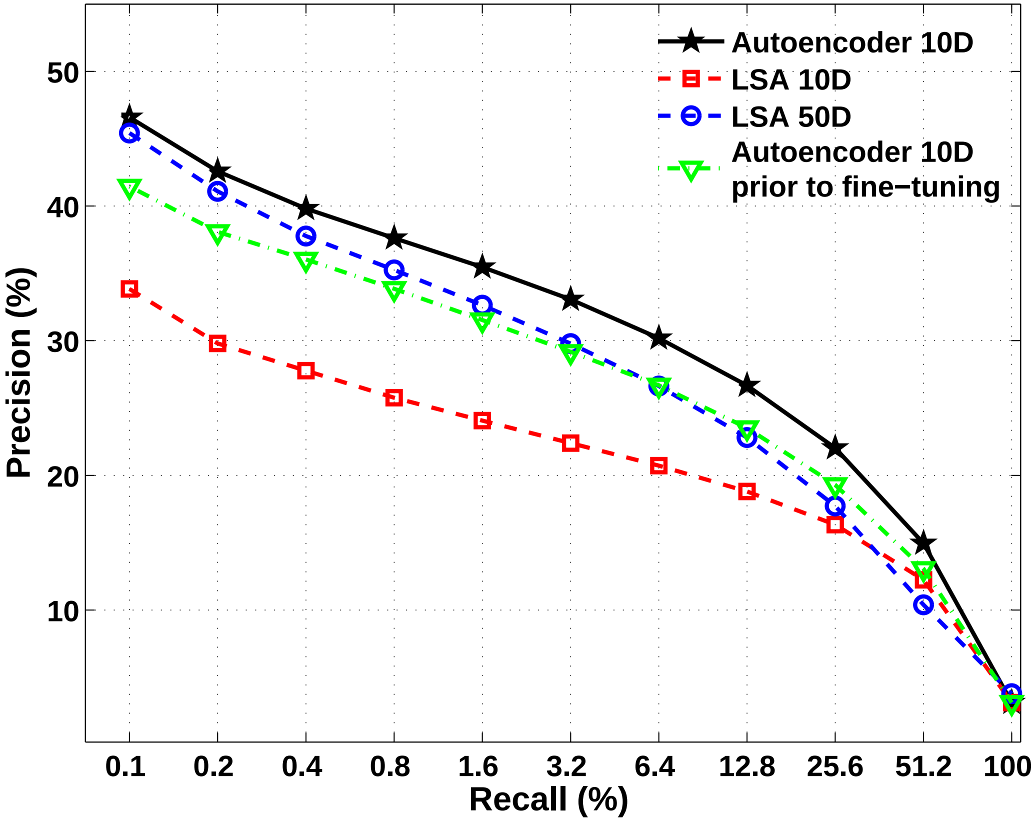
- We use a 2000-500-250-125-2 autoencoder to convert test documents into a two-dimensional code.
- The Reuters Corpus Volume II contains 804,414 newswire stories (randomly split into **402,207** training and **402,207** test).
- We used a simple “bag-of-words” representation. Each article is represented as a vector containing the counts of the most frequently used 2000 words in the training dataset.

Reuters Corpus: Learning 2-D code space

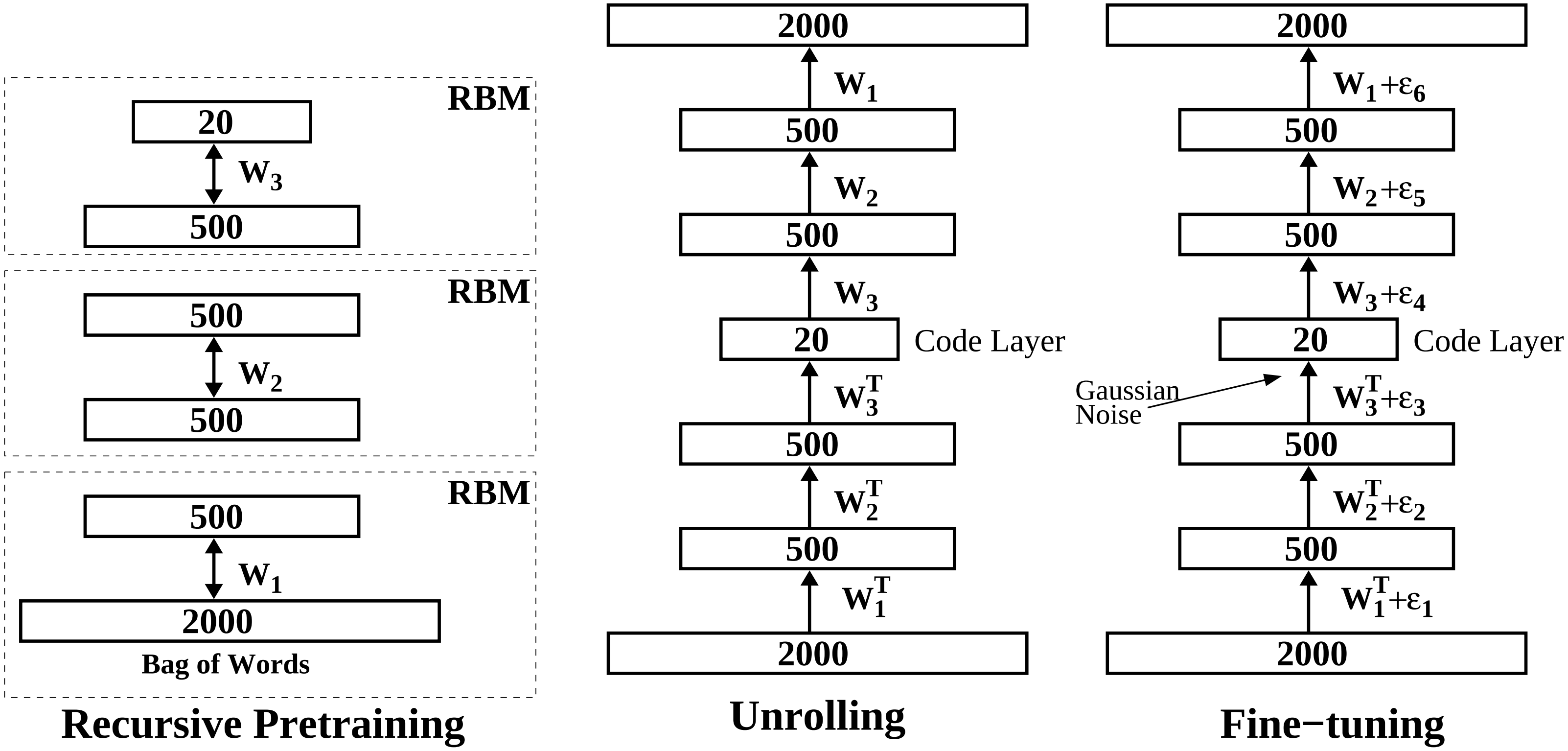


Results for 10-D codes

- We use the cosine of the angle between two codes as a measure of similarity.
- Precision-recall curves when a 10-D query document from the test set is used to retrieve other test set documents, averaged over 402,207 possible queries.

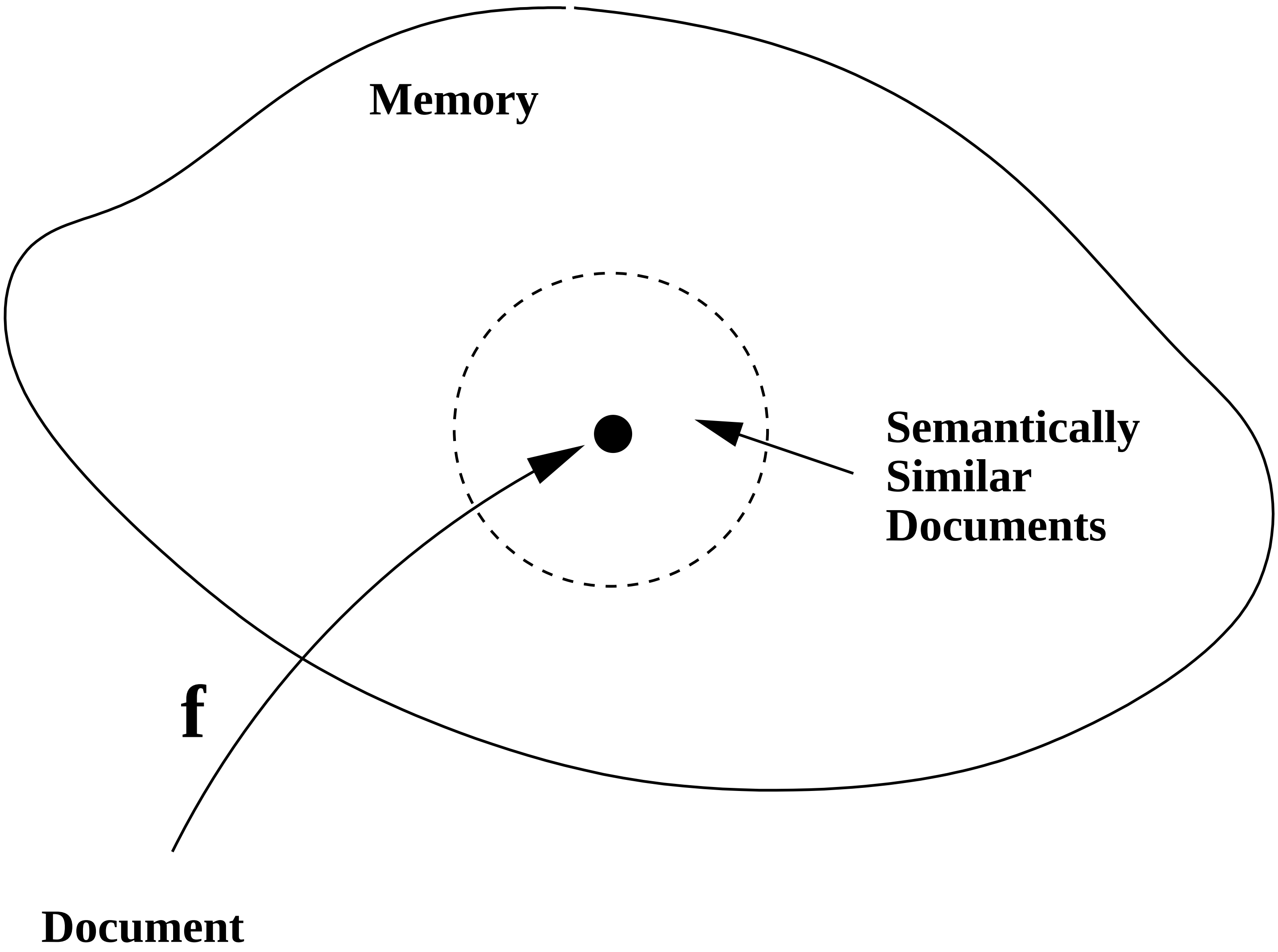


Semantic Hashing



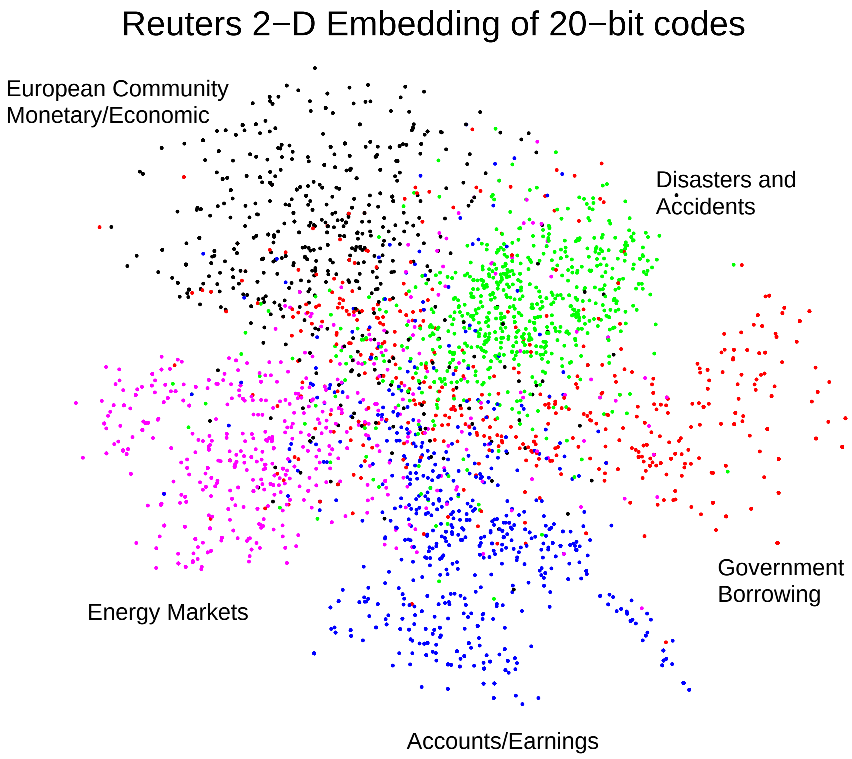
- Learn to map documents into *semantic* 20-D binary code and use these codes as memory addresses.
- We have the ultimate retrieval tool: Given a query document, compute its 20-bit address and retrieve all of the documents stored at the similar addresses **with no search at all**.

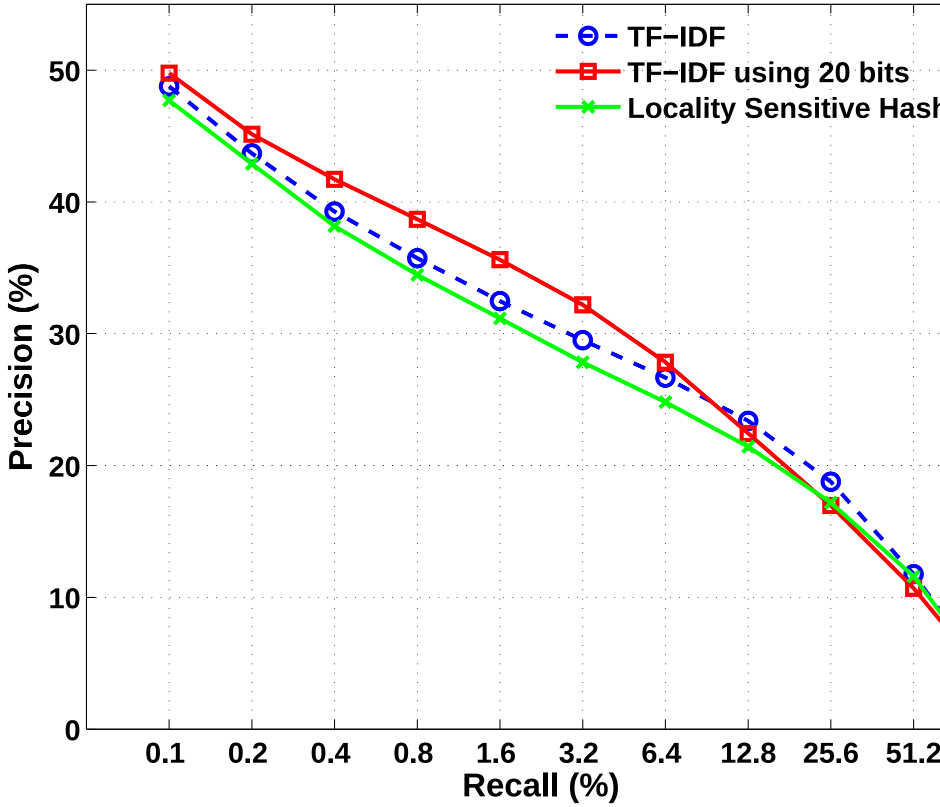
The Main Idea of Semantic Hashing



(See Salakhutdinov and Hinton, SIGIR 2007 workshop on Graphical Models)

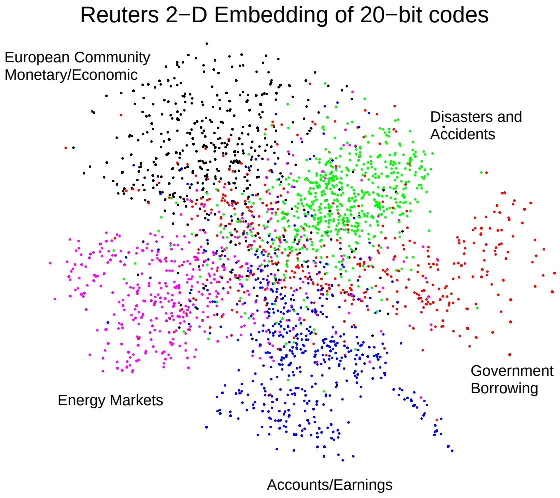
Semantic Hashing

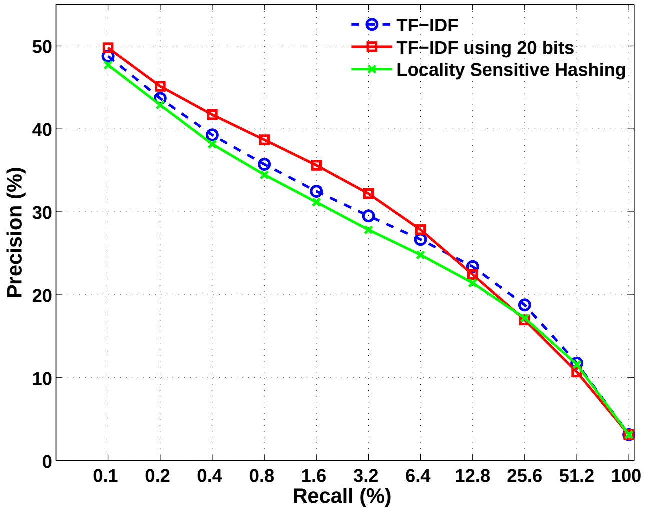




- Left picture shows a 2-dimensional embedding of the learned 20-bit codes using stochastic neighbor embedding.
- Right picture shows Precision-Recall curves when a query document from the test set is used to retrieve other test set documents, averaged over all 402,207 possible queries.

Semantic Hashing





- We used a simple C implementation on Reuters dataset (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.
- It then takes 10 milliseconds to retrieve the top few matches from that short-list using TF-IDF, and it is more accurate than full TF-IDF.
- Scaling up the learning to billion documents would not be too difficult, since learning is linear in the size of the dataset.

Deep Belief Nets → Good Generative Models?

- Remember, the idea behind learning DBN's as a stack of RBM's is that you are always guaranteed to improve the lower bound on the log-probability of the data.

- We would hope that this greedy recursive procedure is capable of learning a good generative model of high-dimensional highly-structured data.

- But are DBN's really better in modeling data than simple mixture models. I was avoiding this answer because I don't know. Due to the presence of the partition function we cannot calculate the log-probability of the test data even for a simple RBM model.

- Instead I was saying that the top-level features that the DBN model learns are good for various discriminative tasks such as classification, regression, or compression.

- This, however, does not justify the claim that DBN's are good generative models of data.

Deep Belief Nets → Good Generative Models?

- We have recently discovered an effective way of estimating the ratio of partition functions of two RBM's using Annealed Importance Sampling (AIS) procedure introduced by Radford Neal (1998). This allows us to compare various RBM's.

- The same procedure can also be used to get unbiased estimate of the partition function for a single RBM model. This allows us to compare RBM's to other generative models.

- Moreover, using variational inference, we can estimate the lower bound on the log-probability that the DBN model assigns to the test data.

- This allows us to get some quantitative judgement of whether DBN's are better when comparing to other generative models.

- Of course when using AIS in practice, we are relying on empirical estimates of its accuracy, which can sometimes be misleading.

THE END