Deep Belief Networks

Ruslan Salakhutdinov and Geoffrey Hinton

University of Toronto, Machine Learning Group

The Snowbird Workshop

March 22, 2007

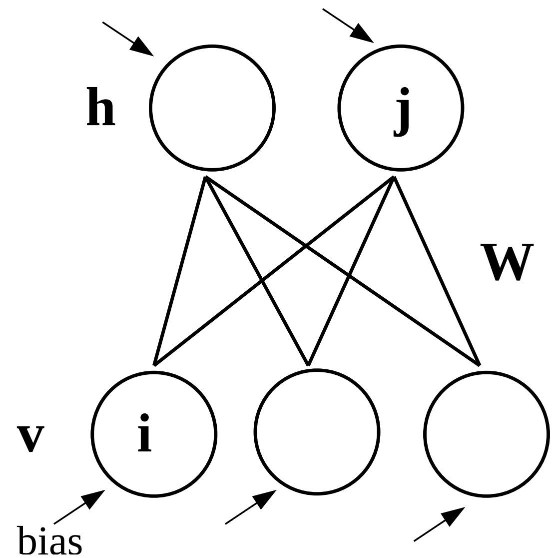
Talk outline

- 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.
  - A Generative Model of Simple Shapes.
- Another Application of Deep Belief Nets (if time permits).
  - Semantic Hashing for Ultra Fast Document Retrieval.

Restricted Boltzmann Machines

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

- RBM is a two-layer network in which visible, binary stochastic pixels $\mathbf{v}$ are connected to hidden binary stochastic feature detectors $\mathbf{h}$.



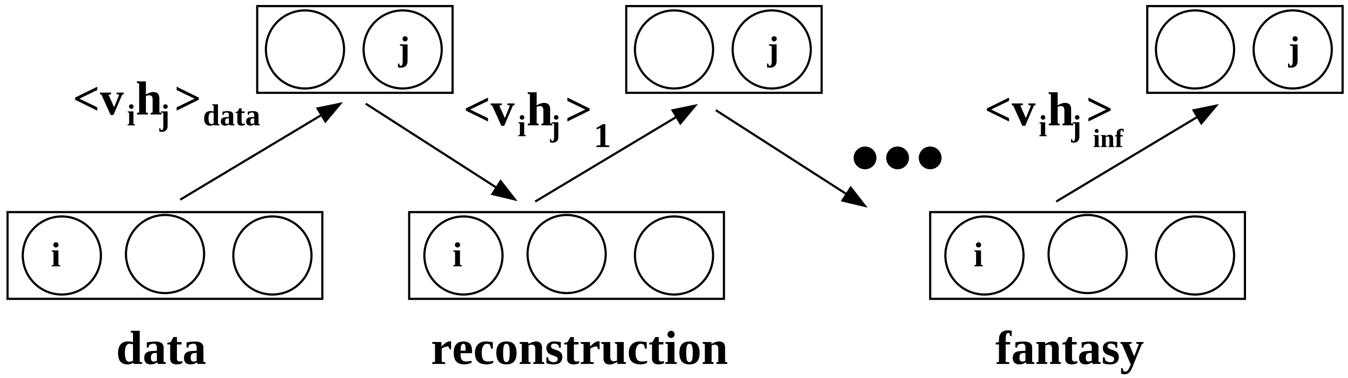
- A joint configuration $(\mathbf{v}, \mathbf{h})$ has an energy:

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

- The probability that the model assigns to $\mathbf{v}$ 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 and Learning



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

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

- Maximum Likelihood learning:

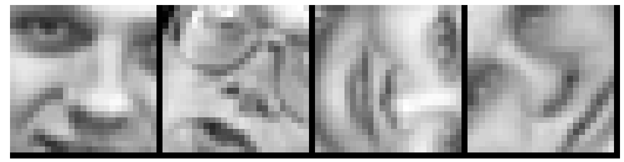
$$\Delta W_{ij} = \epsilon(< v_i h_j >_{data} - < v_i h_j >_{\infty})$$

- Contrastive Divergence (1-step) learning:

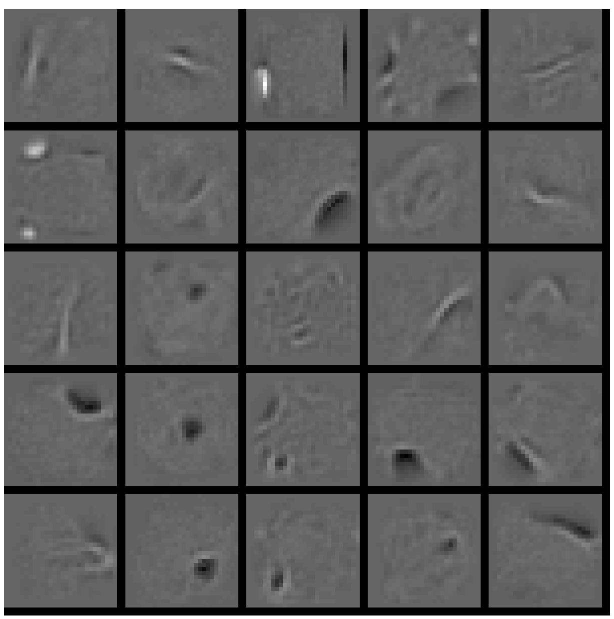
$$\Delta W_{ij} = \epsilon(< v_i h_j >_{data} - < v_i h_j >_1)$$

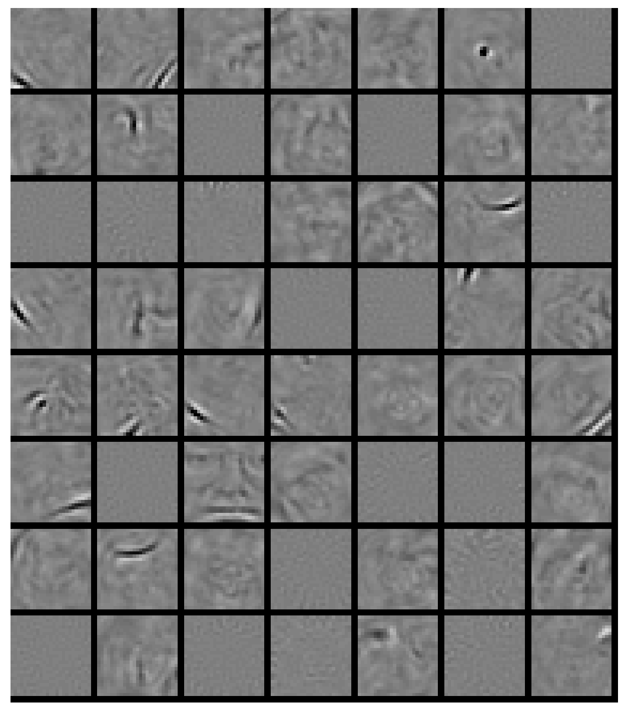
What a single RBM learns

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



- Learned W

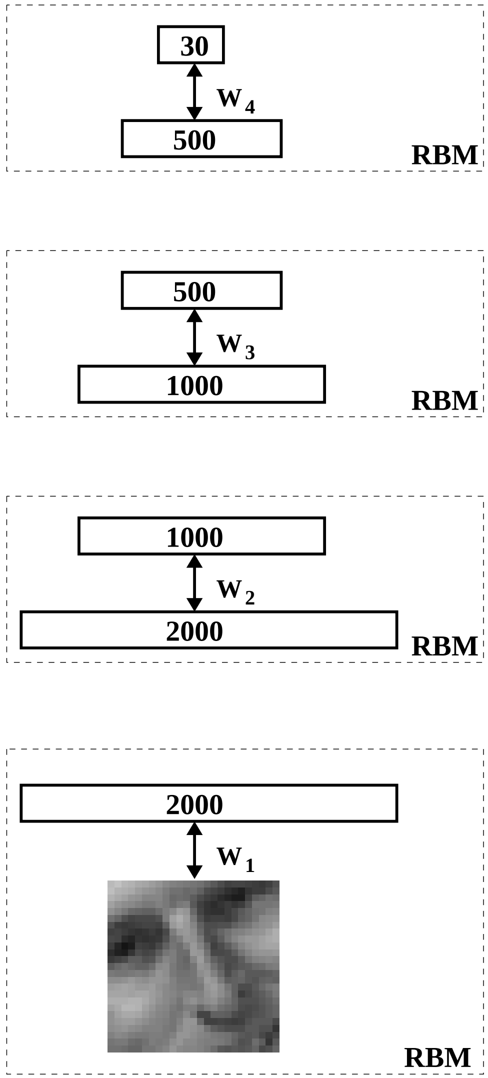




Learning Stacks of RBM's

- 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.

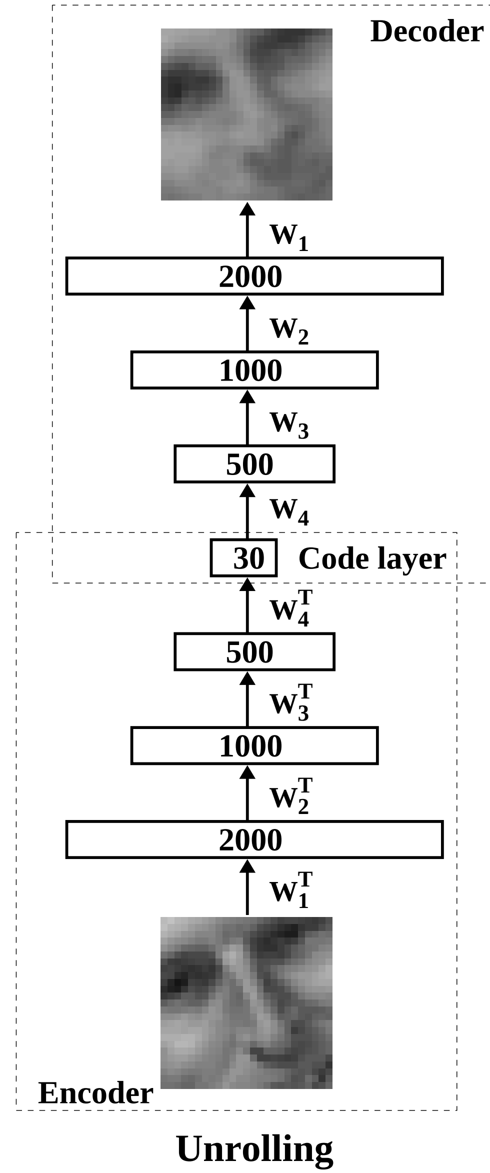


- Under certain conditions adding an extra layer always improves a lower bound on the log probability of data (explained later).

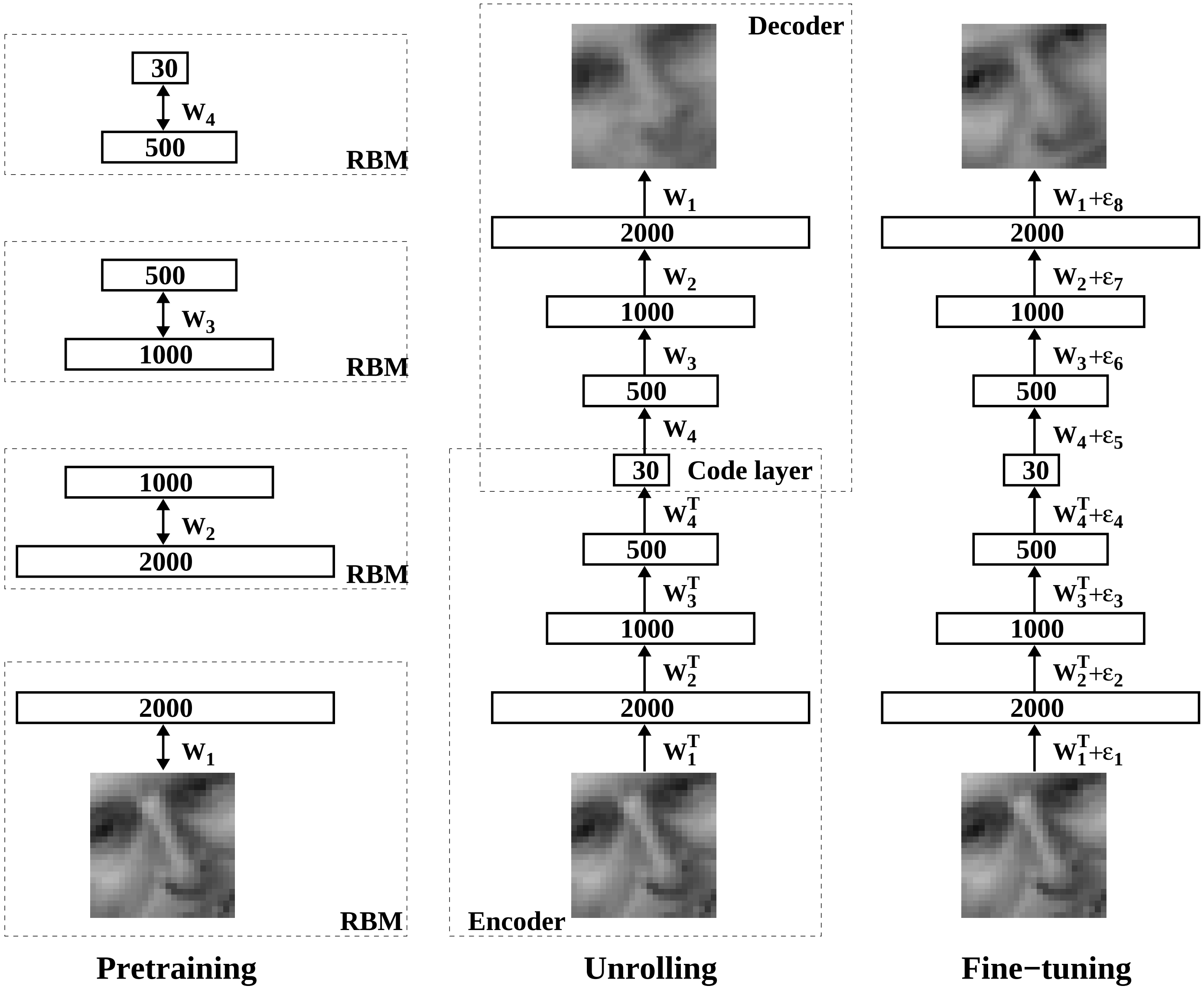
- Each layer of features captures strong high-order correlations between the activities of units in the layer below.

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.

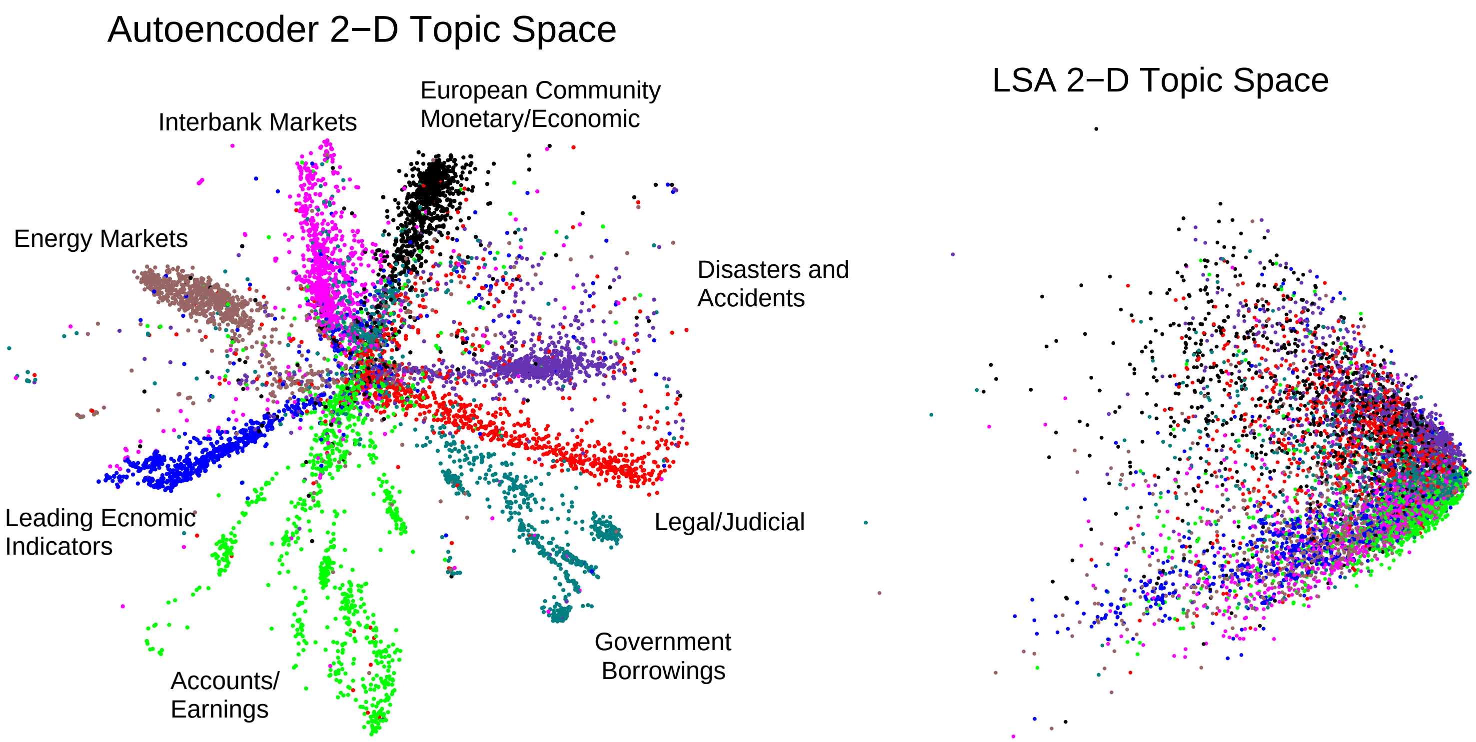


The Big Picture



Show Demo.

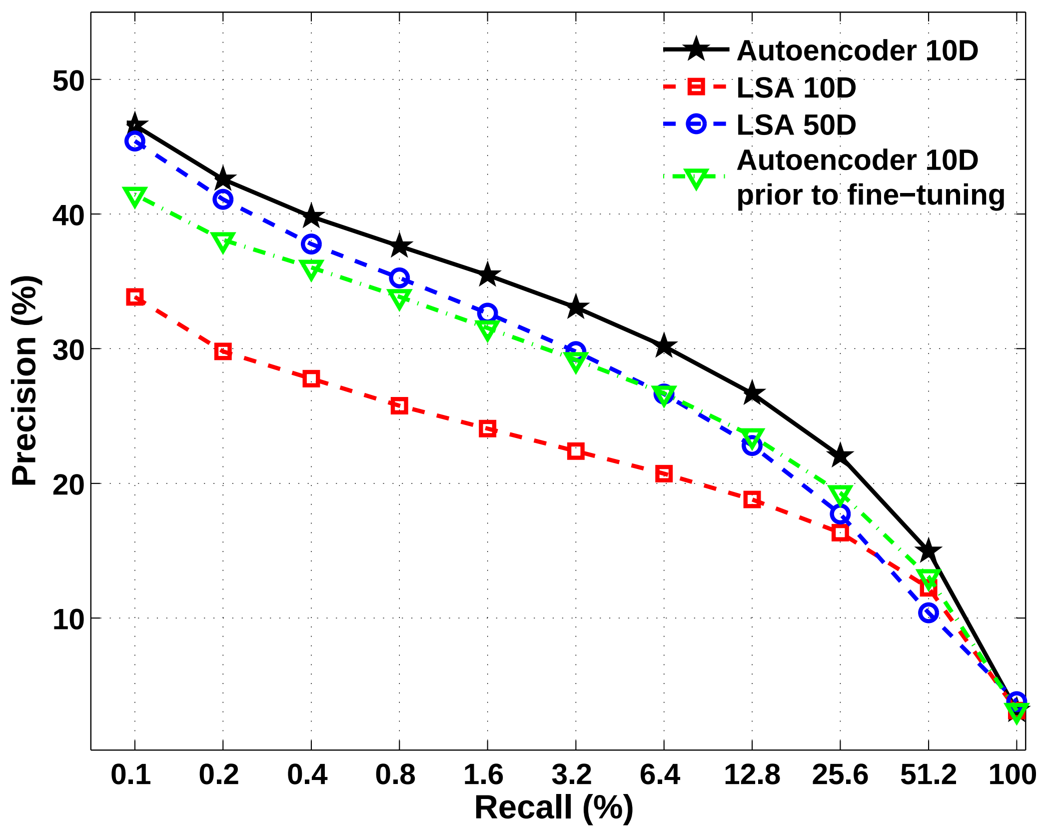
Reuters Corpus: Learning 2-D code space



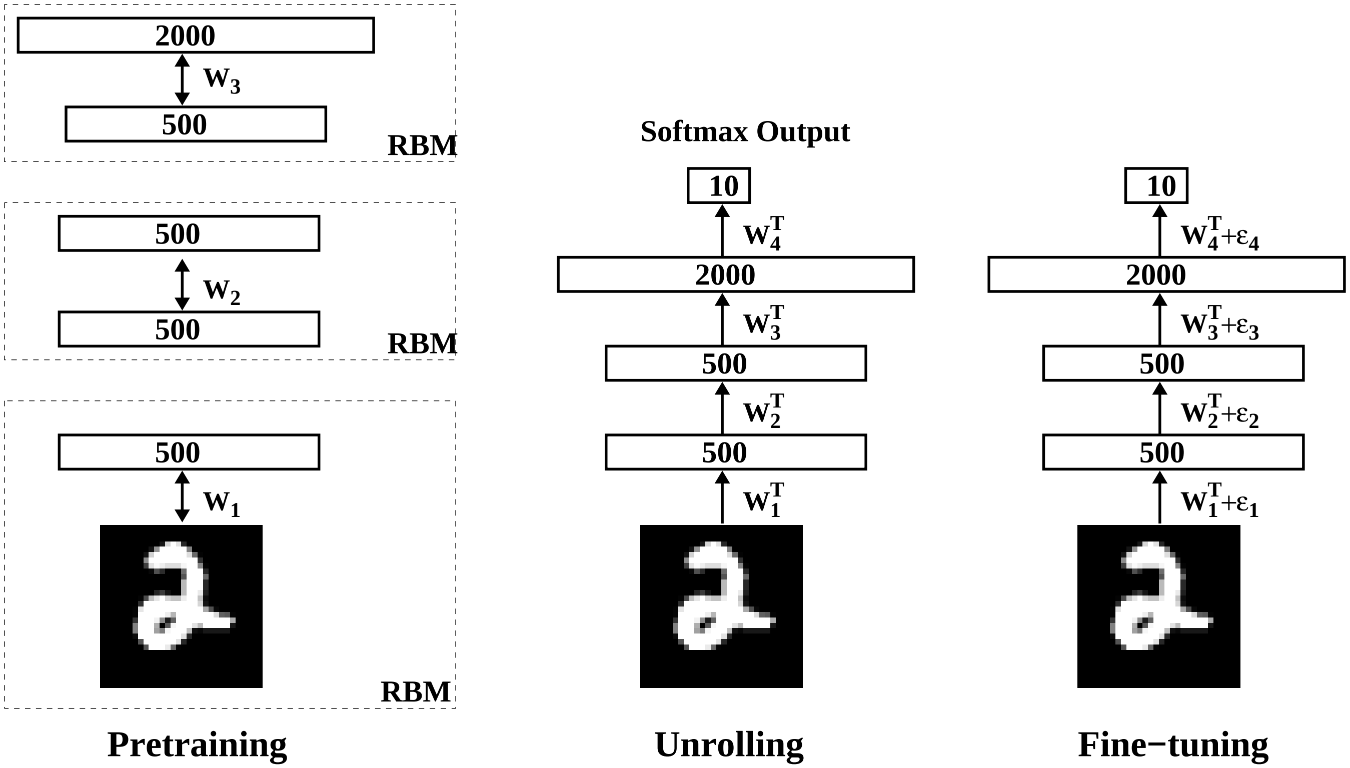
- We use a 2000-500-250-125-2 autoencoder to convert test documents into a two-dimensional code.
- The Reuters Corpus Volume II contains 804,414 newswire stories (randomly split into **402,207** training and **402,207** test).
- We used a simple "bag-of-words" representation. Each article is represented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

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.



Deep Belief Nets for Classification

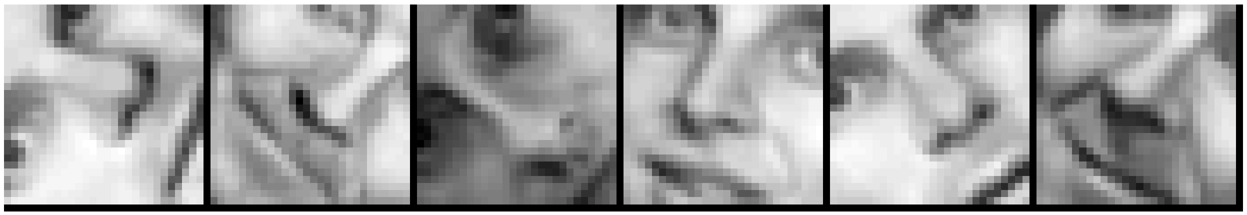


- After layer-by-layer pretraining of a 784-500-500-2000-10 network, discriminative fine-tuning achieves an error rate of 1.2% on MNIST. SVM's get 1.4% and randomly initialized backprop gets 1.6%.
- Clearly pretraining helps generalization. It ensures that most of the information in the weights comes 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.

-66.84 43.48 −57.14 14.22 −35.75 30.01



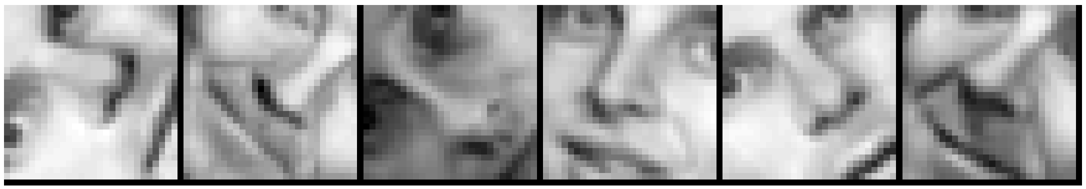
- Labeled Training Data:
  Input: 1000 labeled training patches from Olivetti faces of 30 training people.
  Output: orientation

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

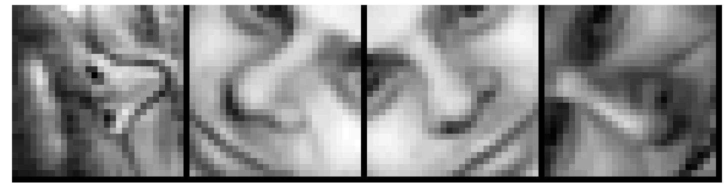
- Gaussian Processes with Gaussian kernel (using Radford Neal's software) achieves a RMSE of 16.35° ($\pm 0.45^\circ$).

Deep Belief Nets for Regression

-66.84 43.48 −57.14 14.22 −35.75 30.01



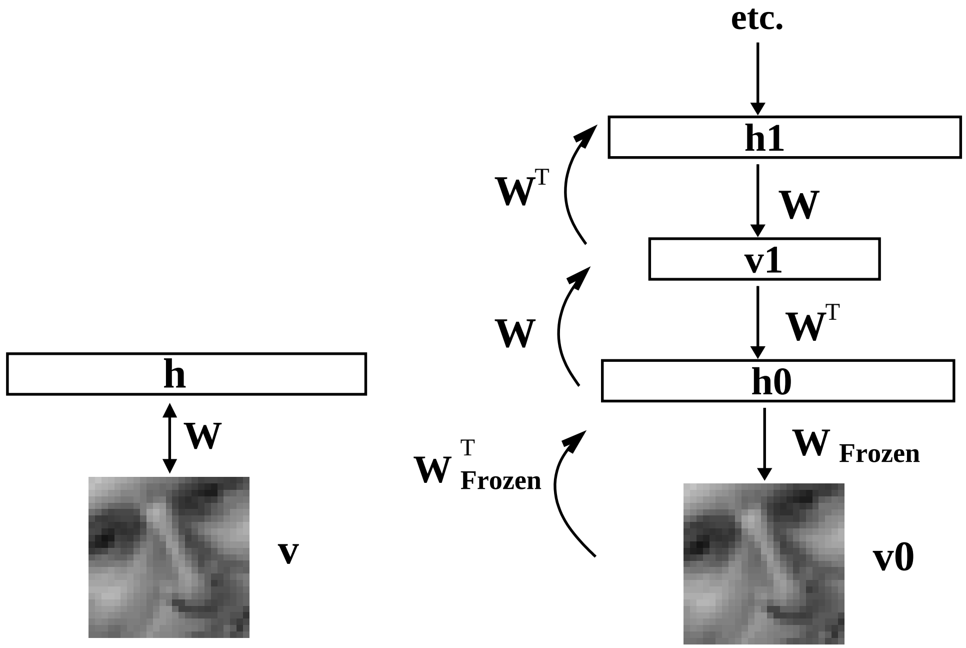
Unlabeled



- Additional Unlabeled Training Data: 12000 face patches from 30 training people.
- Pretrain a stack of RBM's: 784-1000-500.
- **Features were extracted with no idea of the final task.**

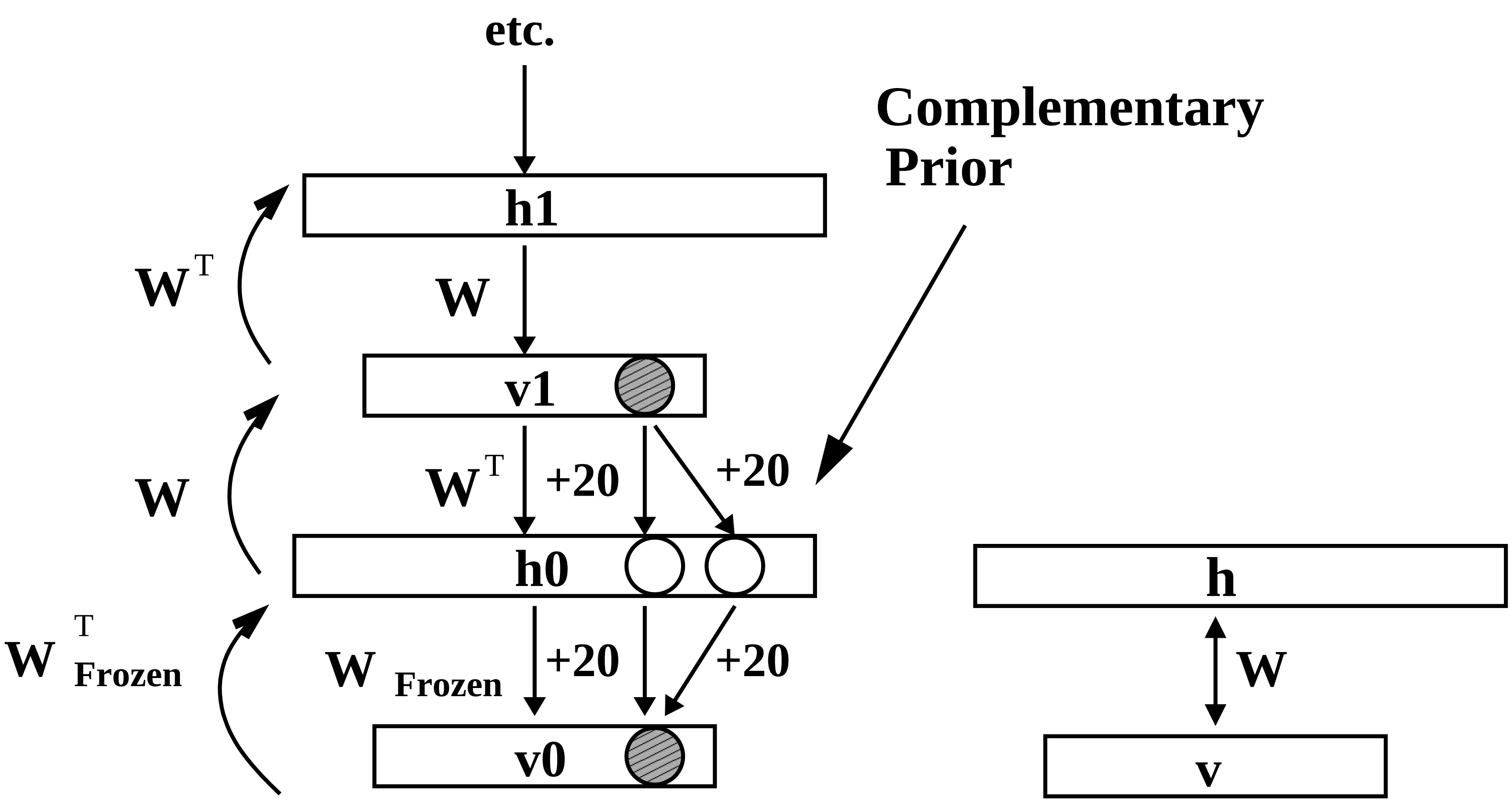
Train a dumb linear regression model on the top-level features using the labeled 1000 training cases:	RMSE 13.73°.
The same GP on the top-level features:	RMSE 10.06° ($\pm 0.36^\circ$).

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.).

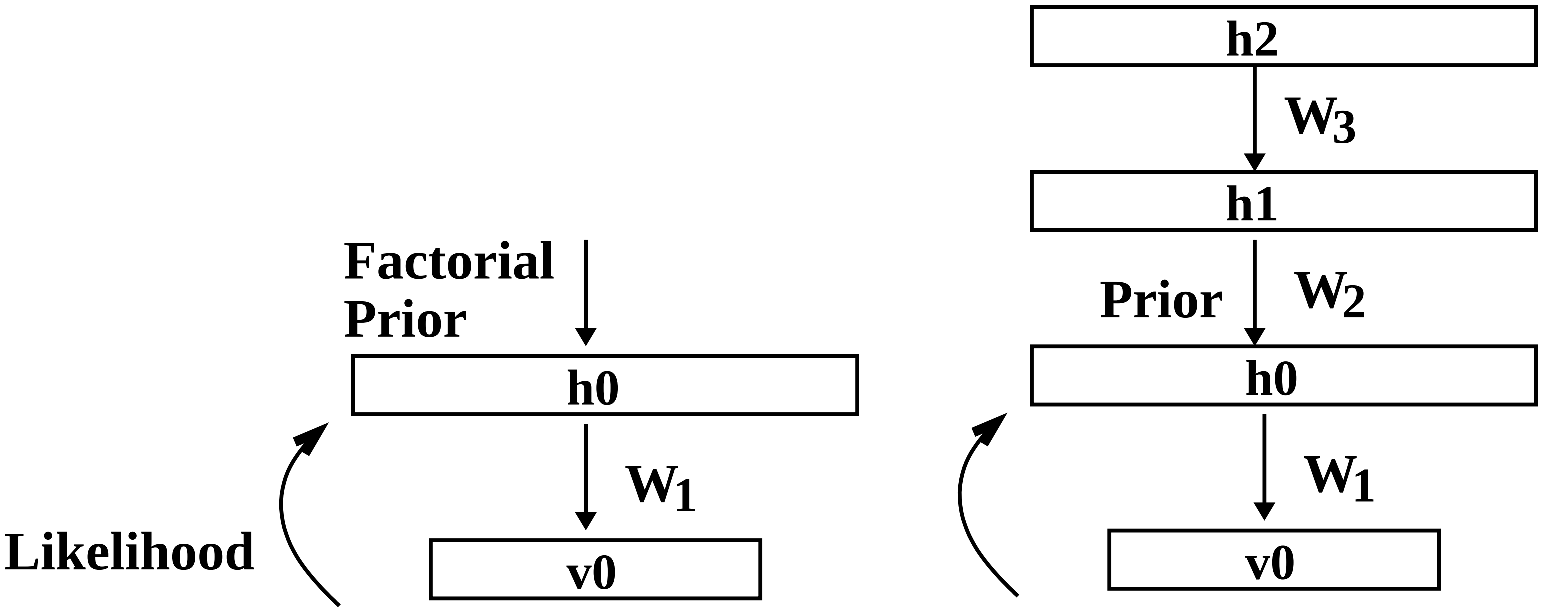
The Generative View of Stacks of RBM's



Likelihood x Prior = $\sigma(W^{T}_{Frozen}\ v0)$

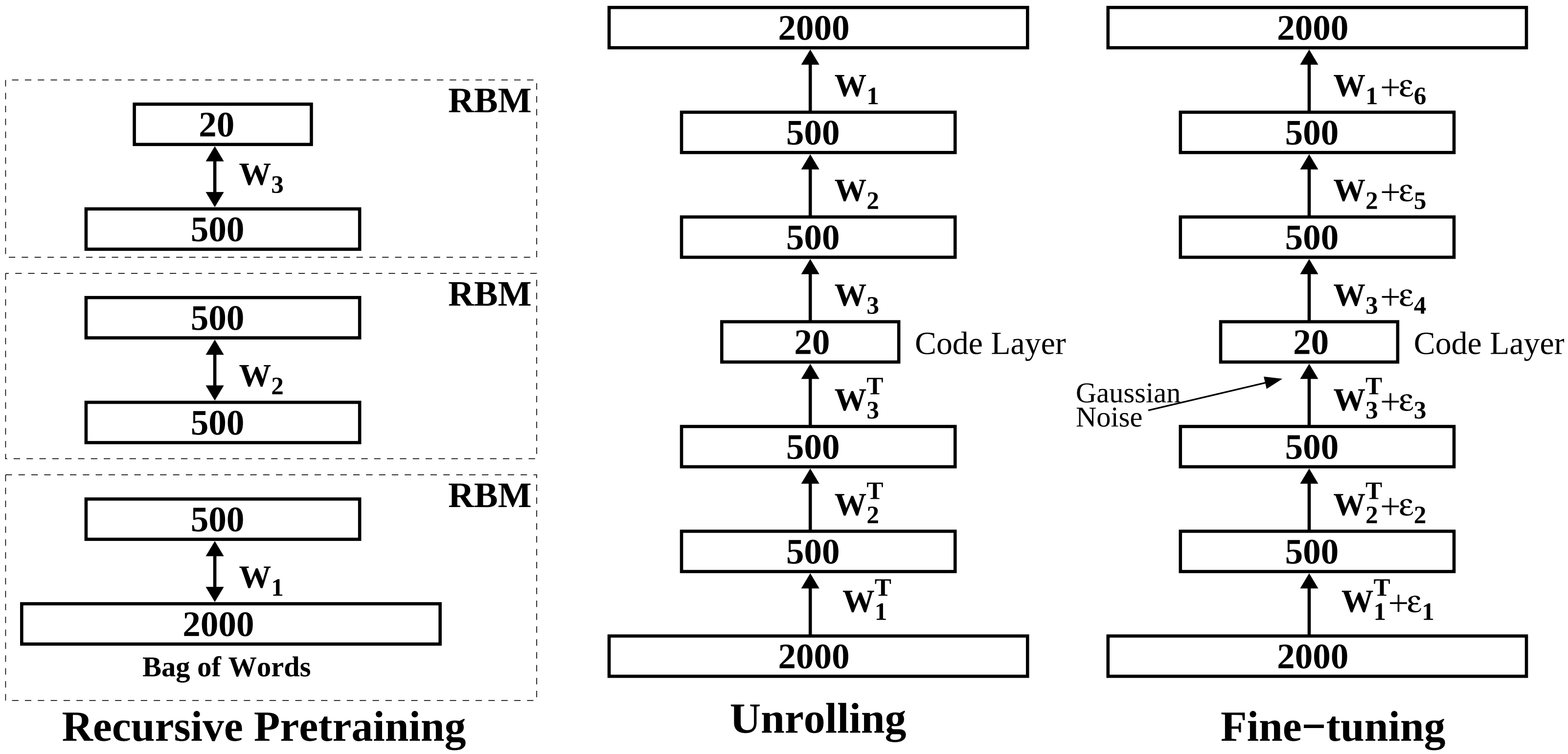
- What about explaining away?
- A complementary prior exactly cancels out correlations created by explaining away! So the posterior factors.

Two Alternatives to Our Method



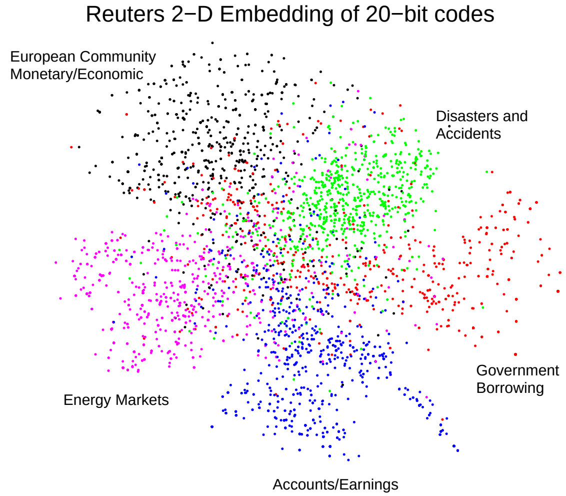
- Alternative 1:
  - Without complementary prior, learning one layer at a time is hard because of explaining away.
- Alternative 2:
  - If we start with different weights in each layer and try to learn them all at once, we have major problems.
  - Just to calculate the prior for h_0 requires integration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

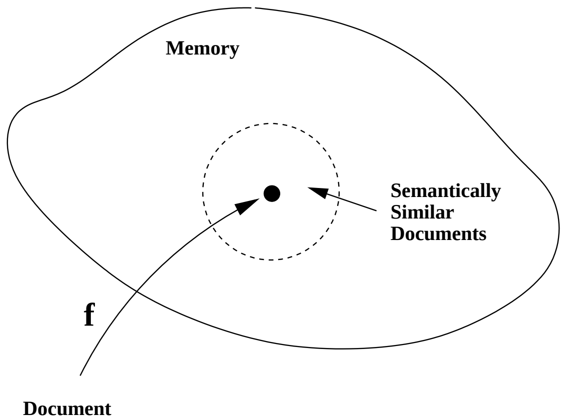
Semantic Hashing

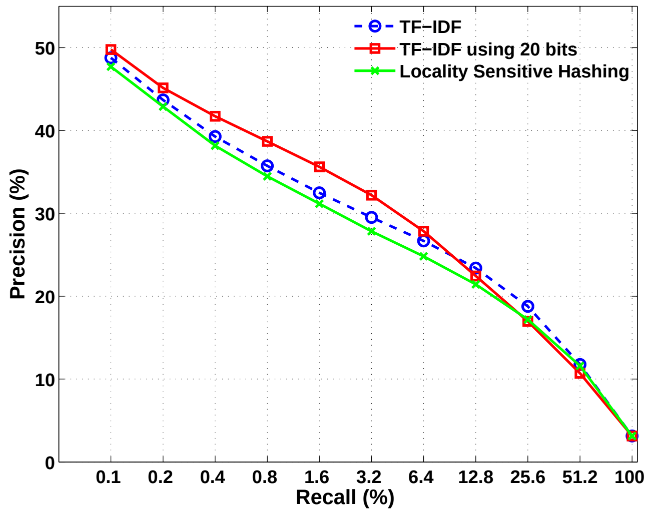


- Learn to map documents into *semantic* 20-D binary code and use these codes as memory addresses.
- We have the ultimate retrieval tool: Given a query document, compute its 20-bit address and retrieve all of the documents stored at the similar addresses **with no search at all**.

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.
- Locality-Sensitive Hashing takes about 500 milliseconds, and is less accurate.
- Our method is 50 times faster than the fastest existing method and is more accurate.

THE END