Learning Deep Hierarchies of Representations

Yoshua Bengio, U. Montreal

Google Research, Mountain View, California

September 23rd, 2009

Thanks to: Aaron Courville, Pascal Vincent, Dumitru Erhan, Olivier Delalleau, Olivier Breuleux, Yann LeCun, Guillaume Desjardins, Pascal Lamblin, James Bergstra, Nicolas Le Roux, Max Welling, Myriam Côté, Jérôme Louradour, Ronan Collobert, Jason Weston

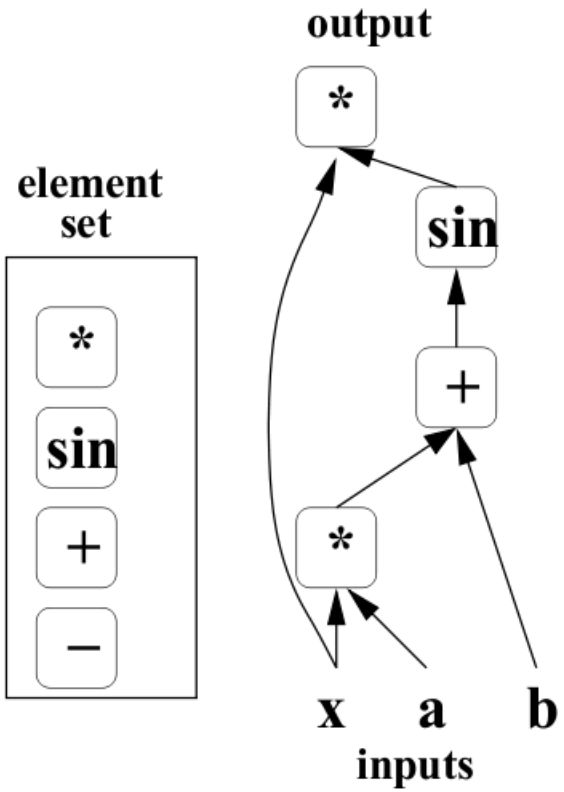
Interesting Experimental Results with Deep Architectures

- Beating shallow neural networks on vision and NLP tasks
- Beating SVMs on visions tasks from pixels (and handling dataset sizes that SVMs cannot handle in NLP)
- Reaching or beating state-of-the-art performance in NLP
- Beating deep neural nets without unsupervised component
- Learn visual features similar to V1 and V2 neurons

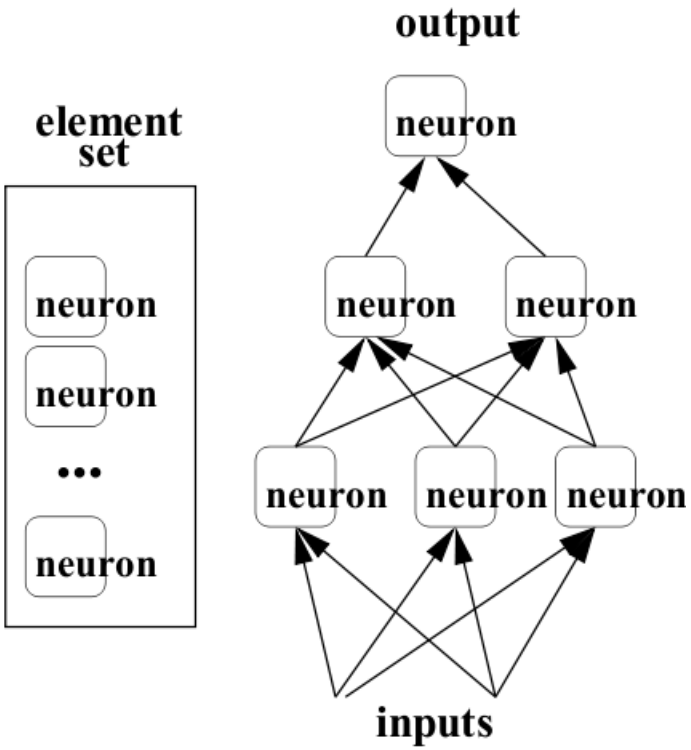
Deep Motivations

- Brains have a deep architecture
- Humans organize their ideas hierarchically, through composition of simpler ideas
- Unsufficiently deep architectures can be exponentially inefficient
- Distributed (possibly sparse) representations necessary to achieve non-local generalization, exponentially more efficient than 1-of-N enumeration latent variable values
- Multiple levels of latent variables allow combinatorial sharing of statistical strength

Architecture Depth



Depth = 4



Depth = 3

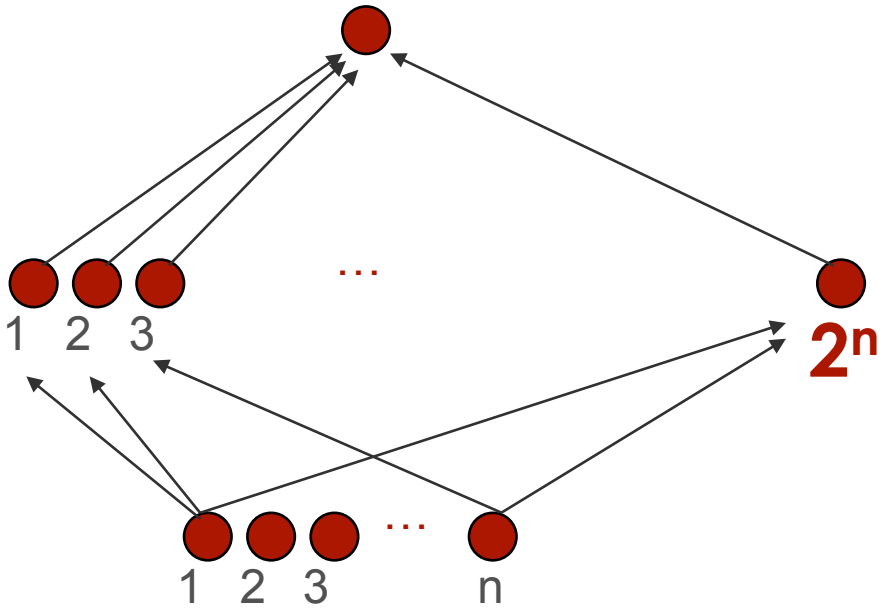
Deep Architectures are More Expressive

Theoretical arguments:

2 layers of { Logic gates / Formal neurons / RBF units } = universal approximator

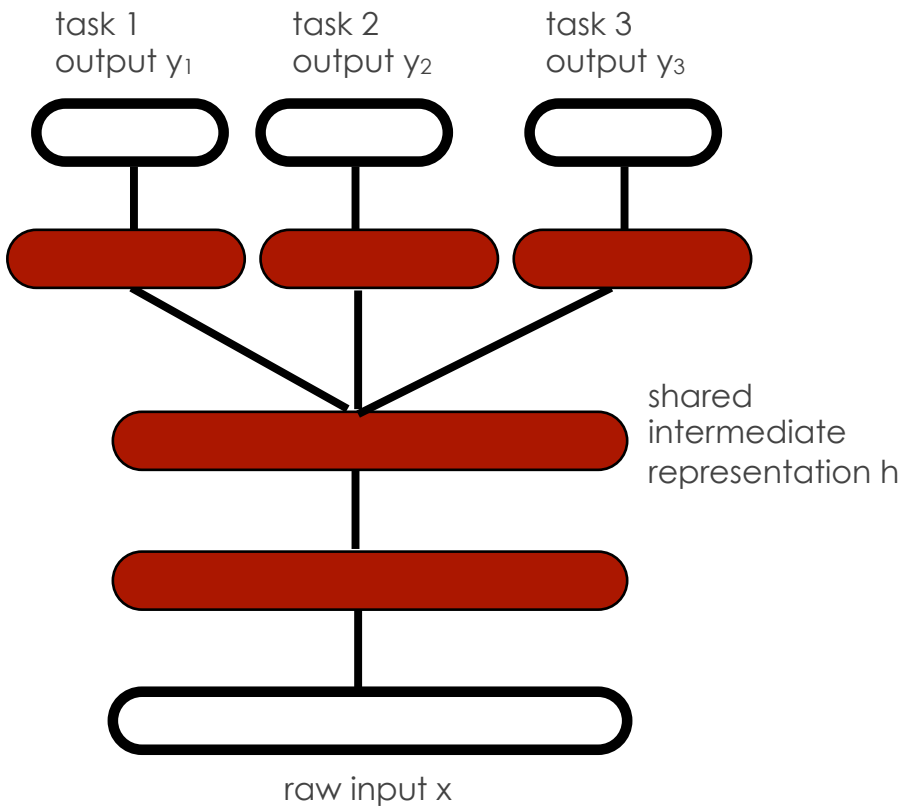
Theorems for all 3:
(Hastad et al 86 & 91, Bengio et al 2007)

Functions compactly represented with k layers may require exponential size with k-1 layers



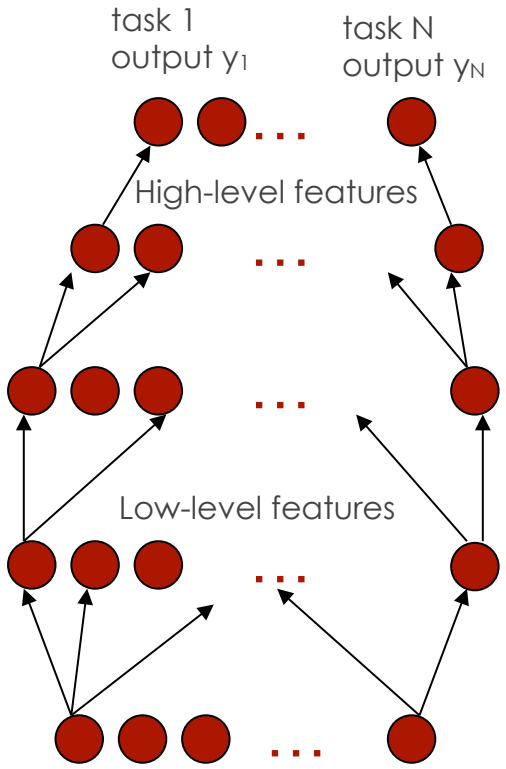
Deep Architectures and Sharing Statistical Strength, Multi-Task Learning

- Generalizing better to new tasks is crucial to approach AI
- Deep architectures learn good intermediate representations that can be shared across tasks
- A good representation is one that makes sense for many tasks

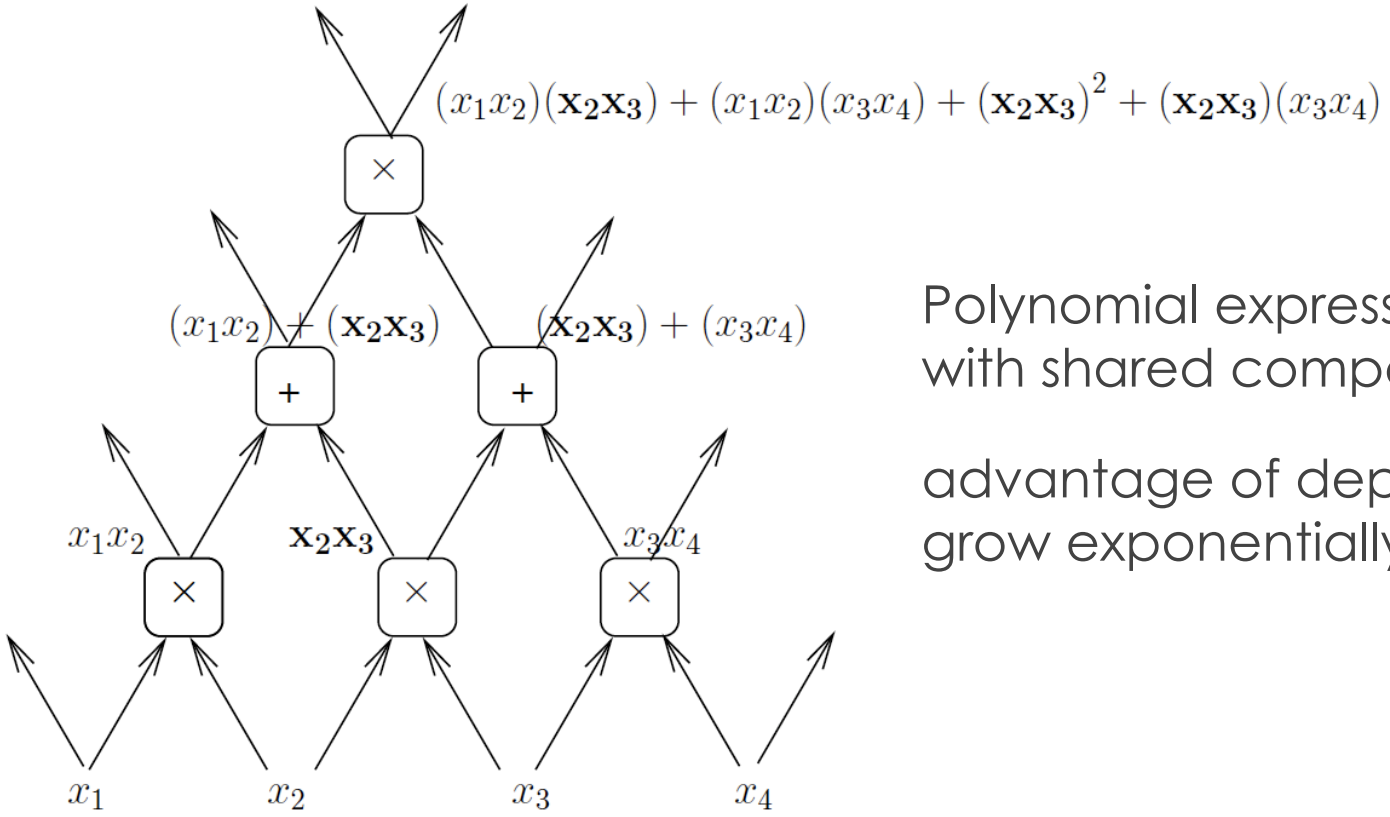


Feature and Sub-Feature Sharing

- Different tasks can share the same high-level feature
- Different high-level features can be built from the same set of lower-level features
- More levels = up to exponential gain in representational efficiency



Sharing Components in a Deep Architecture



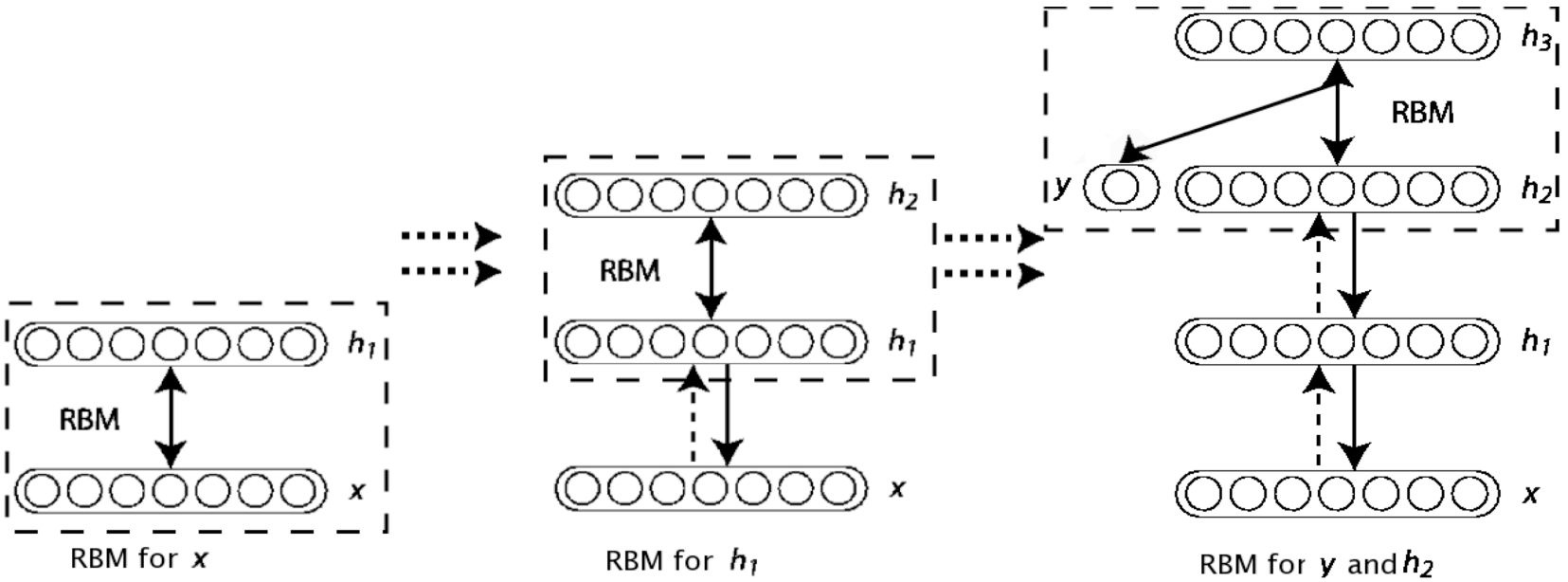
Polynomial expressed with shared components:

advantage of depth may grow exponentially

The Deep Breakthrough

- Before 2006, training deep architectures was unsuccessful, except for convolutional neural nets
- Hinton, Osindero & Teh « A Fast Learning Algorithm for Deep Belief Nets », *Neural Computation*, 2006
- Bengio, Lamblin, Popovici, Larochelle « Greedy Layer-Wise Training of Deep Networks », *NIPS'2006*
- Ranzato, Poultney, Chopra, LeCun « Efficient Learning of Sparse Representations with an Energy-Based Model », *NIPS'2006*

Greedy Layer-Wise Pre-Training



Stacking Restricted Boltzmann Machines (RBM) → Deep Belief Network (DBN)

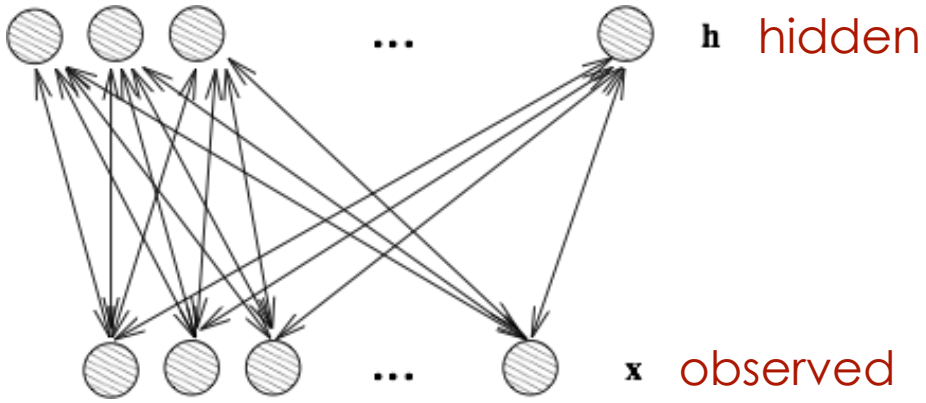
Greedy Layer-Wise Unsupervised Pre-Training Algorithm

- Train unsupervised feature extractor (e.g. RBM, auto-encoder) mapping input x to representation h_1, capturing main factors of variation in x (models $P(x)$)
- Taking $h_1(x)$ as an input, train a second unsupervised feature extractor, obtaining representation h_2 of x
- Etc. to level k gives h_k
- Plug a supervised classifier $P(Y \mid h_k(x))$ on top, taking h_k as input
- Fine-tune parameters of whole system $P(Y \mid x)$ wrt supervised objective

Restricted Boltzman Machine

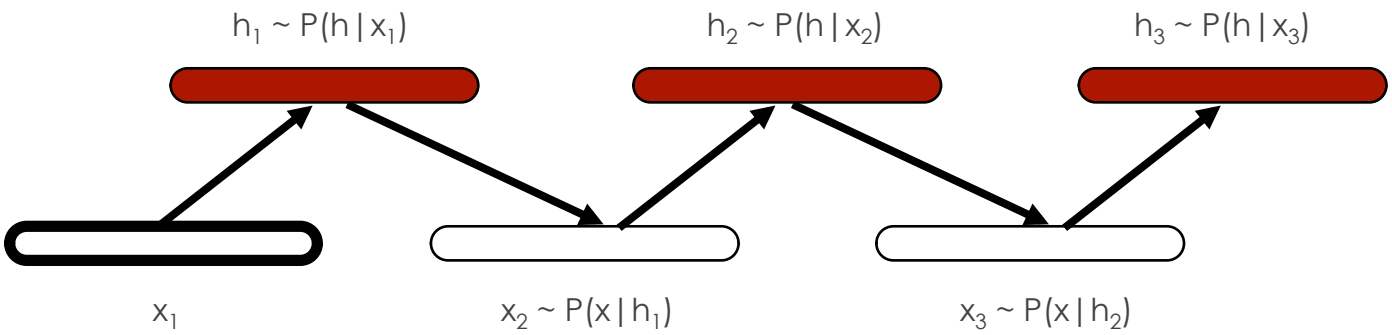
- The most popular building block for deep architectures
  (Smolensky 86, Hinton 2002)

$$P(x,h) = \frac{1}{Z} e^{b^T h + c^T x + h^T W x}$$



- Bipartite undirected graphical model
- *h~P(h | x)*, or *(P(h$_i$=1 | x))* are representations of x

Gibbs Sampling in RBMs



P(h | x) and P(x | h) factorize

$$P(x,h) = \frac{1}{Z} e^{b^T h + c^T x + h^T W x}$$

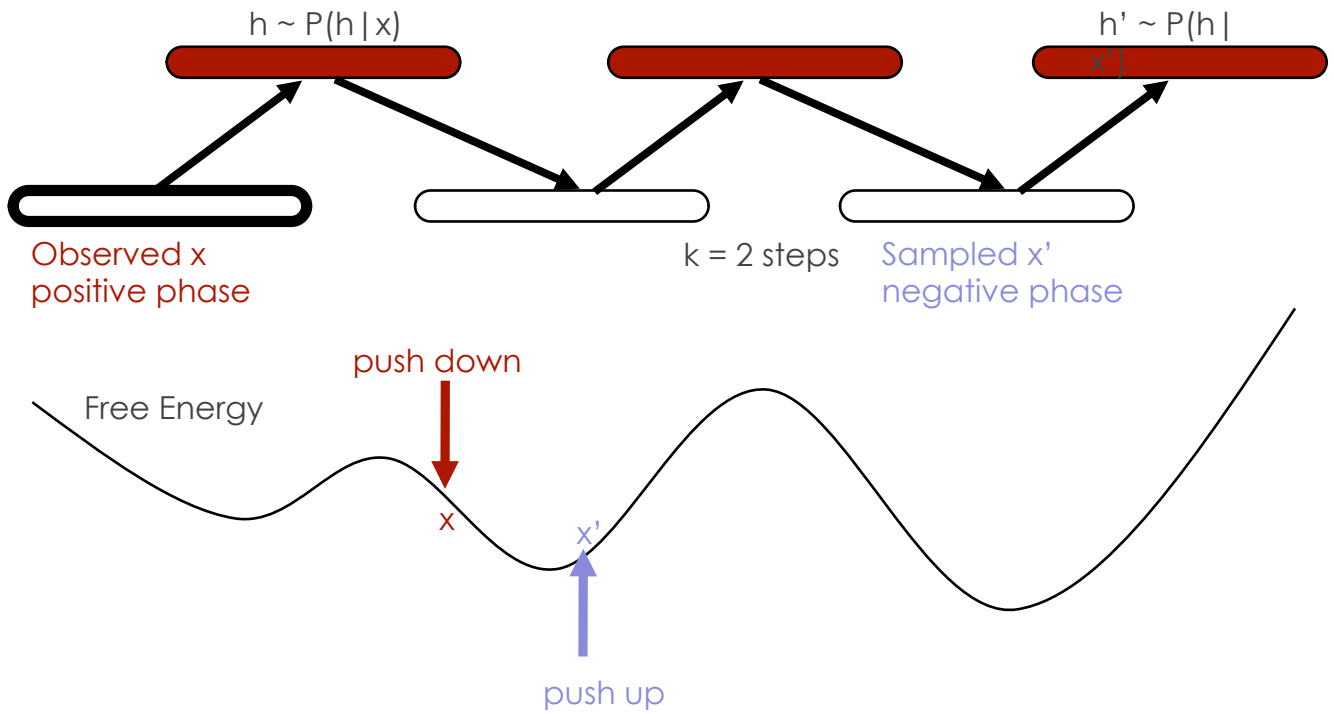
- Easy inference
- Convenient Gibbs sampling
  x→h→x→h…

Training RBMs

Contrastive Divergence: (CD-k)	start negative Gibbs chain at observed x, run k Gibbs steps
Persistent CD: (PCD)	run negative Gibbs chain in background while weights slowly change
Fast PCD:	two sets of weights, one with a large learning rate only used for negative phase, quickly exploring modes
Herding:	(see Max Welling's ICML, UAI and ICML workshop talks)
Tempered MCMC:	use higher temperature to escape modes

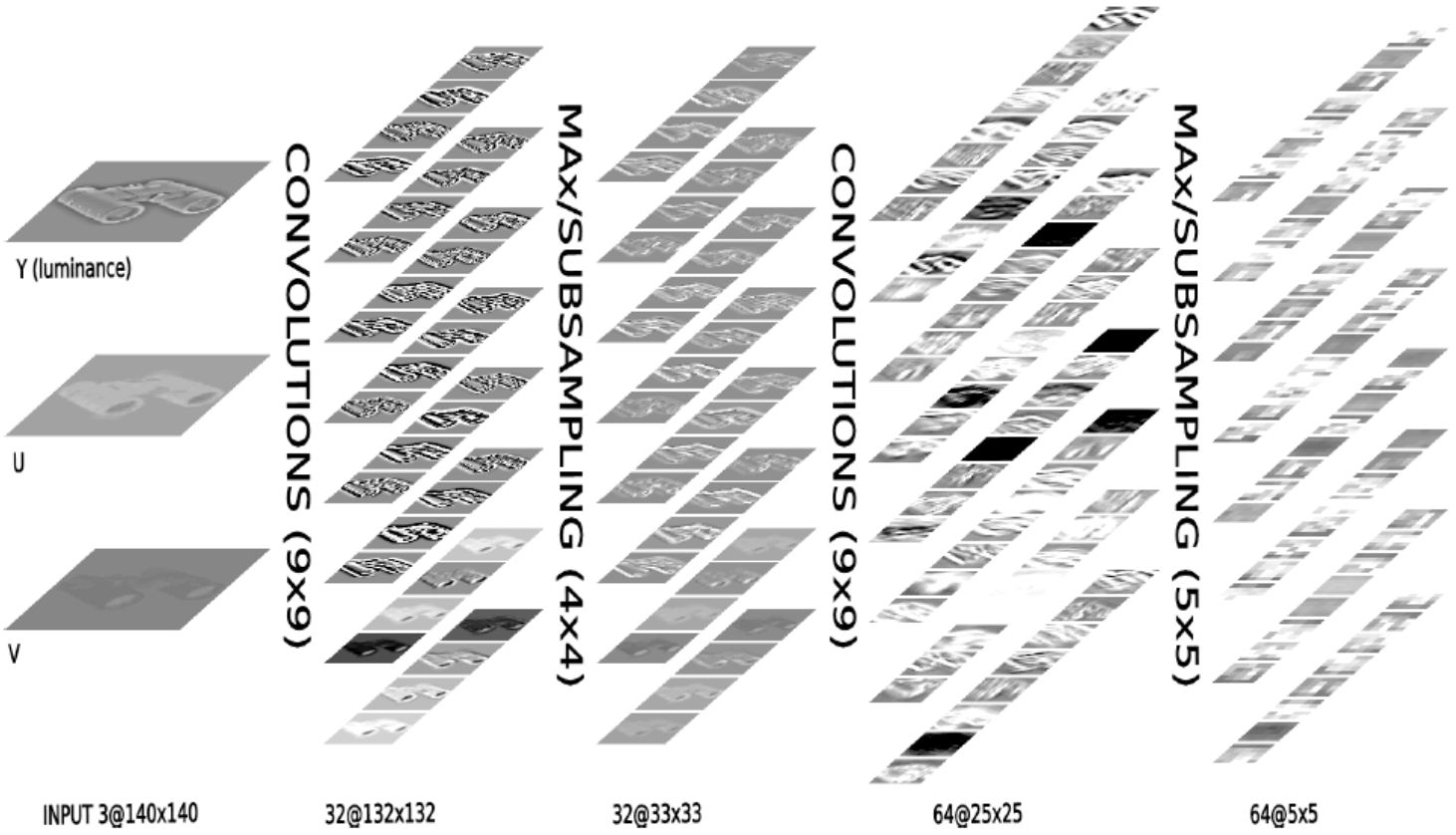
Contrastive Divergence

Contrastive Divergence (CD-k): start negative phase block Gibbs chain at observed x, run k Gibbs steps (Hinton 2002)



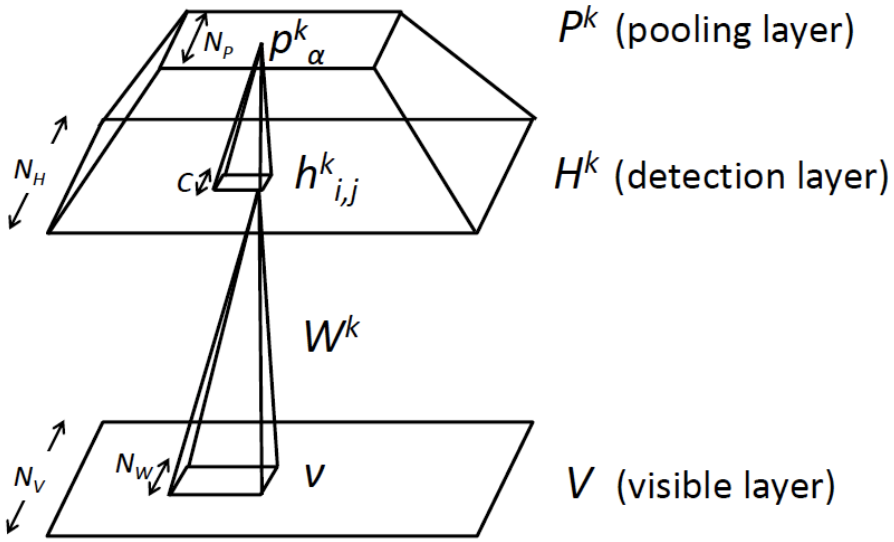
Deep Convolutional Architectures

Mostly from Le Cun's (NYU) and Ng's (Stanford) groups:
state-of-the-art on MNIST digits, Caltech-101 objects, faces



Convolutional DBNs

(Lee et al, ICML'2009)

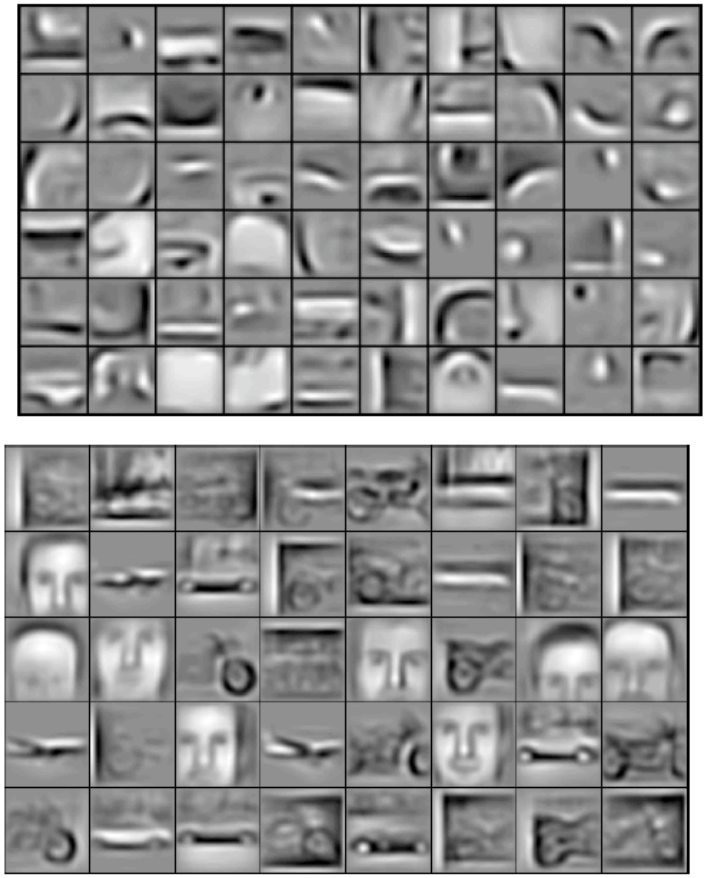


P^k (pooling layer)

H^k (detection layer)

V (visible layer)

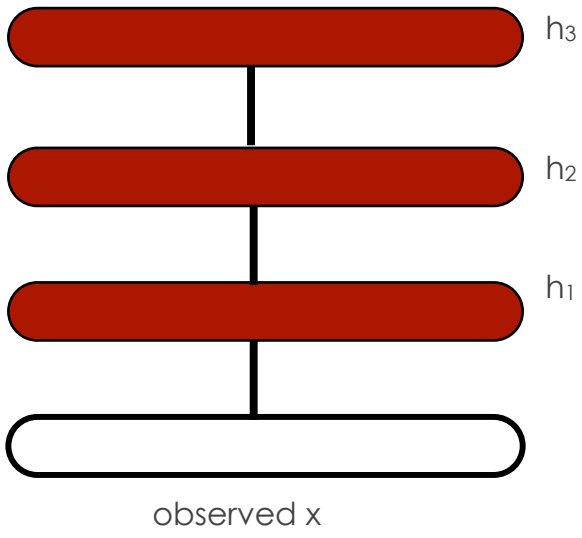
faces, cars, airplanes, motorbikes



Deep Boltzman Machines

(Salakhutdinov et al, AISTATS 2009, Lee et al, ICML 2009)

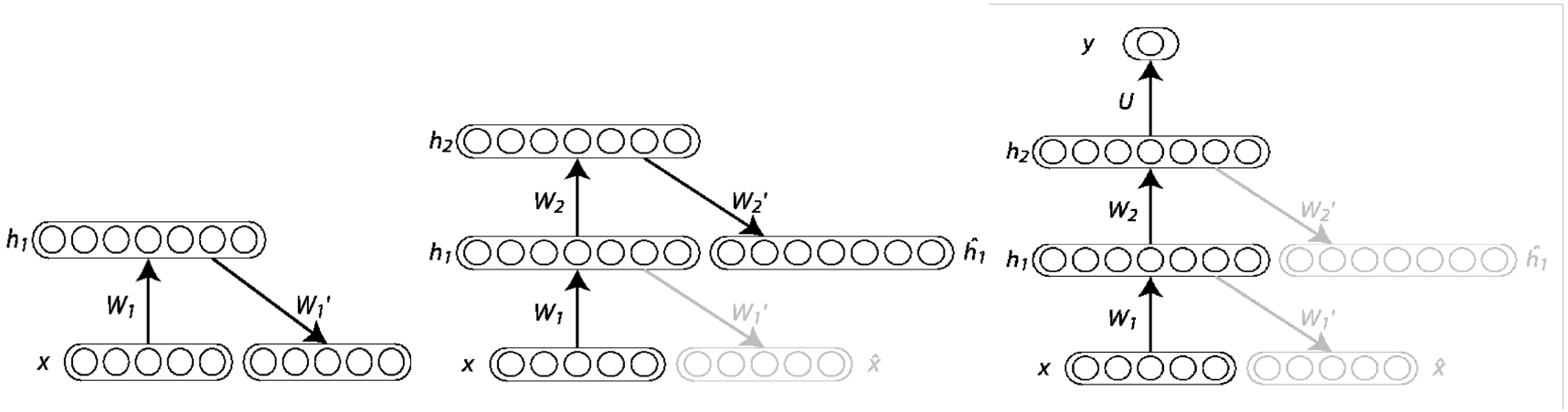
- Positive phase: variational approximation (mean-field)
- Negative phase: persistent chain
- **Can (must) initialize from stacked RBMs**
- Improved performance on MNIST from 1.2% to .95% error
- Can apply AIS with 2 hidden layers



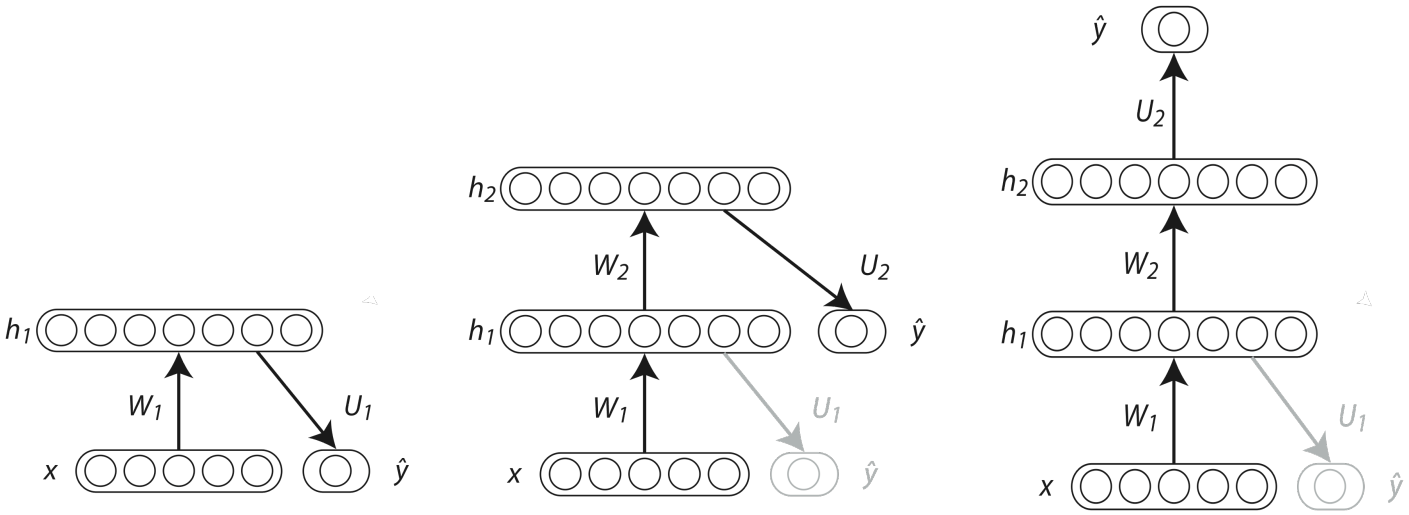
Why are Classifiers Obtained from DBNs Working so Well?

- General principles?
- Would these principles work for other single-level algorithms?
- Why does it work?

Stacking Auto-Encoders



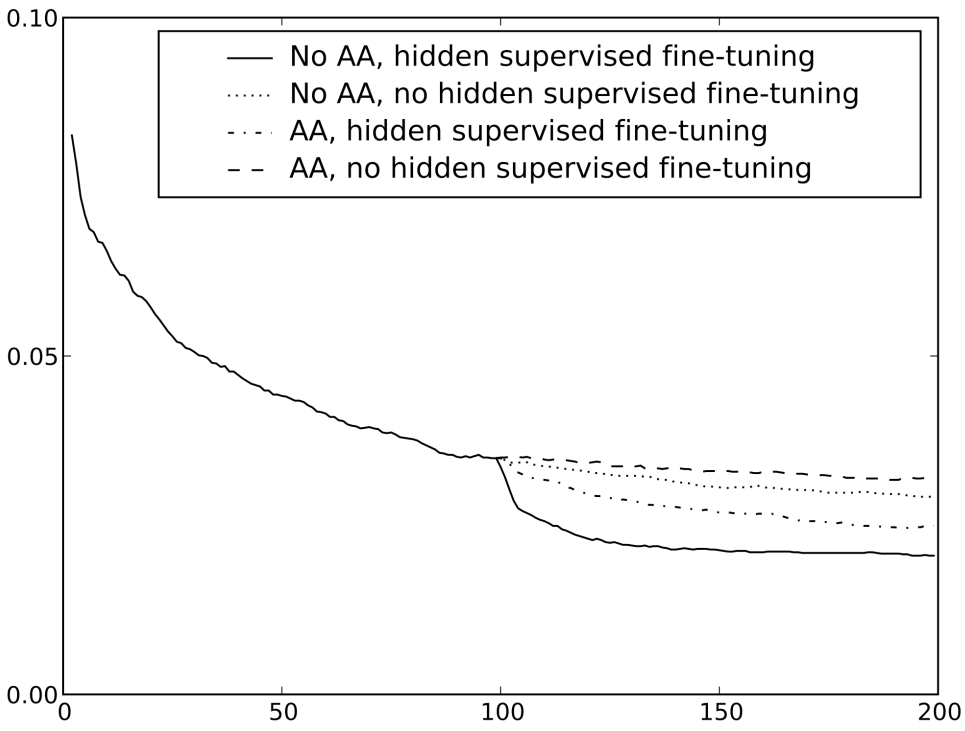
Greedy Layerwise Supervised Training



Generally worse than unsupervised pre-training but better than ordinary training of a deep neural network (Bengio et al. 2007).

Supervised Fine-Tuning is Important

- Greedy layer-wise unsupervised pre-training phase with RBMs or auto-encoders on MNIST
- Supervised phase with or without unsupervised updates, with or without fine-tuning of hidden layers
- Can train all RBMs at the same time, same results

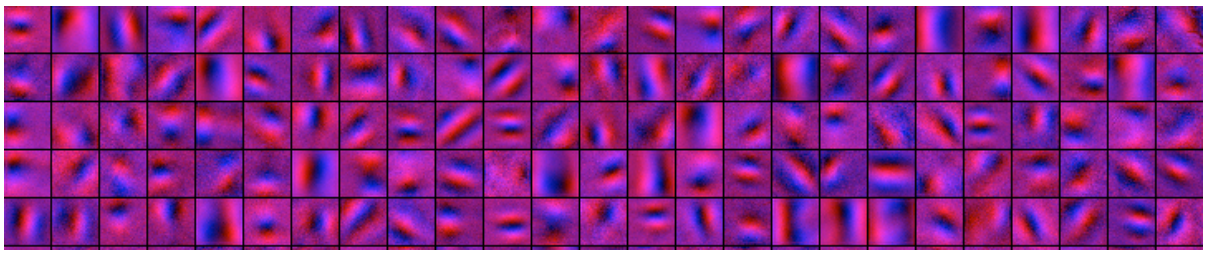


Level-Local Learning is Important

- Initializing each layer of an unsupervised deep Boltzmann machine helps a lot
- Initializing each layer of a supervised neural network as an RBM or auto-encoder helps a lot
- Helps most the layers further away from the target
- Jointly training all the levels of a deep architecture is difficult

Replacing RBMs by Other Layer-Local Unsupervised Learning

- Auto-encoders (Bengio et al, NIPS'2006)
- Sparse auto-encoders (Ranzato et al, NIPS'2006)
- Kernel PCA (Erhan 2008)
- Denoising auto-encoders (Vincent et al, ICML'2008)
- Unsupervised embedding (Weston et al, ICML'2008)
- Slow features (Mohabi et al, ICML'2009, Bergstra & Bengio NIPS'2009)



Sparse Auto-Encoders

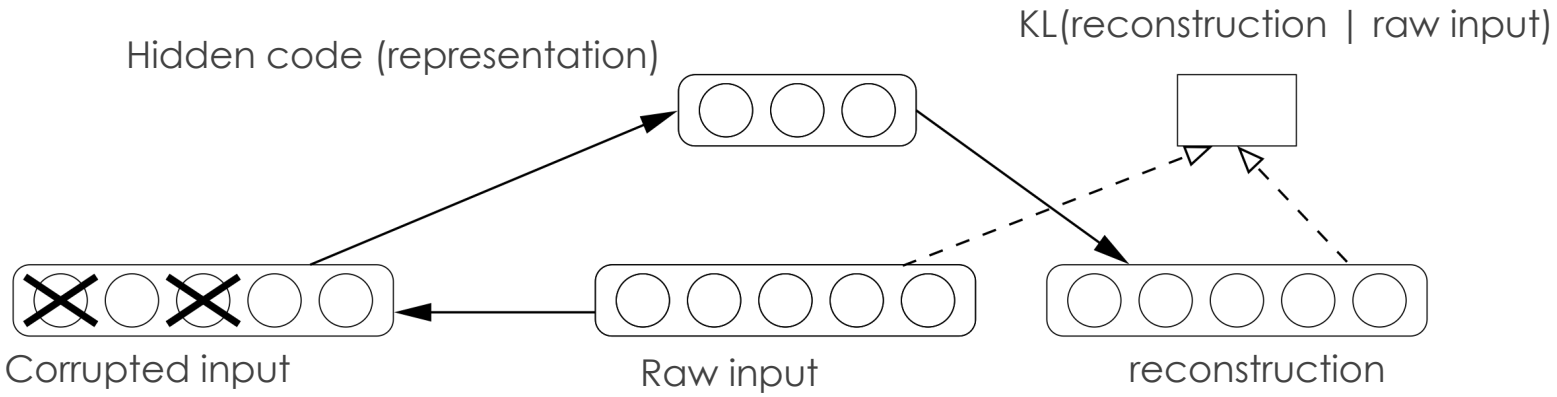
(Ranzato et al, 2007; Ranzato et al 2008)

- Sparsity penalty on the intermediate codes
- Like sparse coding but with efficient run-time encoder
- Sparsity penalty pushes up the free energy of all configurations (proxy for minimizing the partition function)
- Impressive results in object classification (convolutional nets):
  - MNIST 0.5% error = record-breaking
  - Caltech-101 65% correct = state-of-the-art (Jarrett et al, ICCV 2009)
- Similar results obtained with a convolutional DBN (Lee et al, ICML'2009)

Denoising Auto-Encoder

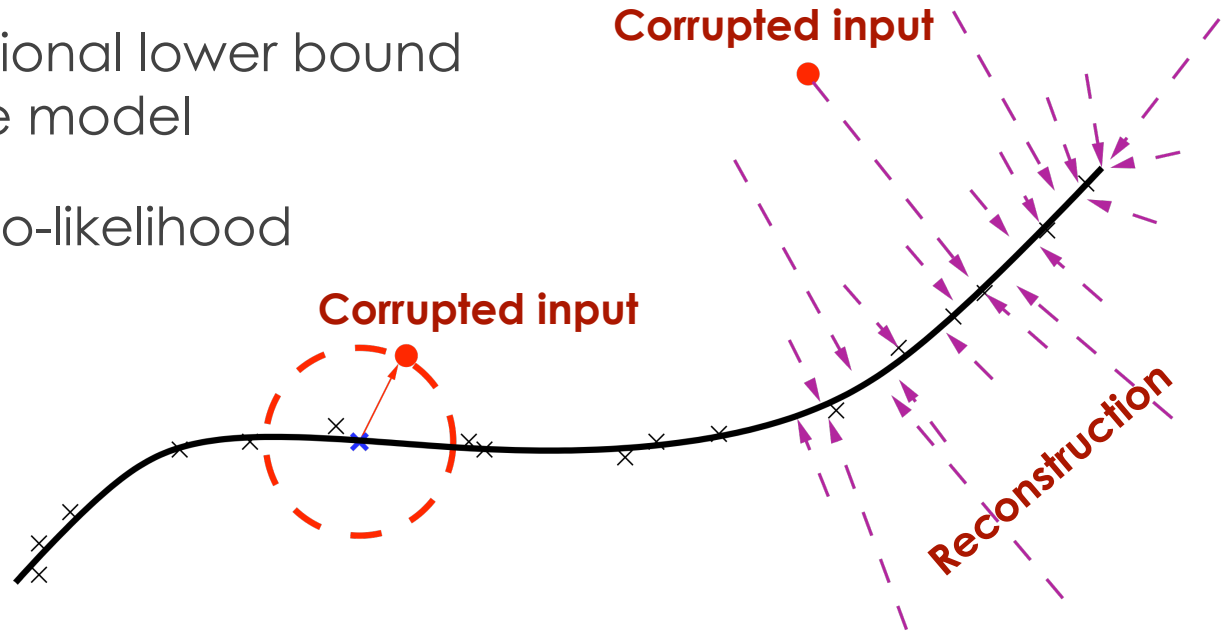
(Vincent et al, ICML 2008)

- Corrupt the input
- Reconstruct the uncorrupted input



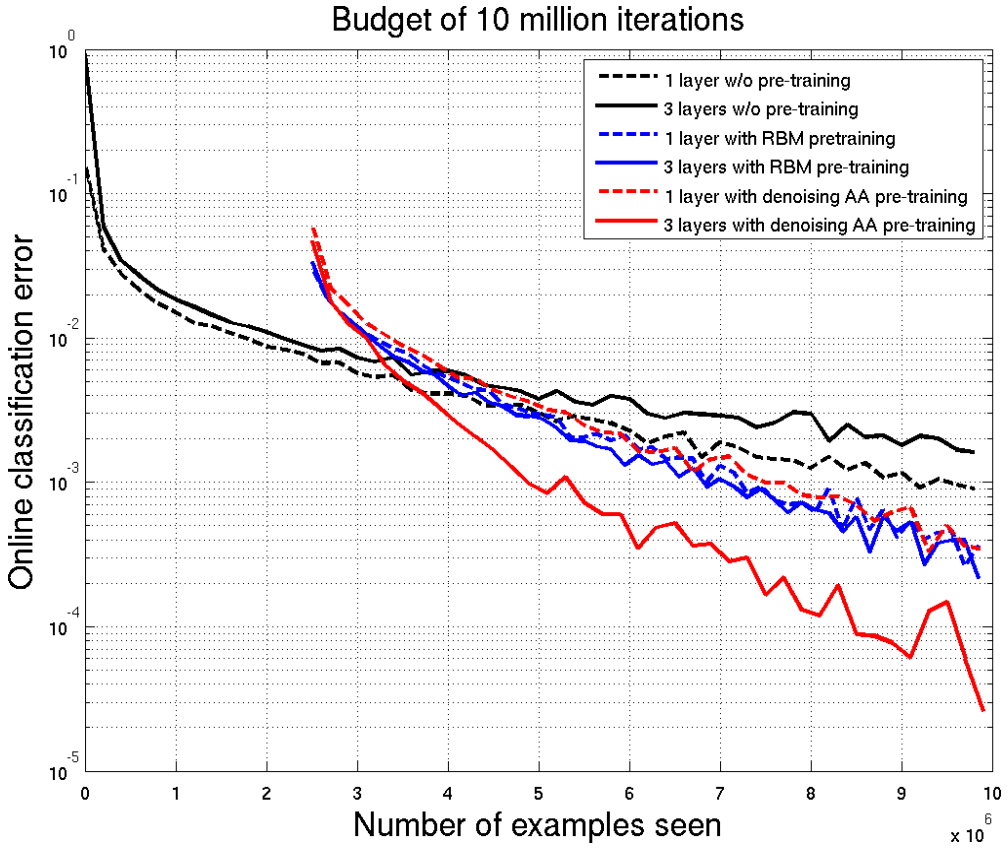
Denoising Auto-Encoder

- Learns a vector field towards higher probability regions
- Minimizes variational lower bound on a generative model
- Similar to pseudo-likelihood



Stacked Denoising Auto-Encoders

- No partition function, can measure training criterion
- Encoder & decoder: any parametrization
- Performs as well or better than stacking RBMs for usupervised pre-training



Infinite MNIST

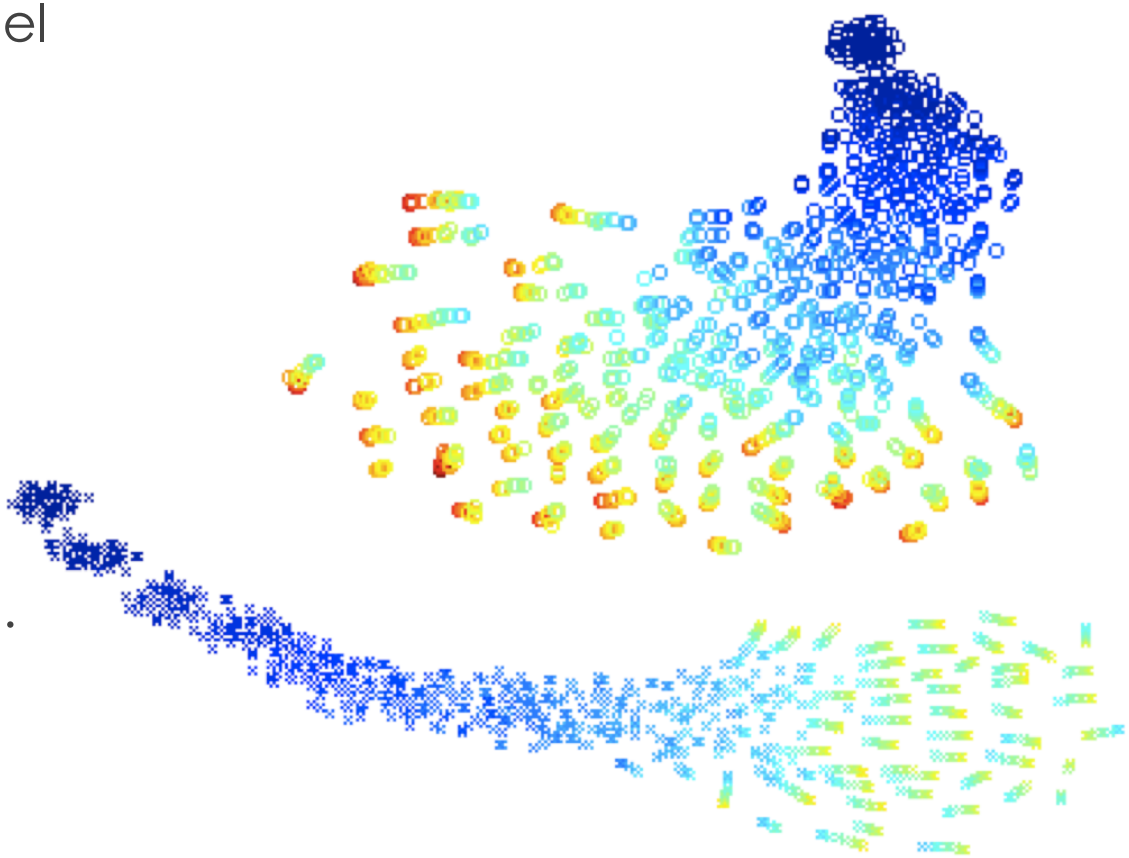
Why is Unsupervised Pre-Training Working So Well?

- Regularization hypothesis:
  - Unsupervised component forces model close to P(x)
  - Representations good for P(x) are good for P(y | x)

- Optimization hypothesis:
  - Unsupervised initialization near better local minimum of P(y | x)
  - Can reach lower local minimum otherwise not achievable by random initialization

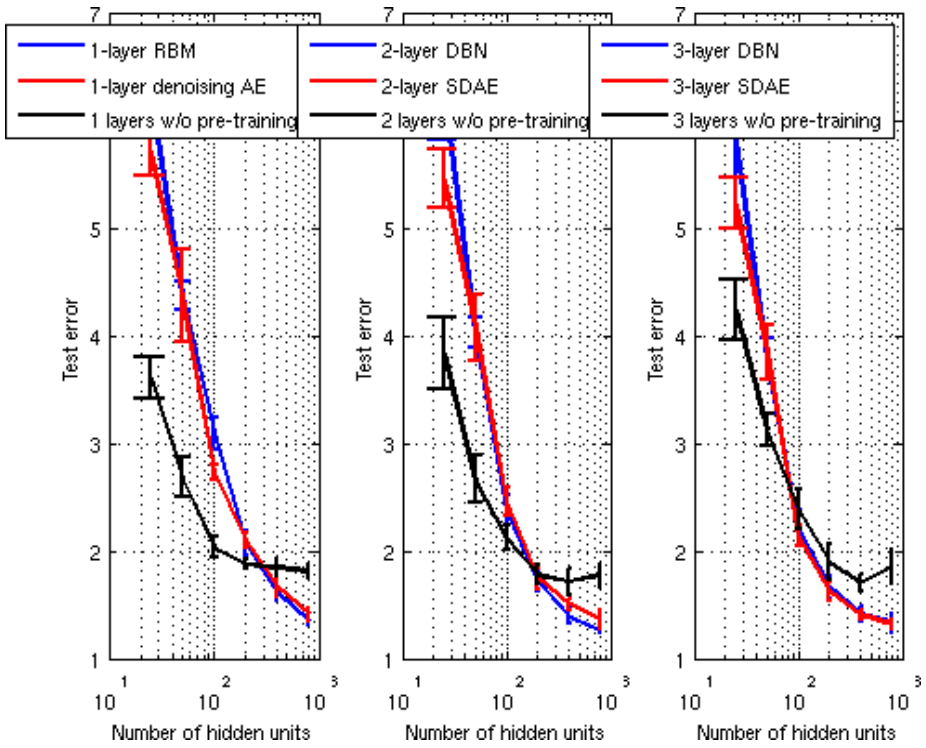
Learning Trajectories in Function Space

- Each point a model in function space
- Color = epoch
- Top: trajectories w/o pre-training
- Each trajectory converges in different local min.
- No overlap of regions with and w/o pre-training



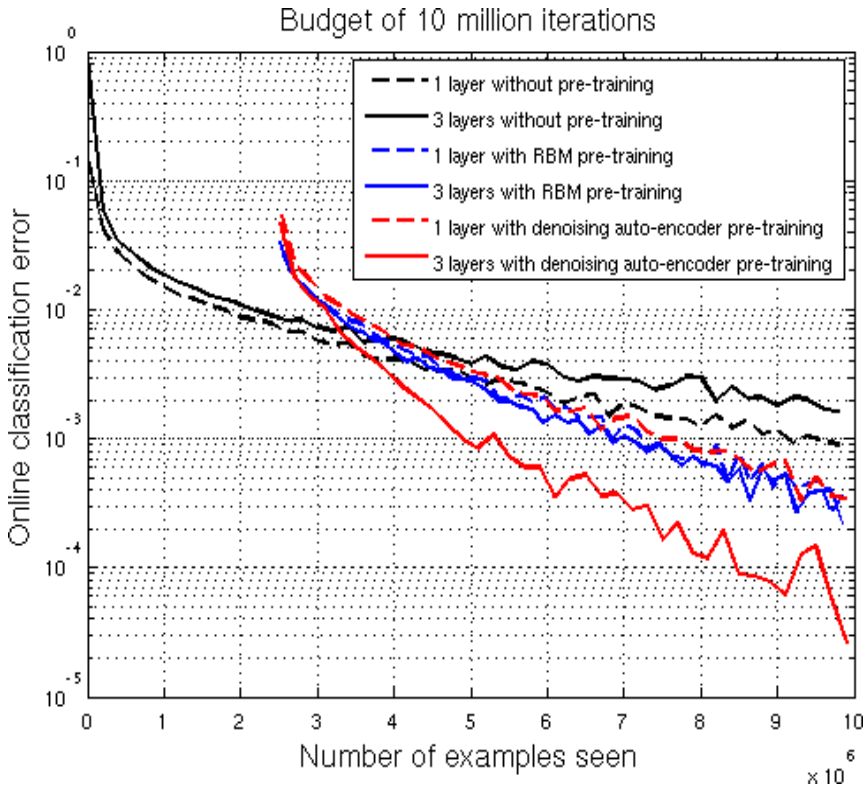
Unsupervised Learning as Regularizer

- Adding extra regularization (reducing # hidden units) hurts more the pre-trained models
- Pre-trained models have less variance wrt training sample
- Regularizer = infinite penalty outside of region compatible with unsupervised pre-training

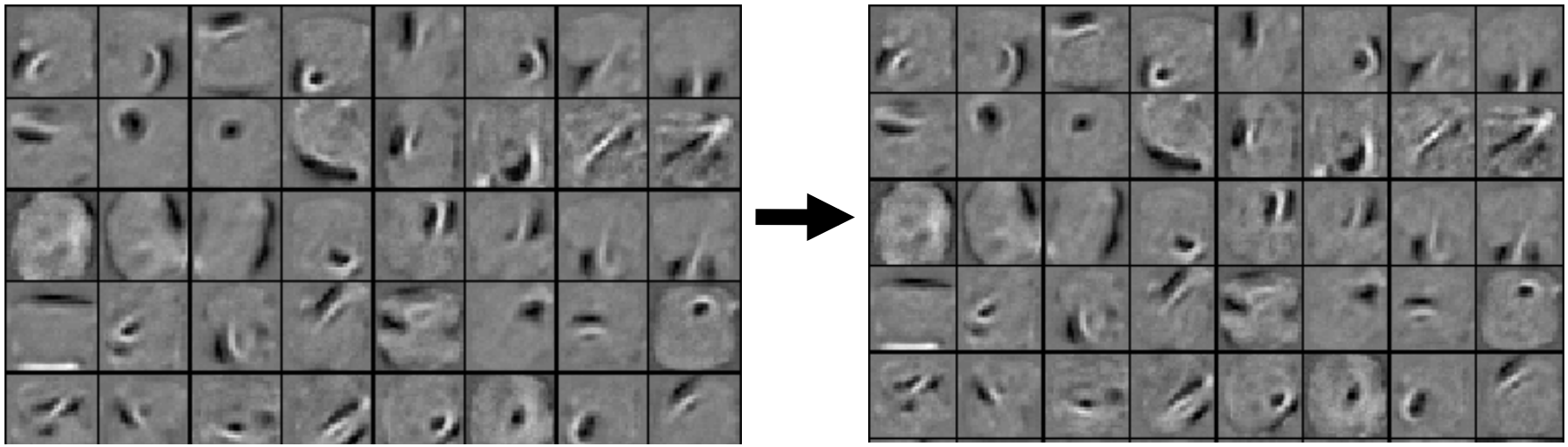


Better Optimization of Online Error

- Both training and online error are smaller with unsupervised pre-training
- As # samples → ∞ training err. = online err. = generalization err.
- Without unsup. pre-training: can't exploit capacity to capture complexity in target function from training data



Learning Dynamics of Deep Nets

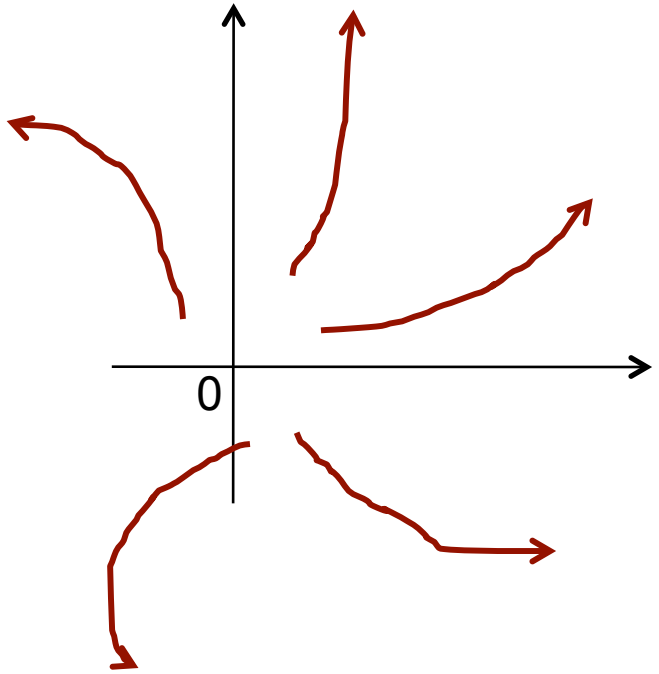


Before fine-tuning

After fine-tuning

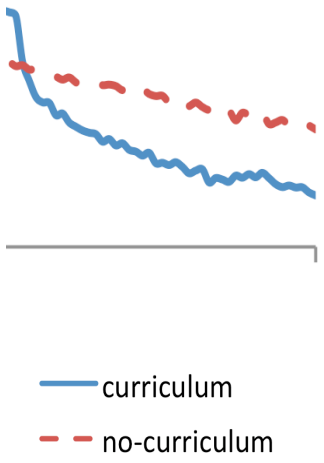
Learning Dynamics of Deep Nets

- As weights become larger, get trapped in basin of attraction ("quadrant" does not change)
- Initial updates have a crucial influence ("critical period"), explain more of the variance
- Unsupervised pre-training initializes in basin of attraction with good generalization properties



Order & Selection of Examples Matters

- Curriculum learning
  (Bengio et al, ICML'2009; Krueger & Dayan 2009)
- Start with easier examples
- Faster convergence to a better local minimum in deep architectures
- Also acts like a regularizer with optimization effect?
- Influencing learning dynamics can make a big difference



Take-Home Messages

- Multiple levels of latent variables: potentially exponential gain in statistical sharing
- RBMs allow fast inference, stacked RBMs / auto-encoders have fast approximate inference
- Gibbs sampling in RBMs sometimes does not mix well, but sampling and learning can interact in surprisingly useful ways
- Unsupervised pre-training of classifiers acts like a strange regularizer with improved optimization of online error
- At least as important as the model: the inference approximations and the learning dynamics

Research Program

- Unsupervised pre-training is good but we want more
- Understand why gradient-based optimization of lower layers of deep supervised architecture gets stuck
- Is it the same reason that global coordination fails in deep Boltzmann machines?
- Is it related to problem with recurrent nets and dynamic Bayes net failing with long-term dependencies? Important to capture contextual effects in sequential data such as video and text.
- Applications to information retrieval:

Deep Representations for Information Retrieval

- Sparsity to help computational & representational efficiency
  - Combine semantic hashing idea (make representation gradually nearly binary) with sparse code penalty (reduced cost of measuring similarity between objects)
  - Allow effective number of non-zeros to vary per example (representational efficiency)
- Combining supervised (ranking) & unsupervised criteria **online**
  - Current training process with phases not practical with huge datasets = online
- Put different modalities (image, query) in the same space
  - May actually help each other during training
  - Learn a distributed representation of requests to generalize across rarely occuring requests

Thank you for your attention!

- Questions?
- Comments?