Understanding and Improving Deep Learning Algorithms

Yoshua Bengio, U. Montreal

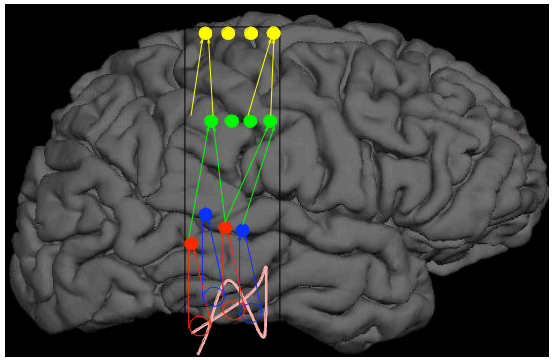
CMU, Pittsburgh

April 22nd, 2010

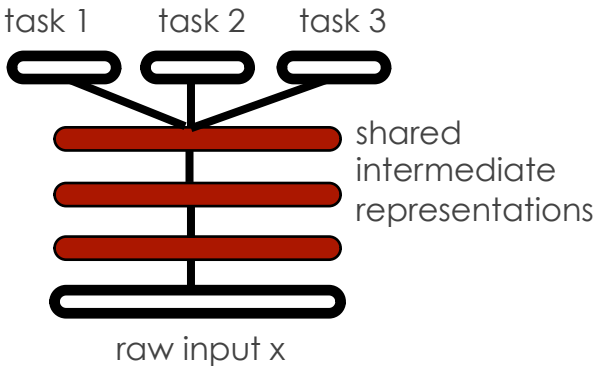
ML Google Distinguished Lecture

Thanks to: Aaron Courville, Xavier Glorot, Dumitru Erhan, Olivier Delalleau, Olivier Breuleux, Pascal Vincent, Guillaume Desjardins, James Bergstra, Pascal Lamblin

Deep Motivations



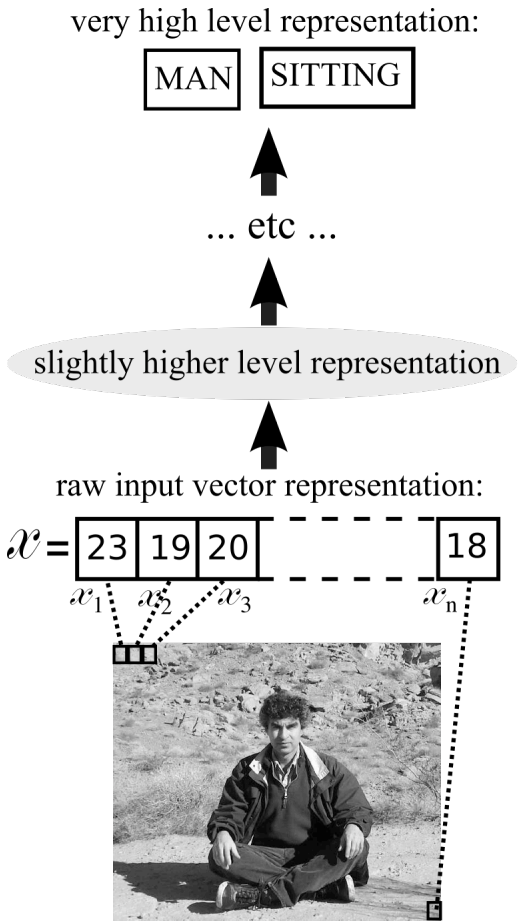
- Brains have a deep architecture
- Humans organize their ideas hierarchically, through composition of simpler ideas
- Unsufficiently deep architectures can be exponentially inefficient
- Distributed (possibly sparse) representations necessary to achieve non-local generalization, exponentially more efficient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



Deep Architecture in our Mind

- Humans organize their ideas and concepts hierarchically
- Humans first learn simpler concepts and then compose them to represent more abstract ones
- Engineers break-up solutions into multiple levels of abstraction and processing

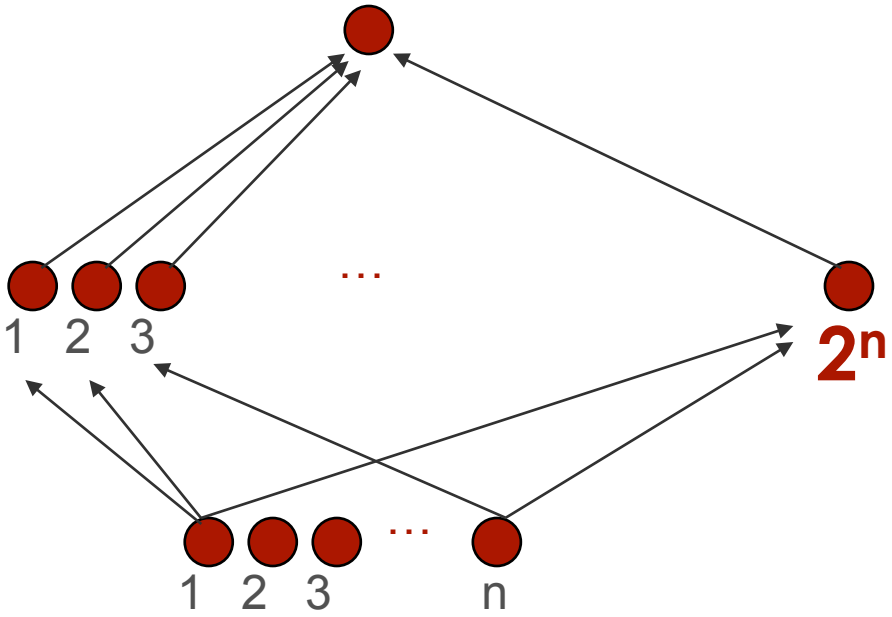
  It would be nice to learn / discover these concepts
  (knowledge engineering failed because of poor introspection?)



Deep Architectures are More Expressive

Theoretical arguments:

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

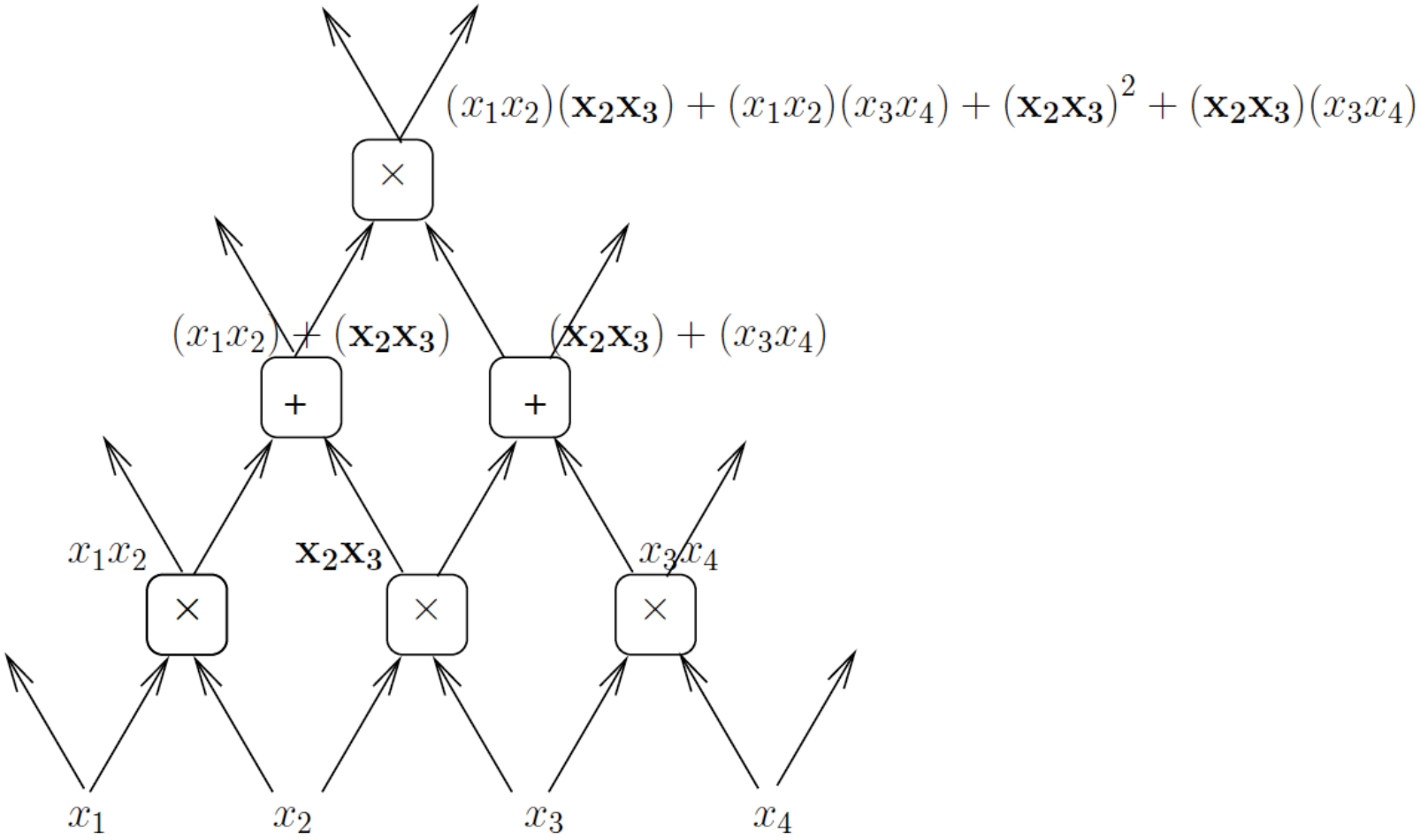


Theorems for all 3:
(Hastad et al 86 & 91, Bengio et al 2007)

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

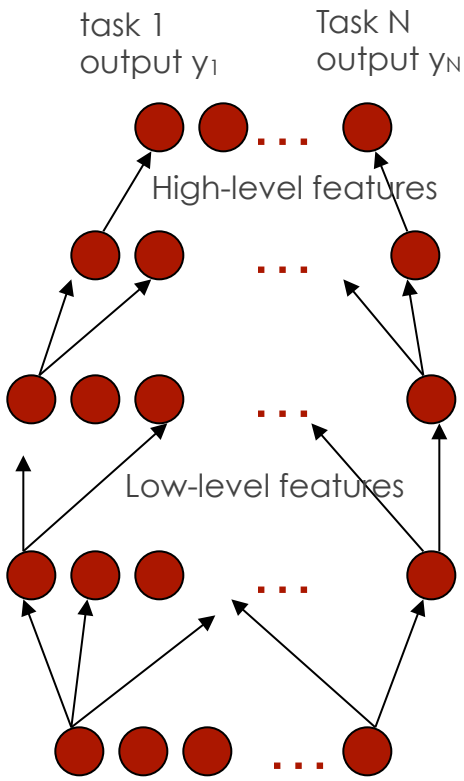
Sharing Components in a Deep Architecture

Polynomial expressed with shared components: advantage of depth may grow exponentially

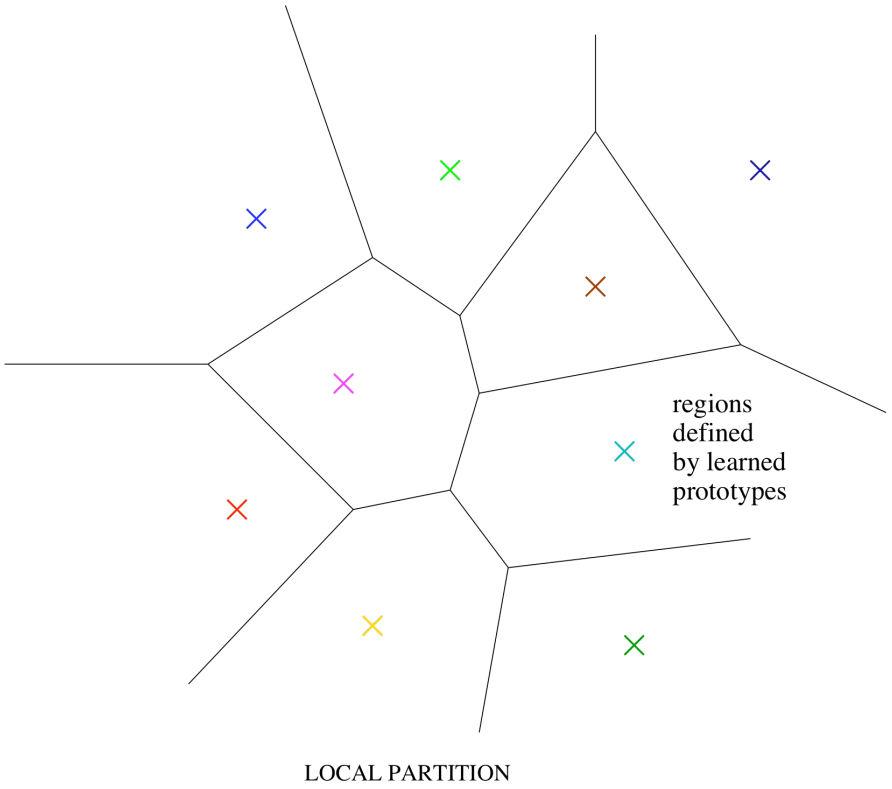


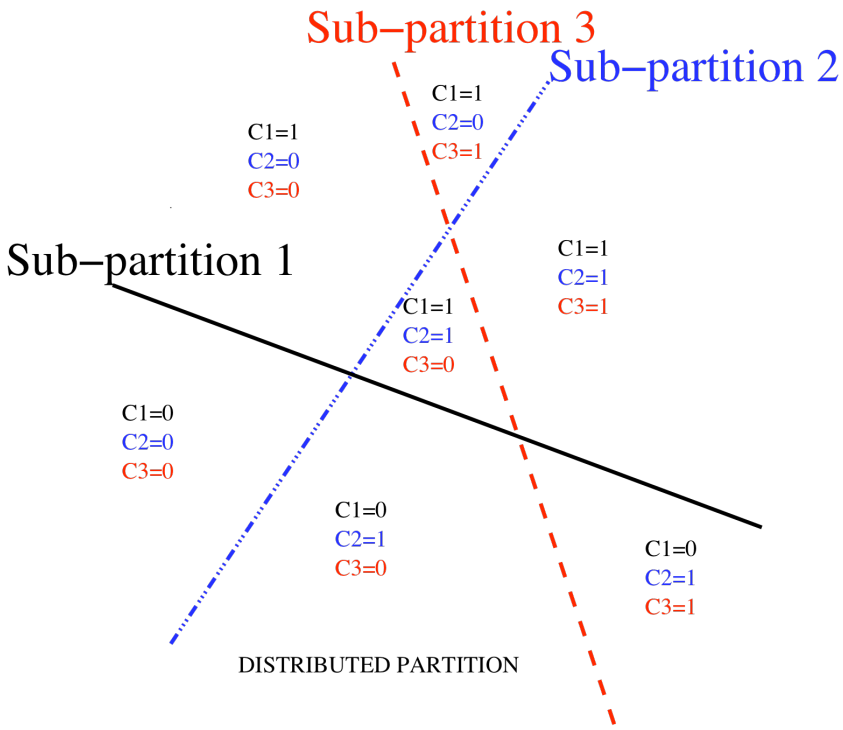
Feature and Sub-Feature Sharing

- 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

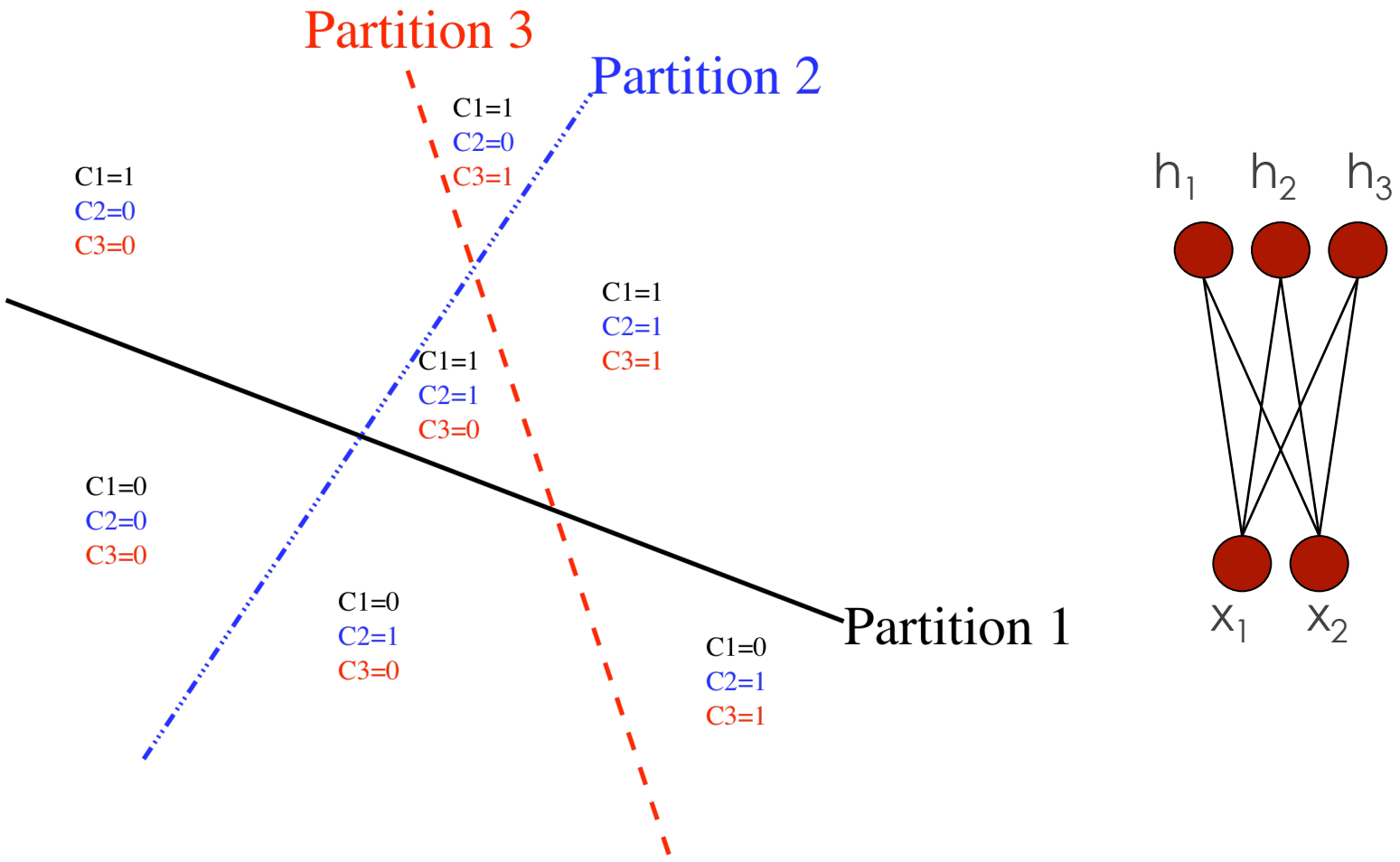


Local vs Distributed





RBM Hidden Units Carve Input Space

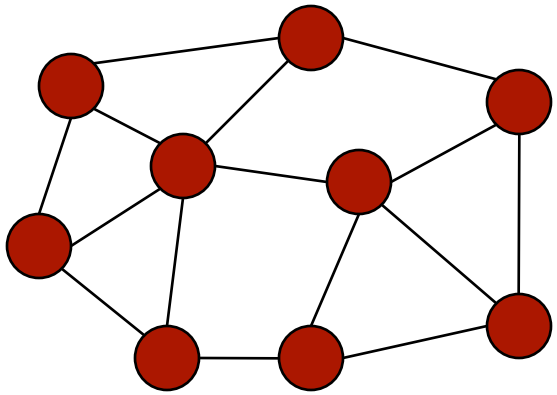


Boltzman Machines and MRFs

- Boltzmann machines: $P(x) = \frac{1}{Z} e^{-\text{Energy}(x)} = e^{c^T x + x^T W x}$
  (Hinton 84)

- Markov Random Fields:

$$P(x) = \frac{1}{Z} e^{\sum_i w_i f_i(x)}$$

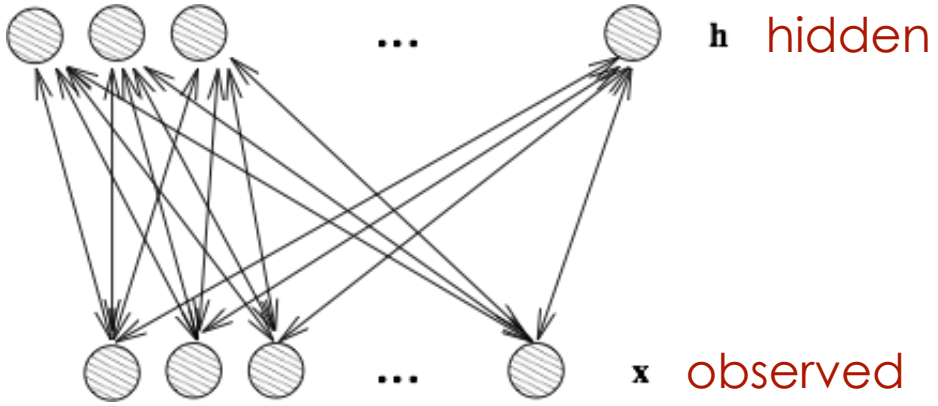


- More interesting with latent variables!

Restricted Boltzman Machine

- The most popular building block for deep architectures

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

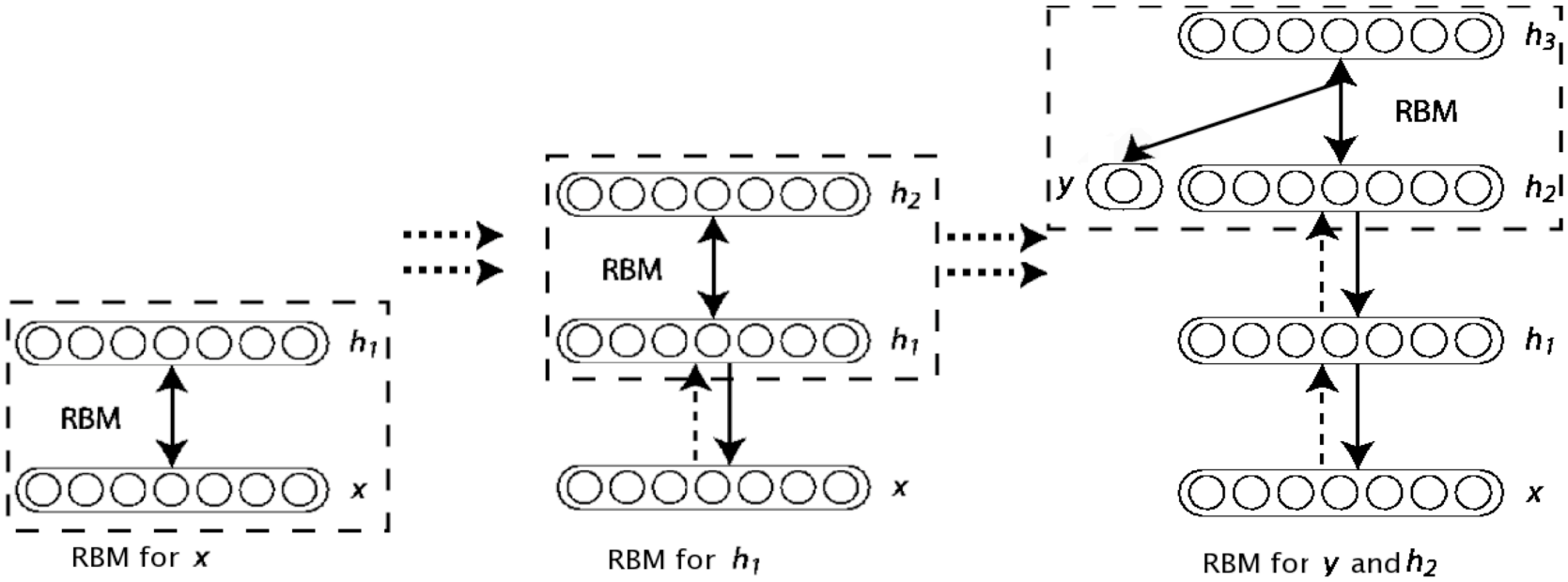


- Bipartite undirected graphical model
- Inference is trivial:

P(h | x) & *P(x | h)* factorize

Layer-Wise Unsupervised Pre-Training: learn multiple levels of representation

(Hinton et al 2006, Bengio et al 2007, Ranzato et al 2007)



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

or Stacking Denoising Auto-Encoders + supervised fine-tuning → classifier

Supervised fine-tuning of deep MLP

- DBN or stack of RBMs or stack of auto-encoders trained to model P(X), initialize a deep Multi-Layer Perceptron (MLP)
- Best classification results with (long) final supervised phase
- Trained by stochastic gradient descent on log P(Y | X)
- Why does unsupervised pre-training help?
- Stanford's alternative: only train a classifier (SVM) on top of concatenation of all hidden layers (and possibly input).

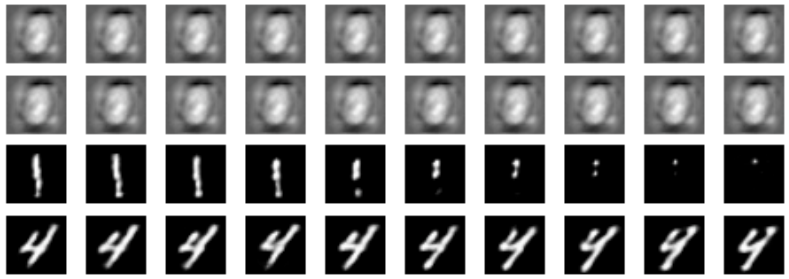
Training RBMs

Contrastive Divergence: (CD-k)	start negative Gibbs chain at observed x, run k Gibbs steps (Hinton 99)
Persistent CD: (PCD)	run negative Gibbs chain in background while weights slowly change (Tieleman 08)
Fast PCD:	two sets of weights, one with large learning rate only for negative phase, quickly exploring modes (Tieleman et al 09)
Herding:	PCD at 0-temperature, large learning rate (Welling 09)
Tempered MCMC:	use higher temperature to escape modes (Salakhutdinov 10; Desjardins et al 09)

Problems with CD & Gibbs Sampling

In practice, Gibbs sampling does not always mix well… (Tieleman 08)

RBM trained by CD on MNIST

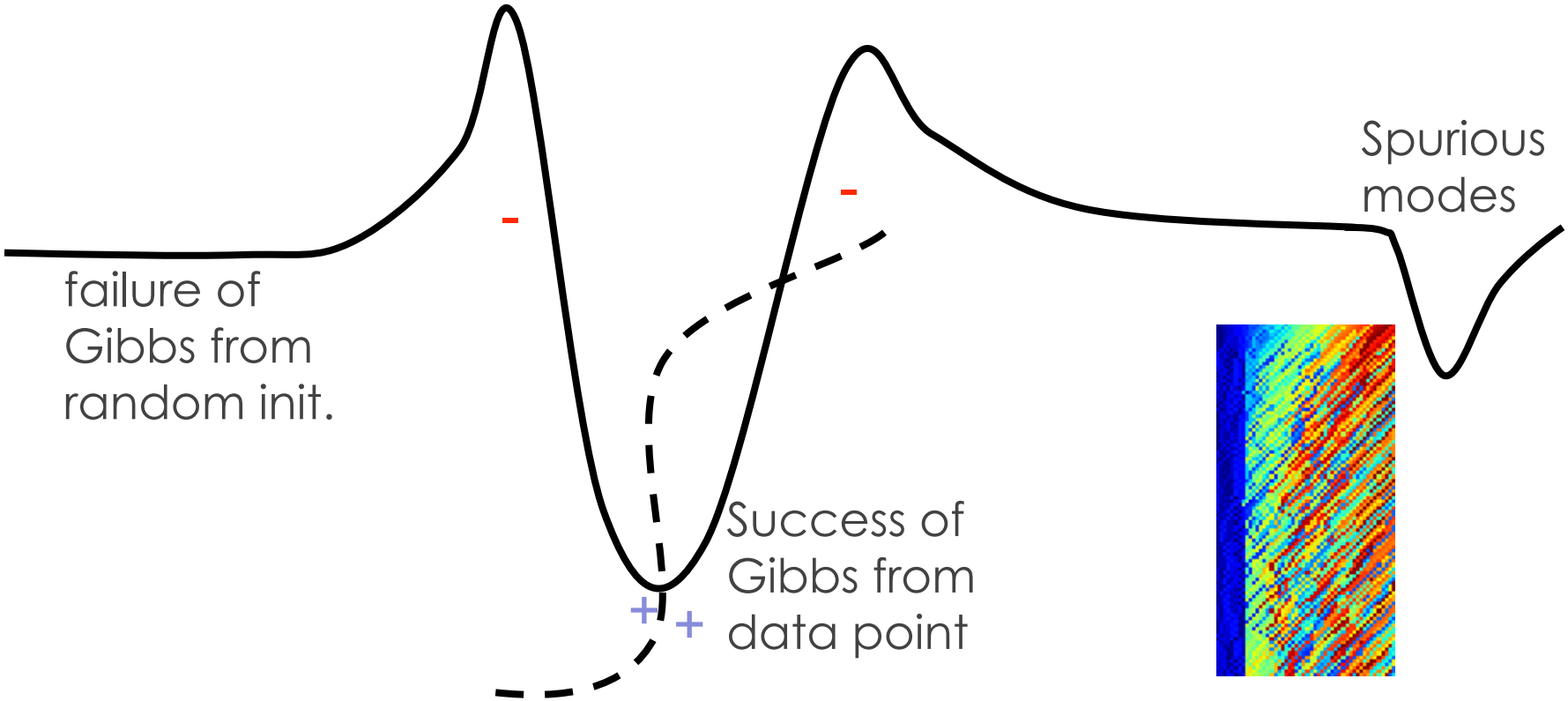


Chains from random state

Chains from real digits

Hypothesis to Explain Poor Mixing in CD-Trained RBMs (G. Desjardins)

- CD-training creates wells (or valleys) at (or between) data points and barriers around them



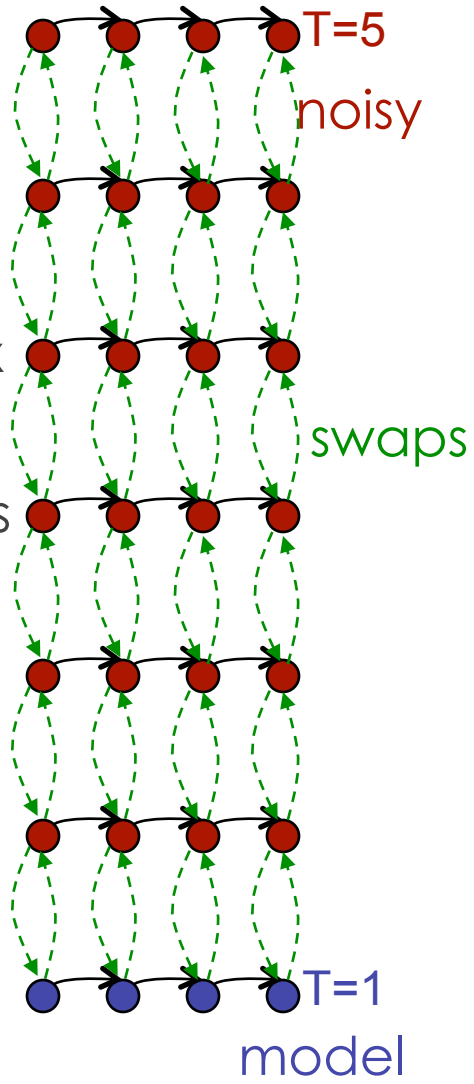
Parallel Tempered MCMC

(Desjardins, Courville, Bengio, Vincent, Delalleau: AISTATS 2010)

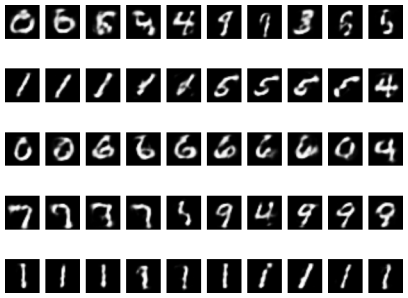
- Higher temperature = more noise = smoothed energy landscape: can escape modes faster, mix better
- Consider multiple chains at different temperatures and reversible swaps between adjacent chains
- Model samples are from T=1
- Swap prob = $[\ p_k(x_{k+1})\ p_{k+1}(x_k)\ ]\ /\ [\ p_k(x_k)\ p_{k+1}(x_{k+1})\ ]$

  $= \exp((b_k - b_{k+1})(E(x_k) - E(x_{k+1})))$

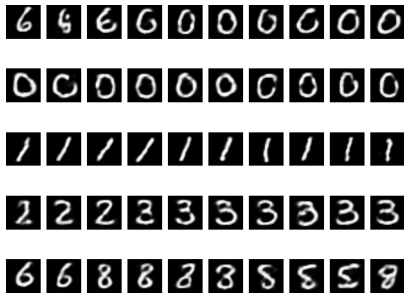
  where b_k = inv. temp., $E(x)$ = energy of x.
- PCD learning with PTMCMC chain



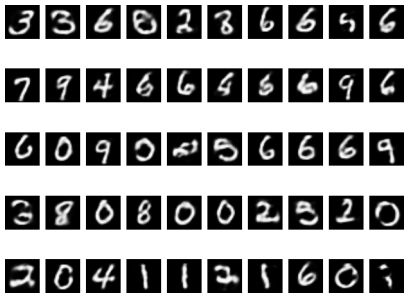
Parallel Tempered MCMC



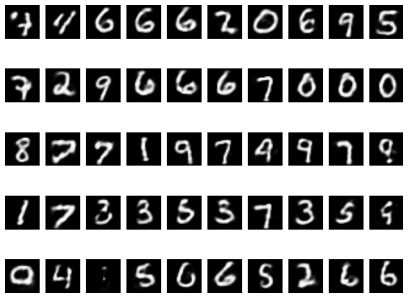
PCD-Gibbs during training.
Mixes well



PCD-Gibbs after training.
Does not mix



TMCMC during training.
Mixes well



TMCMC after training.
Mixes well

Sample Generation Procedure			
Training Procedure	TMCMC	Gibbs (random start)	Gibbs (test start)
TMCMC	208	211	210
FPCD	180	175	176
PCD	80	128	139
CD	-1979	-854	37

Indirect Sampling Likelihood

Unlearning for Better Mixing

(with Breuleux, AISTATS breaking news)

- Fast Persistent CD (FPCD) and Herding exploit impressively faster mixing achieved when parameters change quickly (large learning rate) while sampling

- FPCD and Herding decrease probability of just-visited states, but change other probabilities through RBM parametrization. FPCD is more conservative by making those changes temporary with exponentially fast decay.

Unlearning for Better Mixing

- All chains with the same leading eigenvector go to the same distribution but may converge at very different rates!
- Consider an MCMC with transition probability matrix A with asymptotic distribution p = A p.
- Want to spend less time in modes: reduce probability of staying in same state by λ:

$$\text{Let } \hat{A} = (A - \lambda I)/(1-\lambda)$$

- Proposition:

$$\hat{A}\, p = p,$$

i.e. converge to the same distribution.

Rates-FPCD Sampler

- Estimate sufficient statistics (s_i s_j) during training
  (Rates idea from Welling's Herding)
- During sampling, change weights to move away from current state (FPCD idea)
- But update moves towards preserving sufficient statistic

Gibbs vs Rates-Herding vs Rates-FPCD

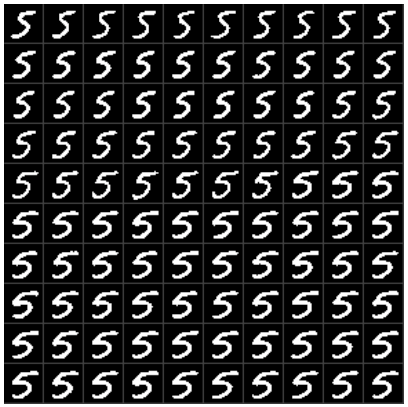
Rates-FPCD

Mixes very well!

Rates-Herding

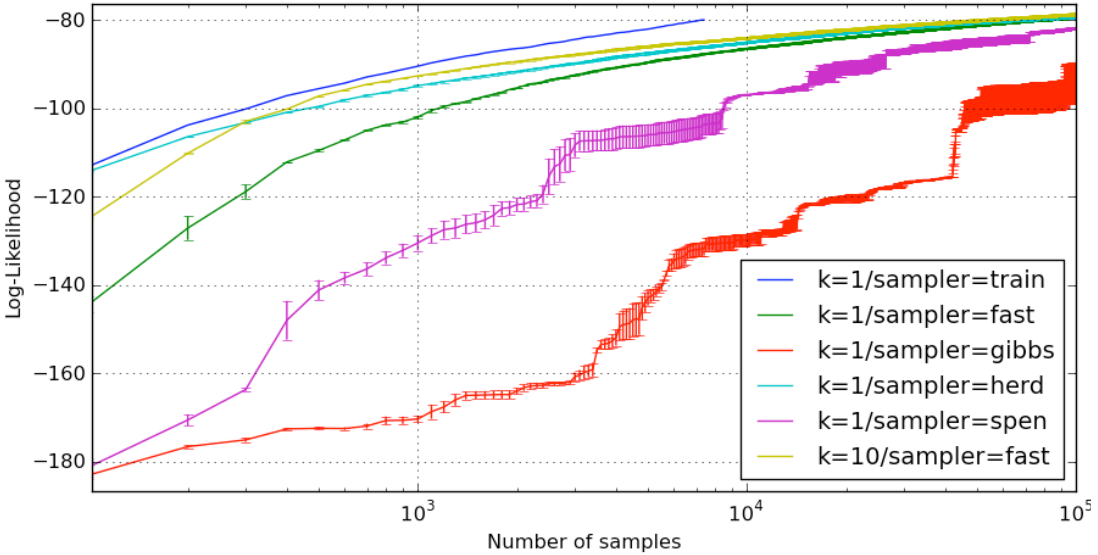
Very surprising:
Rates-herding = *deterministic dynamical system*
→ very good at sampling from a trained RBM!

Mixes very slowly!

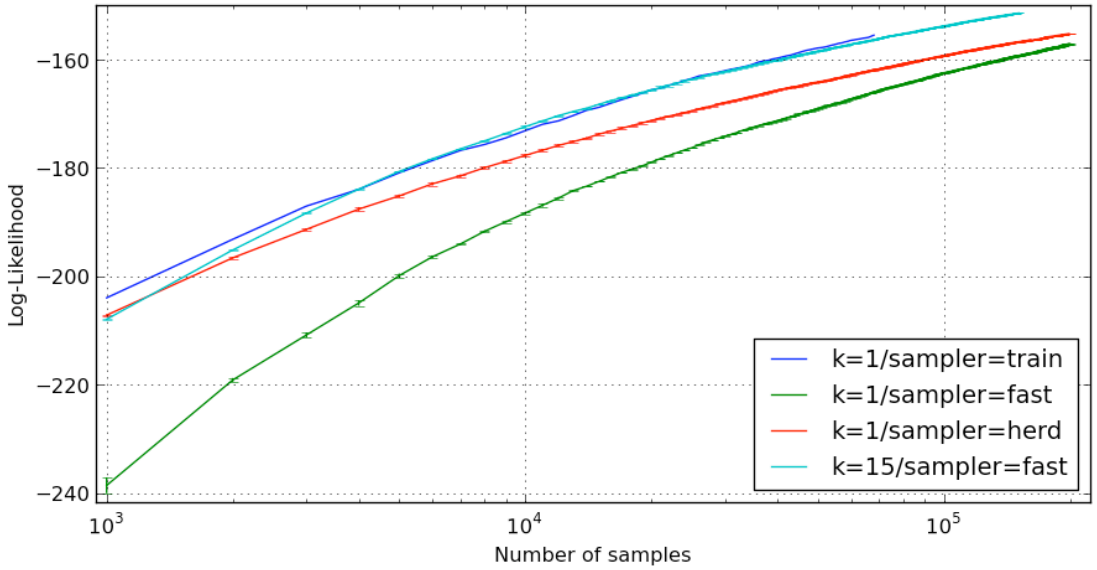


Gibbs

Gibbs vs Rates-Herding vs Rates-FPCD



USPS.
.



MNIST.
Gibbs too poor to show.

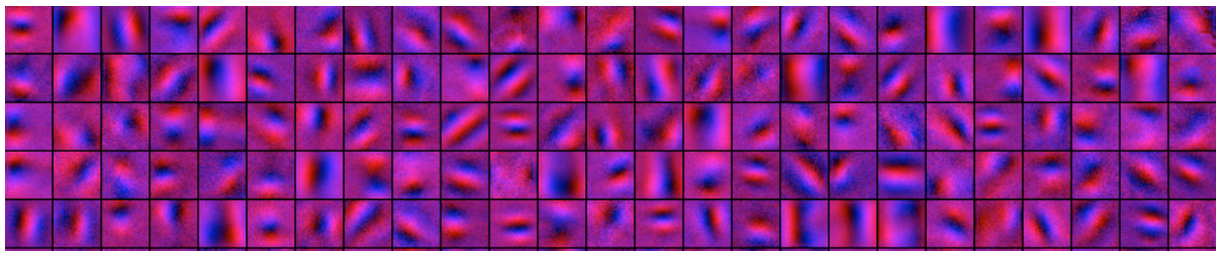
Rates-FPCD converges faster.

Why are Classifiers Obtained from Pre-Trained Deep Nets Working so Well?

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

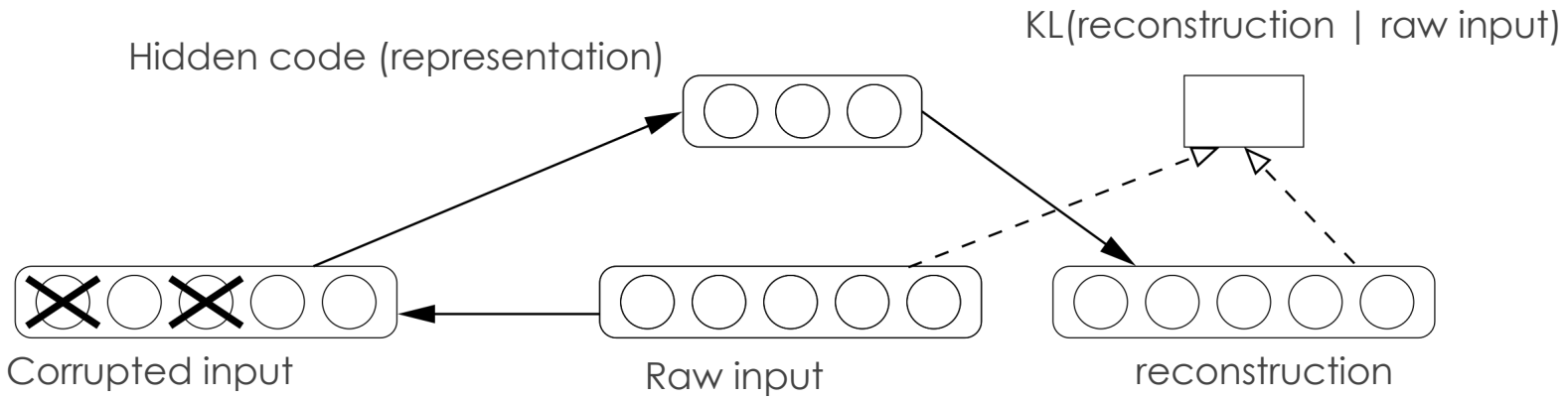
Replacing RBMs by Other Layer-Local Unsupervised Learning

- Auto-encoders (Bengio et al, NIPS'06)
- Sparse coding (Lee et al NIPS'06, Bradley & Bagnell NIPS'09)
- Sparse Predictive Decomposition (Ranzato et al, NIPS'06)
- Kernel PCA (Erhan 2008)
- **Denoising auto-encoders (Vincent et al, ICML'08), simpler than RBMs and matches or beats RBMs**
- Unsupervised embedding (Weston et al, ICML'08)
- Slow features (Mohabi et al, ICML'09, Bergstra & Bengio NIPS'09)



Denoising Auto-Encoder

- Stacked, unsupervised pre-training (Vincent et al, ICML'08)
- Corrupt the input (e.g. set 25% of randomly chosen inputs to 0)
- Reconstruct the uncorrupted input
- Use uncorrupted encoding as input to next level



Denoising Auto-Encoders vs RBMs

- DAE easier to grasp & teach than RBMs
- DAEs worked as well or better than RBMs as feature learners
- DAEs training criterion lower bounds log-lik of generative models, but their generative ability has not been well studied (for now I would still use RBMs to generate data, e.g. for animations)
- DAEs more flexible (can choose any parametrization for encoders and decoders)

Level-Local Learning is Important

- Initializing each layer of an unsupervised deep Boltzmann machine as an RBM helps a lot (Salakhutdinov & Hinton 2009).
- Initializing each layer of a supervised neural network as an RBM or denoising 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: initialization matters!

Why is ***Unsupervised*** Pre-Training Working So Well?

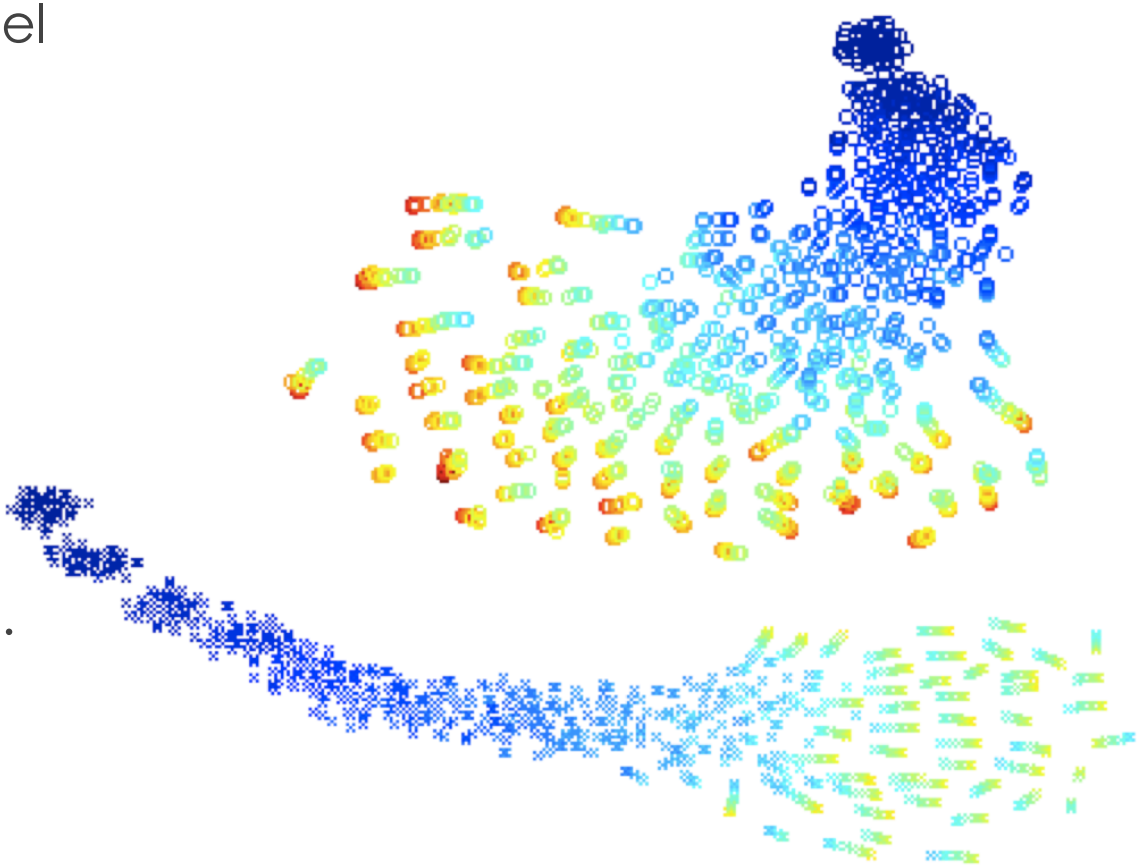
(with Erhan, Courville, Manzagol, Vincent, Bengio: JMLR, 2010)

- 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 supervised training error
  - Can reach lower local minimum otherwise not achievable by random initialization

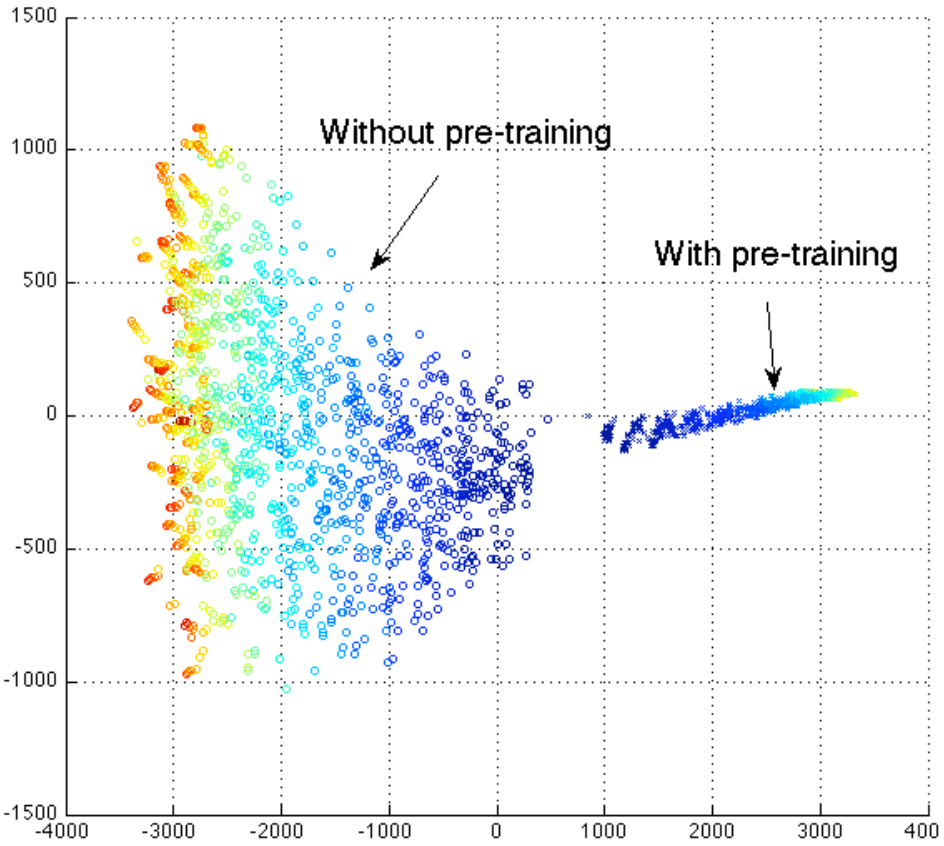
Learning Trajectories in Function Space

- 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



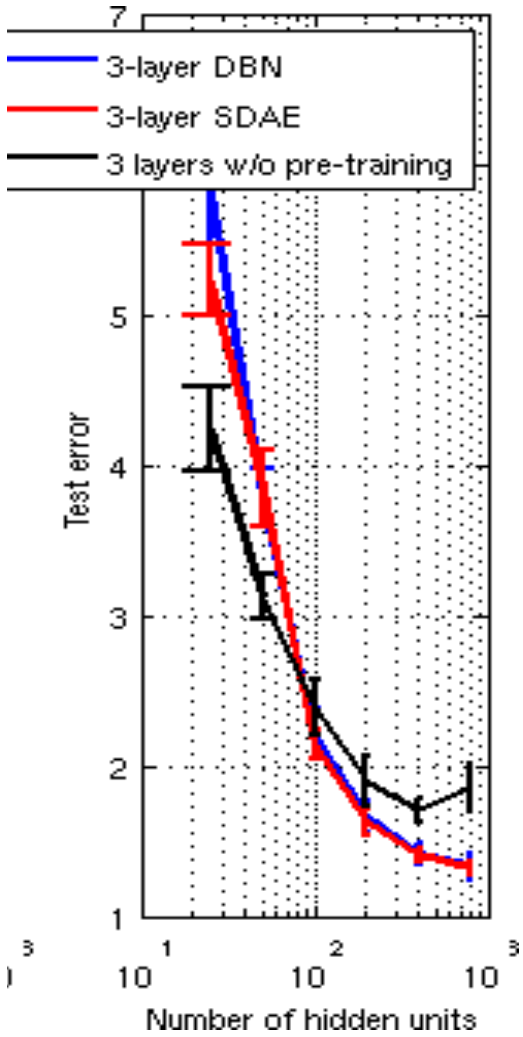
Visualization in Function Space

- Using ISOMAP instead of t-SNE, preserve distances
- Pre-training: small volume compared to without.



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



Unsupervised Disentangling of Factors of Variation

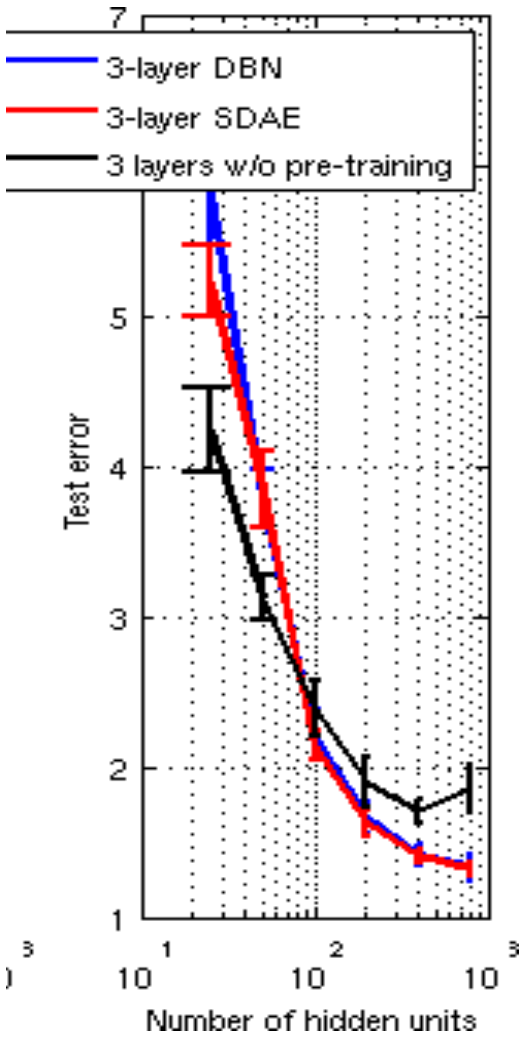
- **Explanatory theory**:

(Denoising Auto-Encoder)

- Stacked RBMs & DAE disentangle factors of variation in *P(x)* (Goodfellow et al, NIPS'09)
- Most salient factors are unrelated to *y*, but some factors are highly predictive of *y*

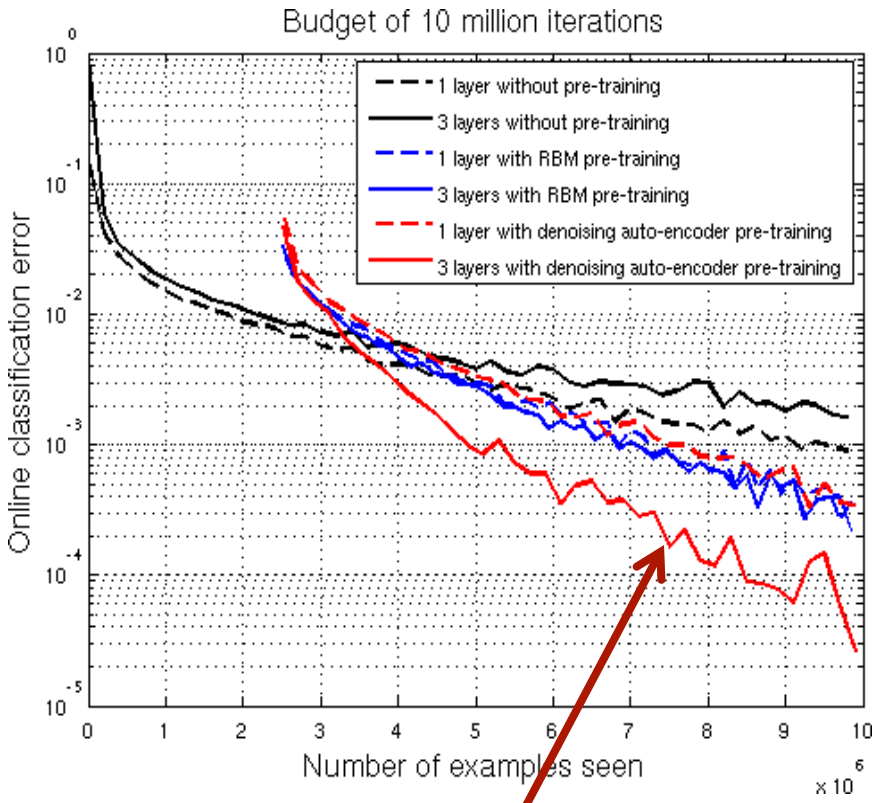
→RBMs with too few units learn features worse at predicting *y* than randomly initialized networks

→RBMs with many hidden units are much more predictive of *y*

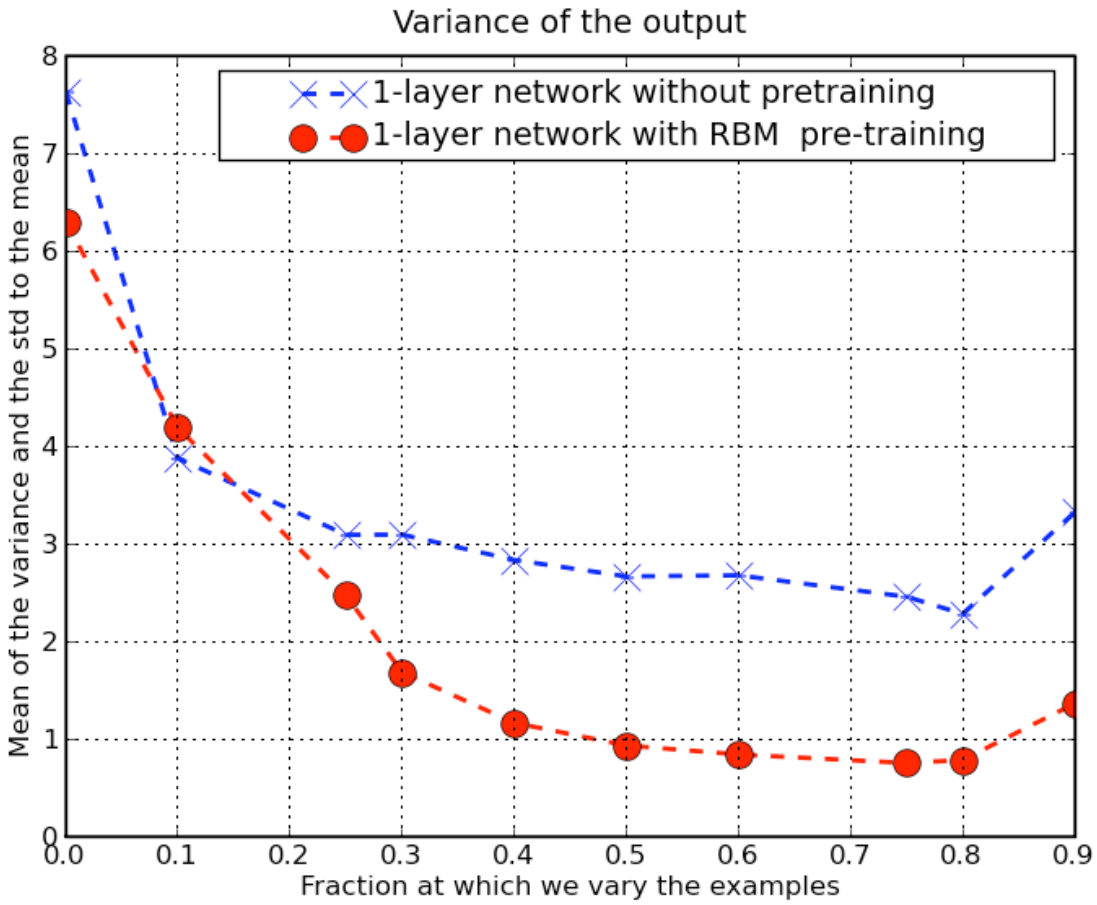


Better Optimization of Online Error

- 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

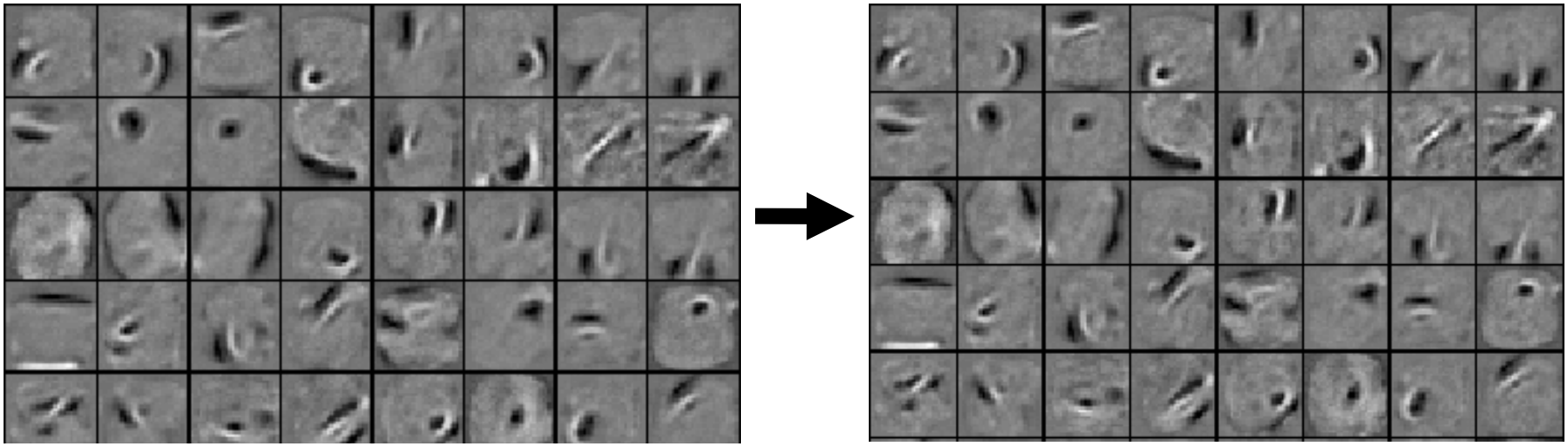


Initial Examples Matter More (critical period?)



Vary 10% of the training set at the beginning, middle, or end of the online sequence. Measure the effect on learned function.

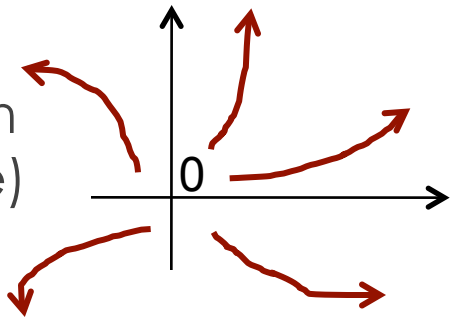
Learning Dynamics of Deep Nets



Before fine-tuning

After fine-tuning

- As weights become larger, get trapped in basin of attraction (sign does not change)
- Critical period. Initialization matters.



The Credit Assignment Problem

- Even with the correct gradient, lower layers (far from the prediction, close to input) are the most difficult to train
- Lower layers benefit most from unsupervised pre-training
  - Local unsupervised signal = extract / disentangle factors
- Credit assignment / error information not flowing easily?
- Related to difficulty of credit assignment through time?

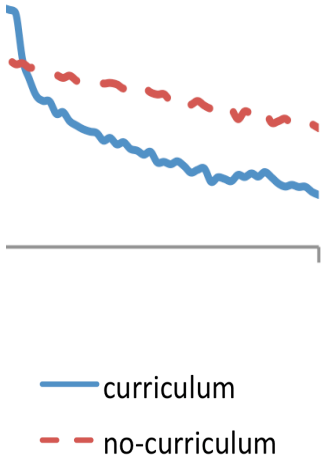
Order & Selection of Examples Matters

(with Louradour, Collobert & Weston, ICML'2009)

- 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



Understanding the difficulty of training deep feedforward neural networks

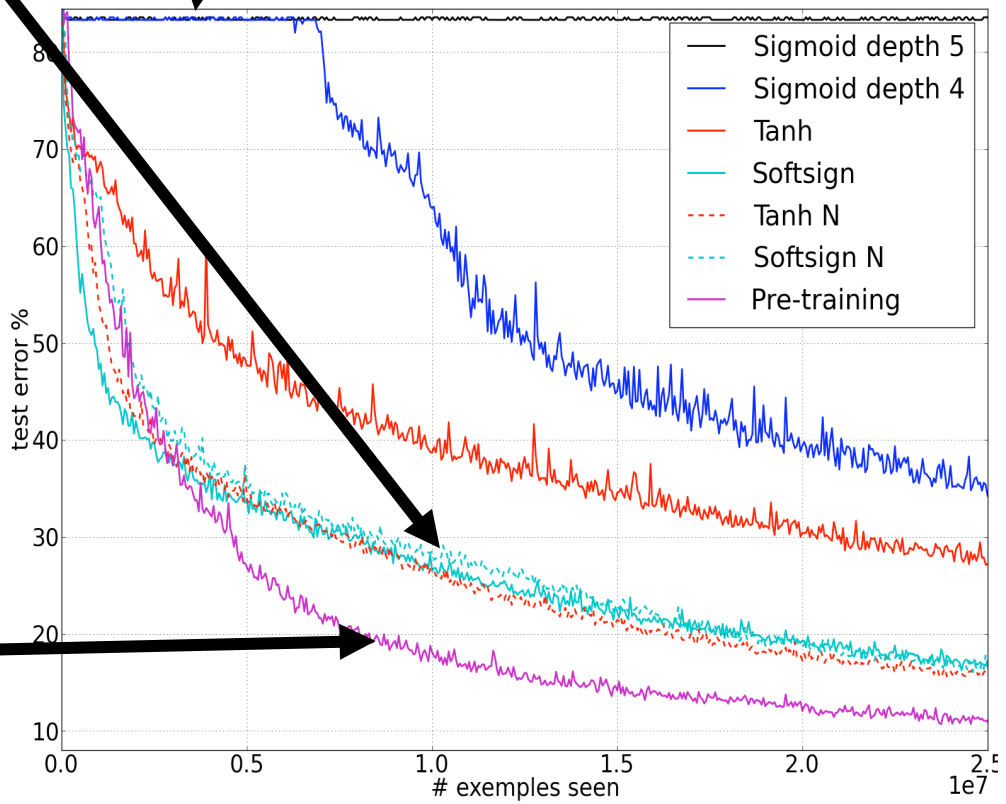
Study the activations and gradients

- wrt depth
- as training progresses
- for different initializations
- for different activation non-linearities

Empirical Analyses of Gradient Propagation (Glorot & Bengio, AISTATS 2010)

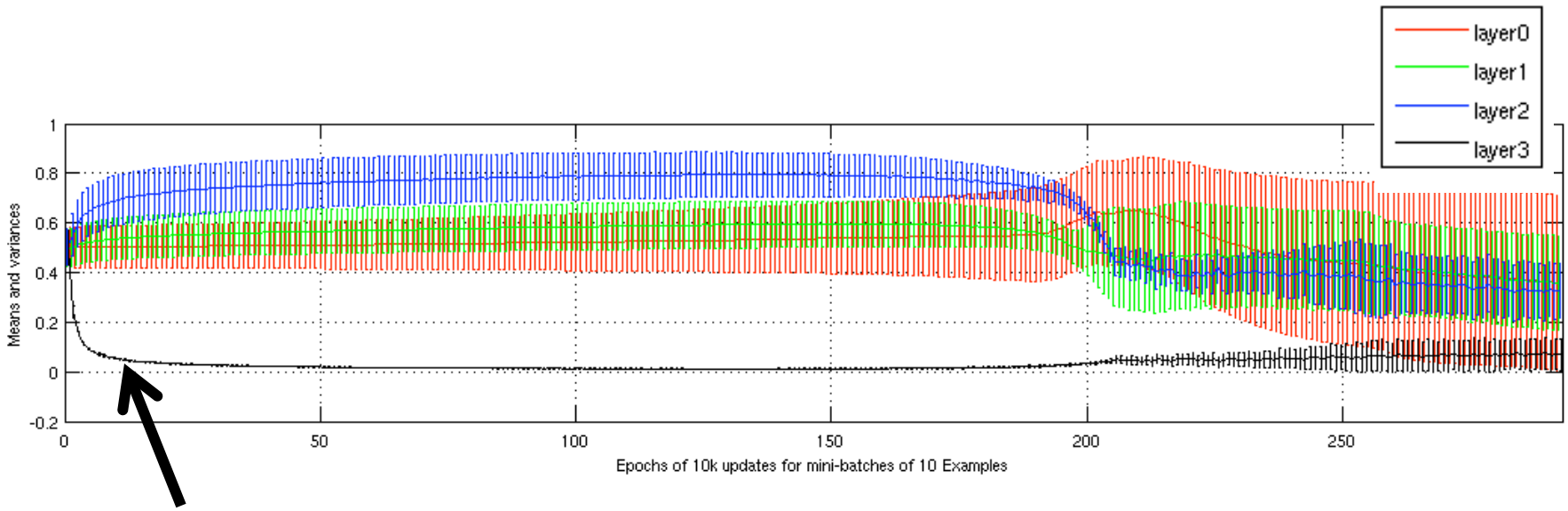
- Understood conditions under which top hidden layer with sigmoids goes into initial saturation
- No saturation with softer (non-exp.) non-linearity, e.g. softsign(x)=**x/(1+|x|)**
- Effect of initialization, too small or too large, on activations and gradients
- Want Jacobian around 1: variance preserving

Unsupervised pre-training: Automatically variance-preserving!



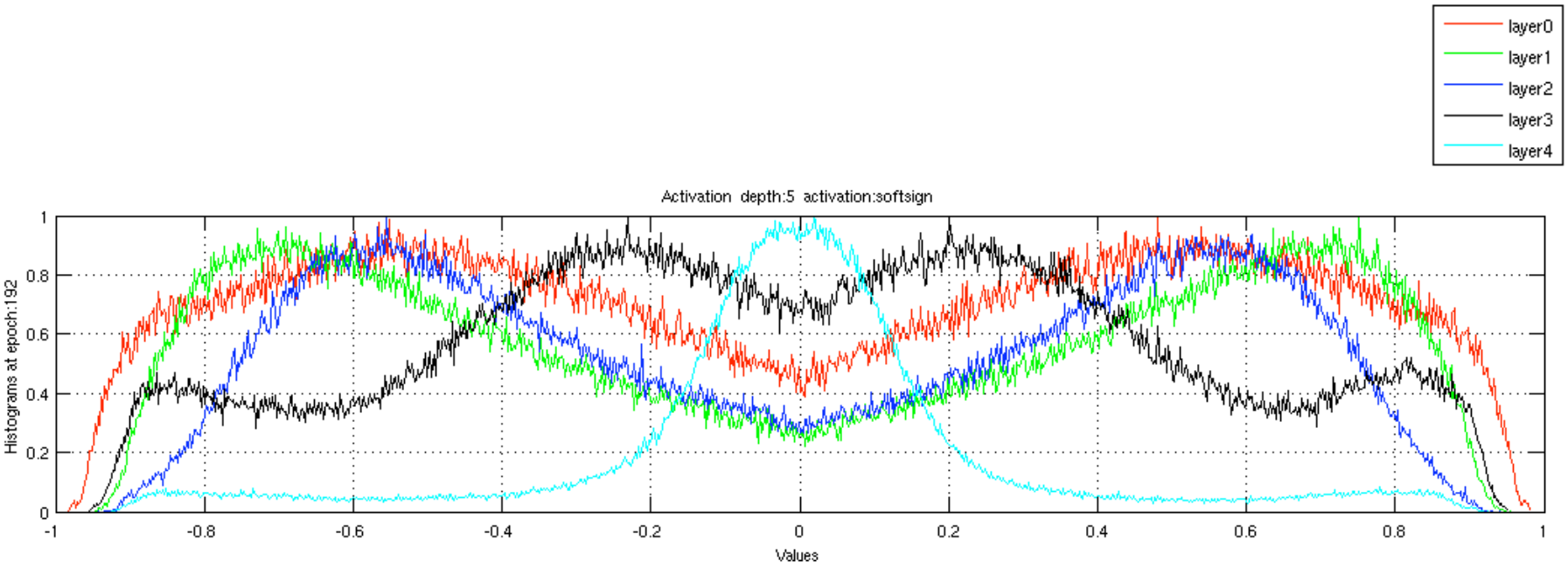
Sigmoids Saturate and then UnSaturate!

- Top hidden layer quickly (50k updates) saturates, error stuck on plateau, then slowly desaturates as lower layers learn something more sensible.



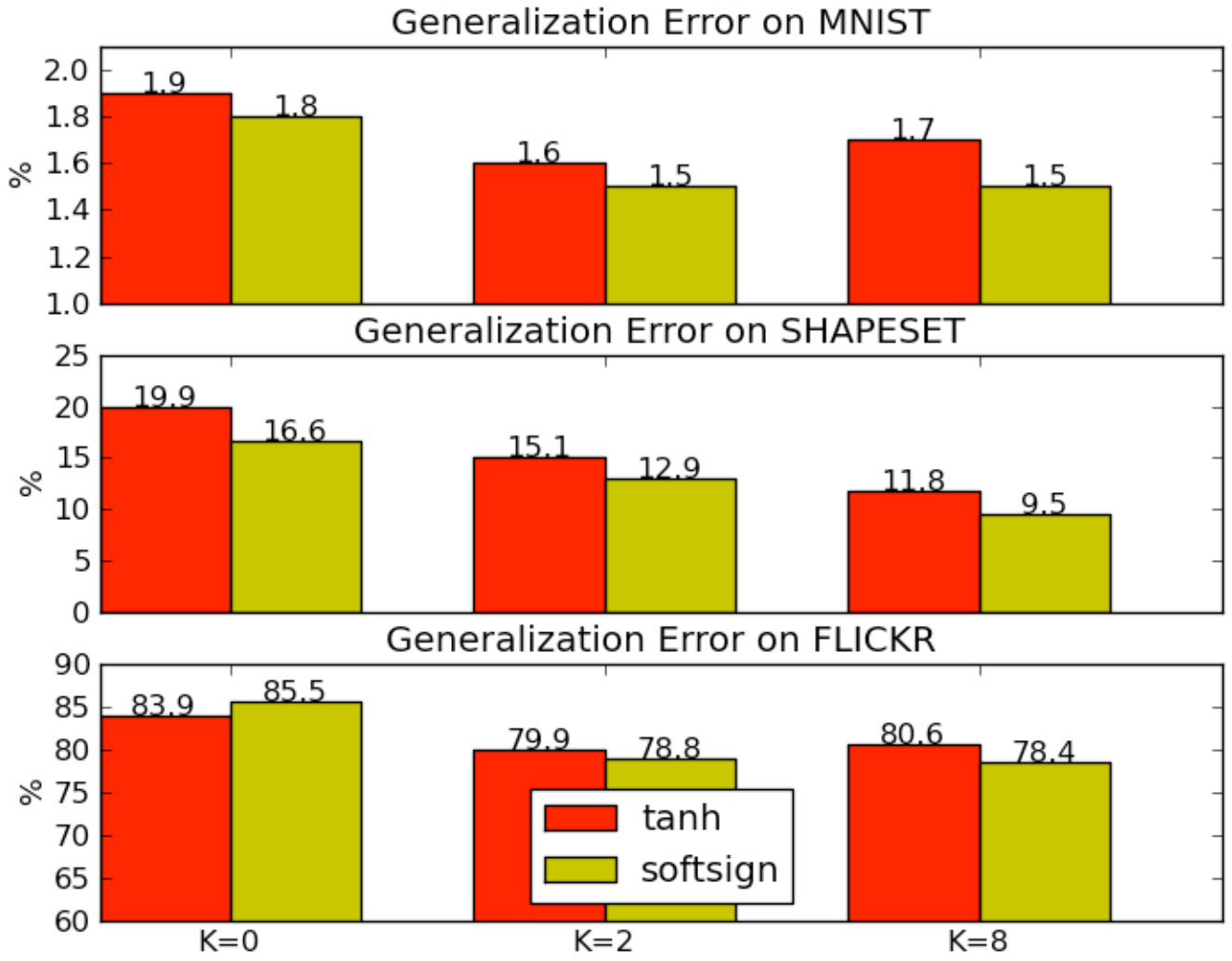
Most initial hidden units almost uncorrelated with target: want to kill off output W'h

No Saturation with Softsign = x/(1+|x|)



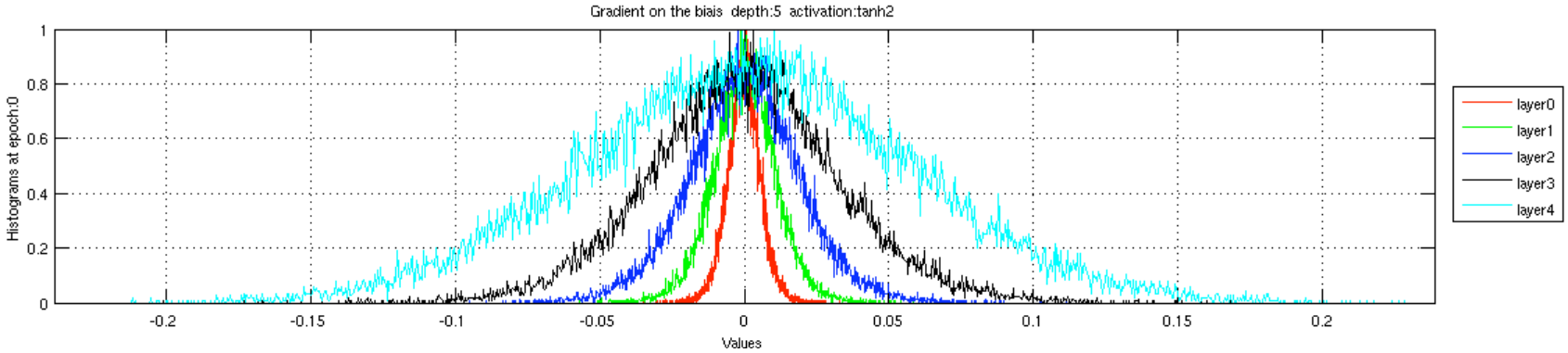
Softsign often beats tanh & sigmoid

- Deep supervised dense nets

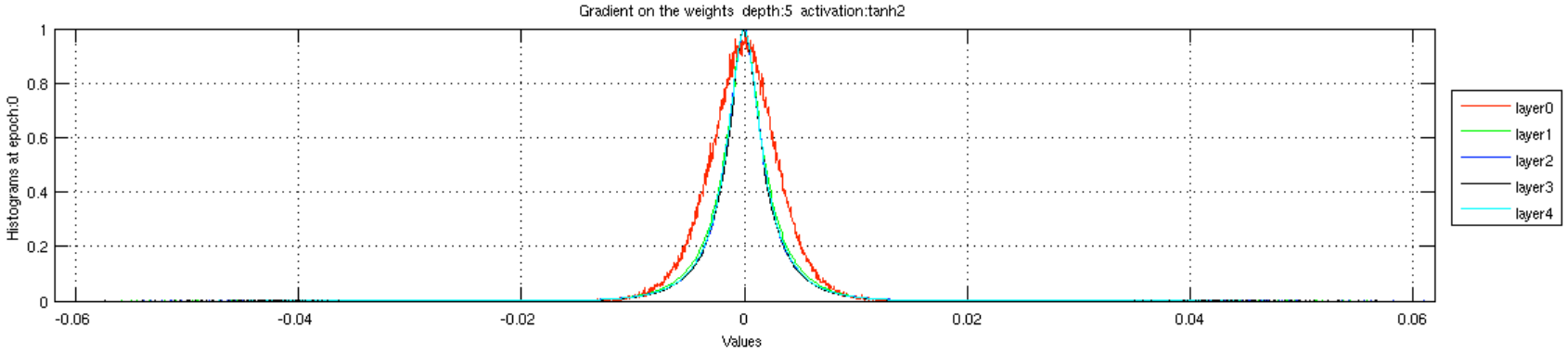


Initial layer-wise gradients vanish

- First observed by D. Bradley (CMU thesis, 2009)
- With 1/sqrt(fan_in) initialization, tanh units.

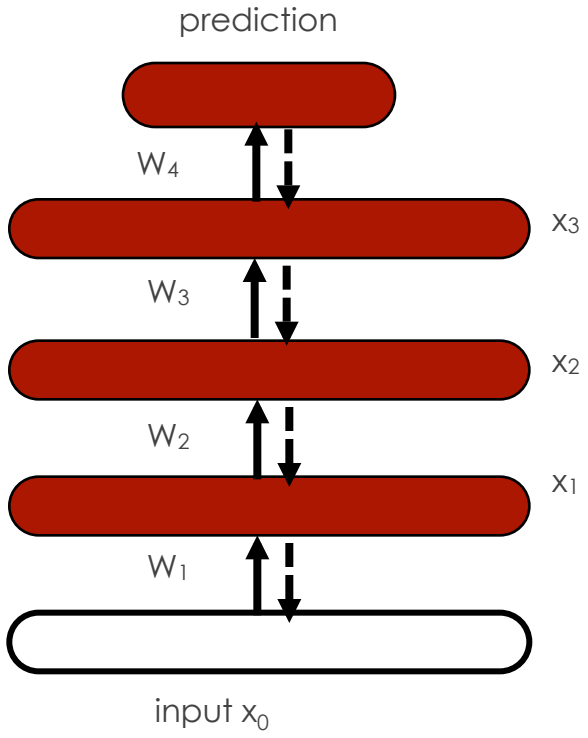


- But not weight gradients! Var(dC/dW) at level k ~ a^k a^{d-k} = $\mathbf{a^d}$

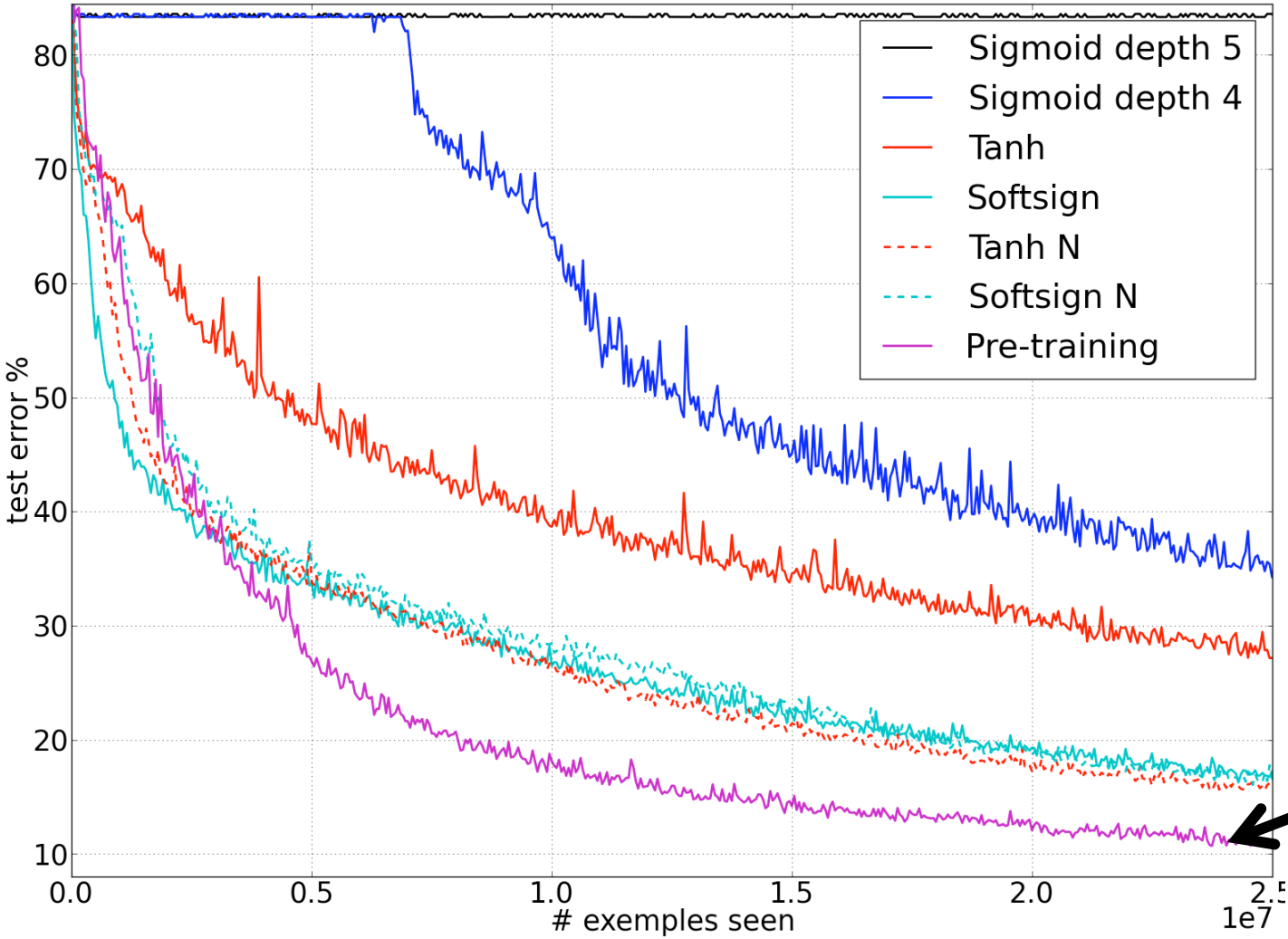


Effect of Small Jacobian

- Activations become smaller going from input to output by factor ***a***
- Activation gradients become smaller going from target towards input by factor ***a***
- Product $dC/dWi = dC/dx_i * x_{i-1}$ is constant in linear(ized) case, proportional to $\boldsymbol{a}^{depth}$



Normalized Initialization with Variance-Preserving Jacobians

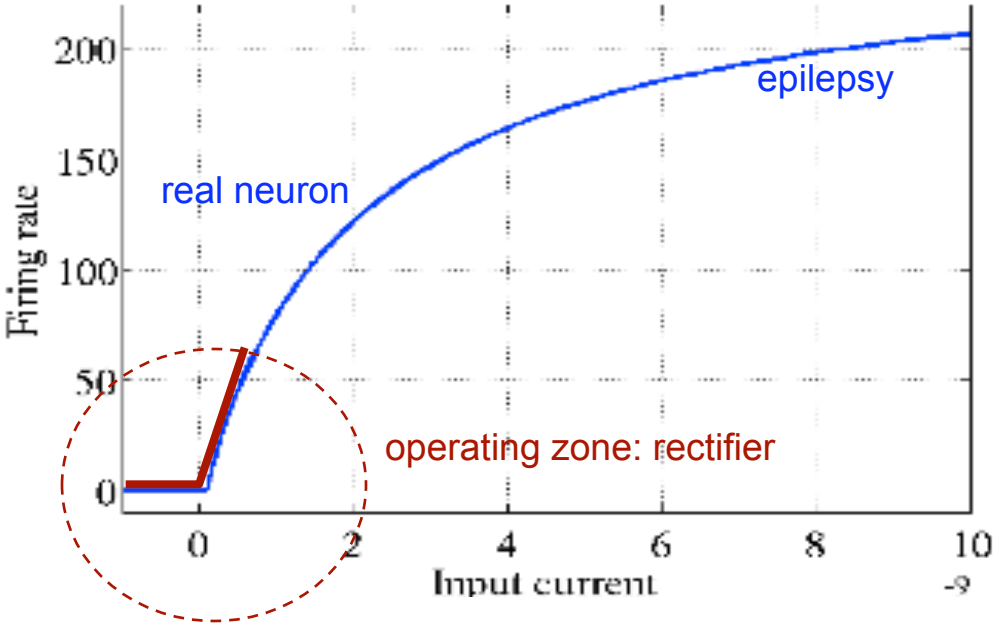


Shapeset 2x3 data

Unsupervised pre-training: Automatically variance-preserving!

Rectifier Neurons & Sparsity: Motivations (Glorot & Bengio, Learning Workshop 2010)

- Biologically more plausible activation function
- Leads to sparse outputs: real neurons activation is sparse ~1% – 4%
- Motivations:
  - Information disentangling
  - Variable-size representation
  - Easier separability
  - Humans define concepts sparsely (from few others)
  - Regularizer

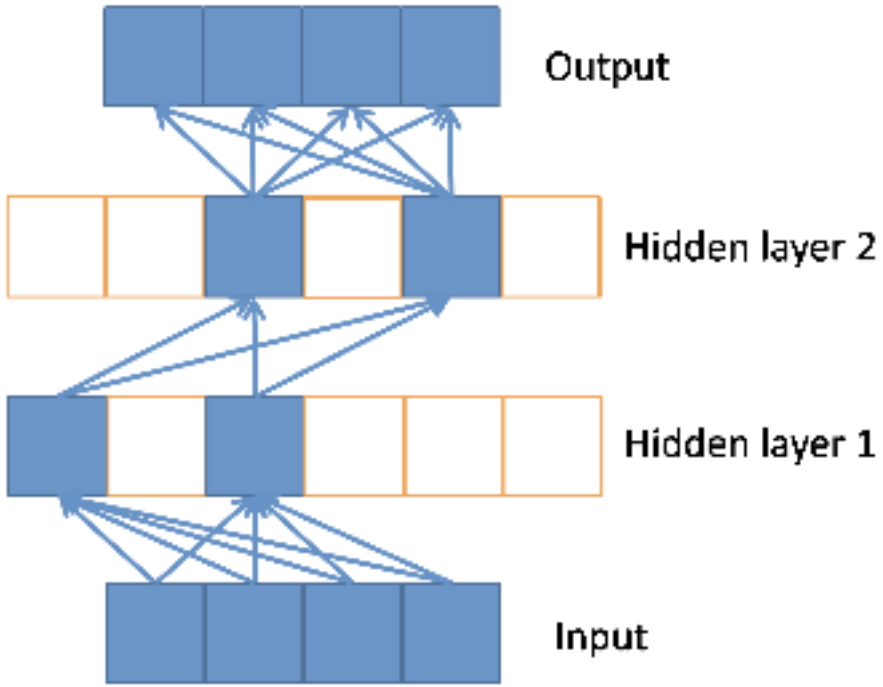


Few bits of information

Many bits of information

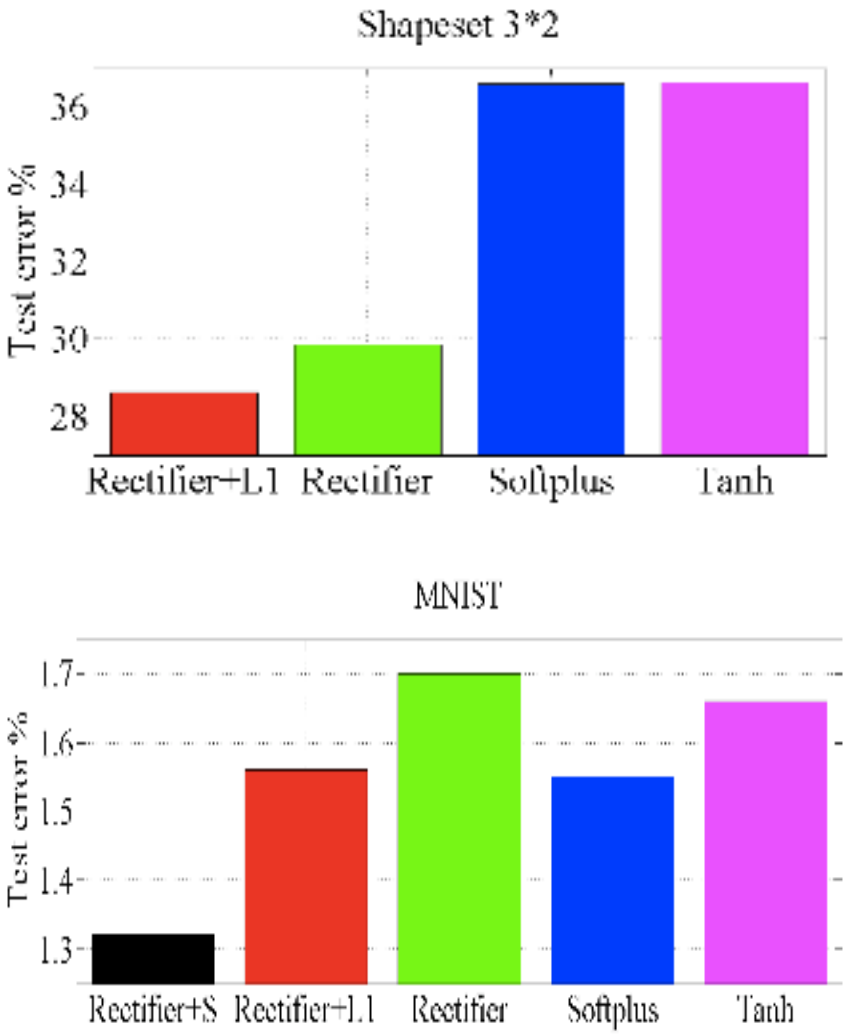
Gradient flow in sparse nets

- Non-linearity = paths selection
- Linear given paths
- Exponential number of possible paths / linear models
- Gradients flow well through selected paths
- Does the hard non-linearity hurt training? Apparently not!



Sparse Rectifier Nets: Results

- 3 hidden layers, purely supervised
- Beats soft rectifier & tanh
- Achieves **80%-95% sparsity**
- With layer-wise normalization, achieves ***best result ever*** *without convolution, pre-training or deformations*: **1.3%**
- Also helps stacked denoising auto-encoders



Deep Learning Research: AI is Back

- Biological inspiration – comp. neuro. (Cox, DiCarlo, Pinto @ MIT)
- Not just neural nets: **machine learning towards AI**
  - Object grammars, parts models (McAllester @ TTI-C)
  - Markov logic networks (Domingos @ U. Washington)
  - Graphical models with many levels of latent var.
  - Hierarchical / stacking sparse coding (Bach @ ENS)
- DARPA Deep Learning program (Ng + Koller, LeCun + Bengio + Fergus + Collobert)
- See http://deeplearning.net

http://deeplearning.net

http://deeplearning.net/software/theano : numpy → GPU

Conclusions

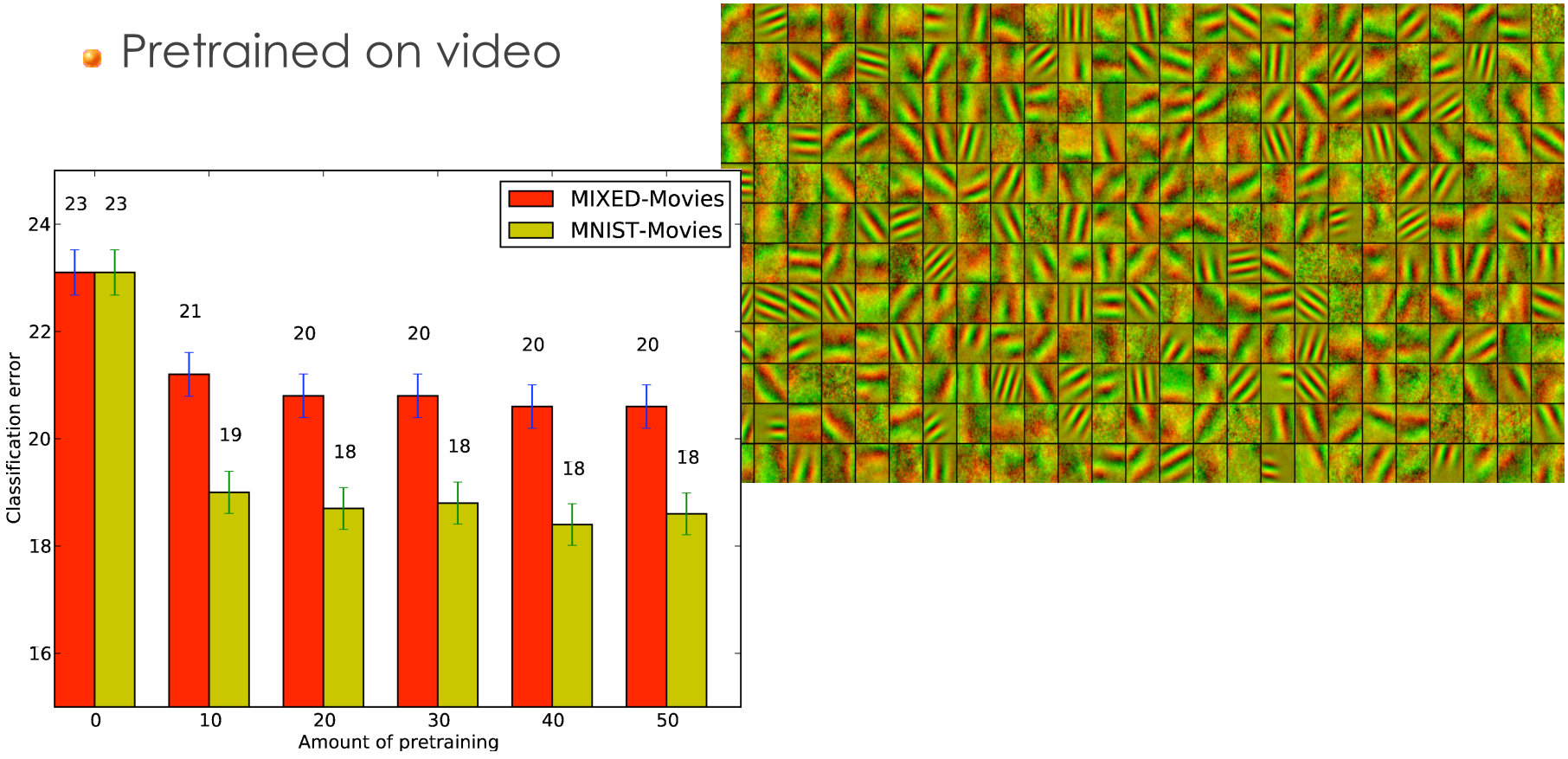
- Deep architectures: potential exponential gain in representation thanks to composition of abstractions
- Unsupervised pre-training helps a lot: RBMs, auto-encoders,…
- CD and Gibbs for RBMs have weaknesses
- Improved training and sampling algorithms for RBMs
- Why does unsupervised pre-training work? Regularization and optimization effects intertwined
- Why is it difficult to train deep neural nets? Principled ways to initialize and changing the non-linearity can help quite a bit!
- A more biologically plausible non-linearity brings sparsity and better faster training.

Thank you for your attention!

- Questions?
- Comments?

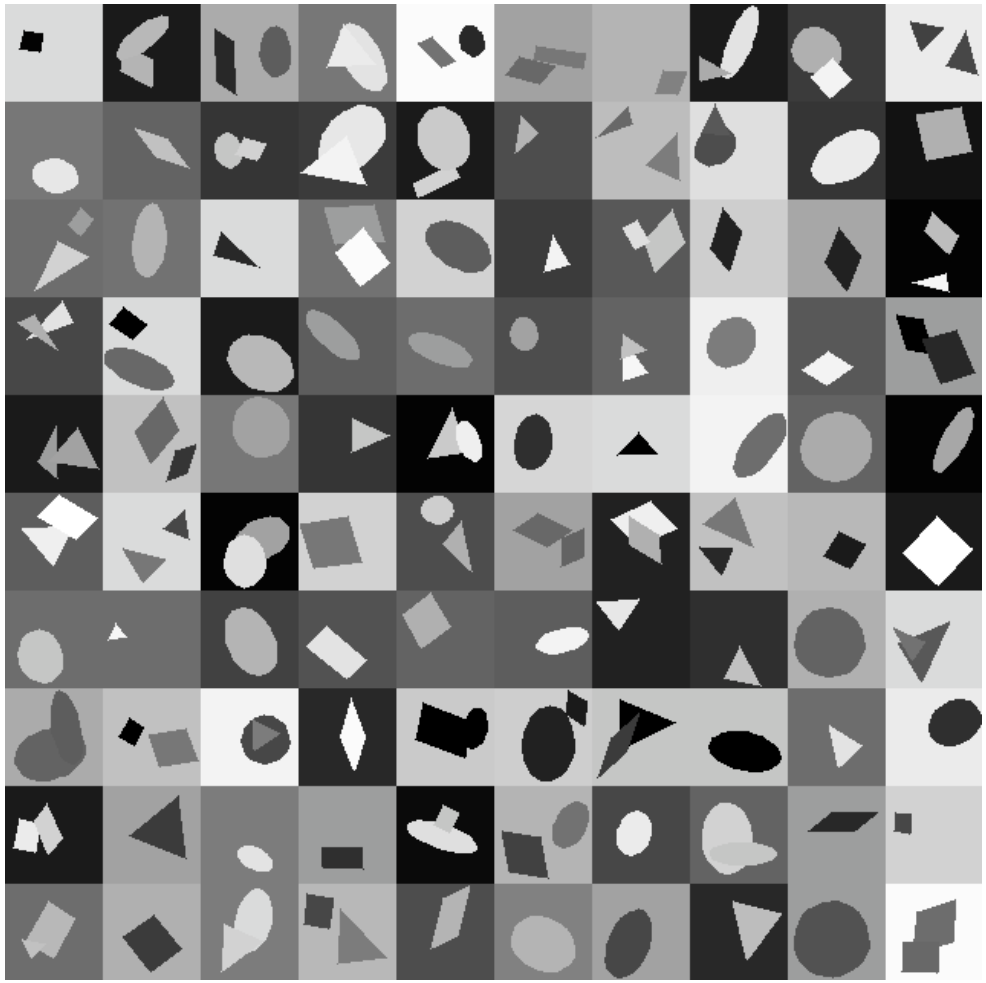
Learning Slow Features

- NIPS'2009 paper with James Bergstra
- Combine efficient computation of slowness with quadratic (complex cell, sum of squared filters)
- Pretrained on video

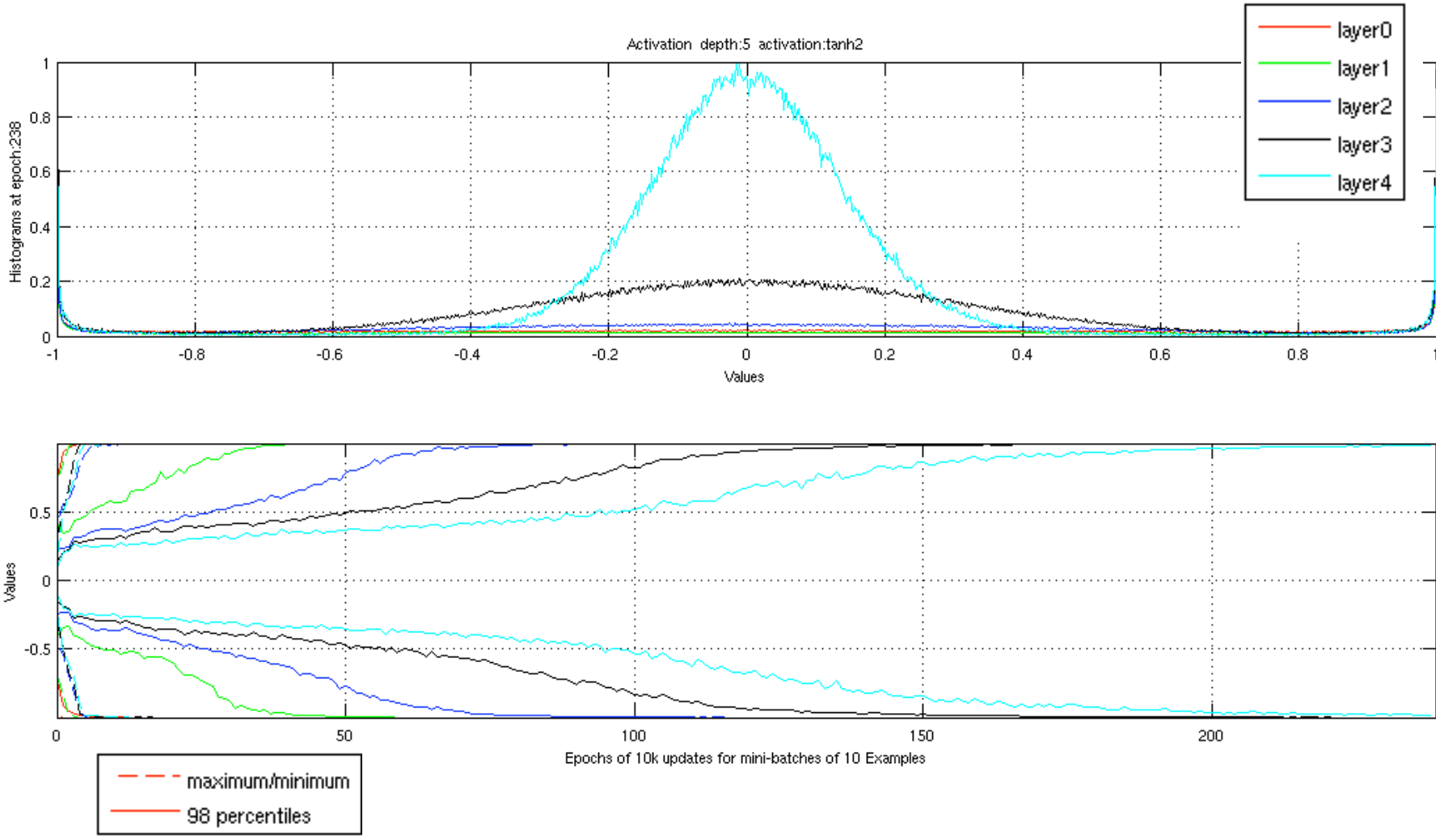


Online Multi-Shape Detection

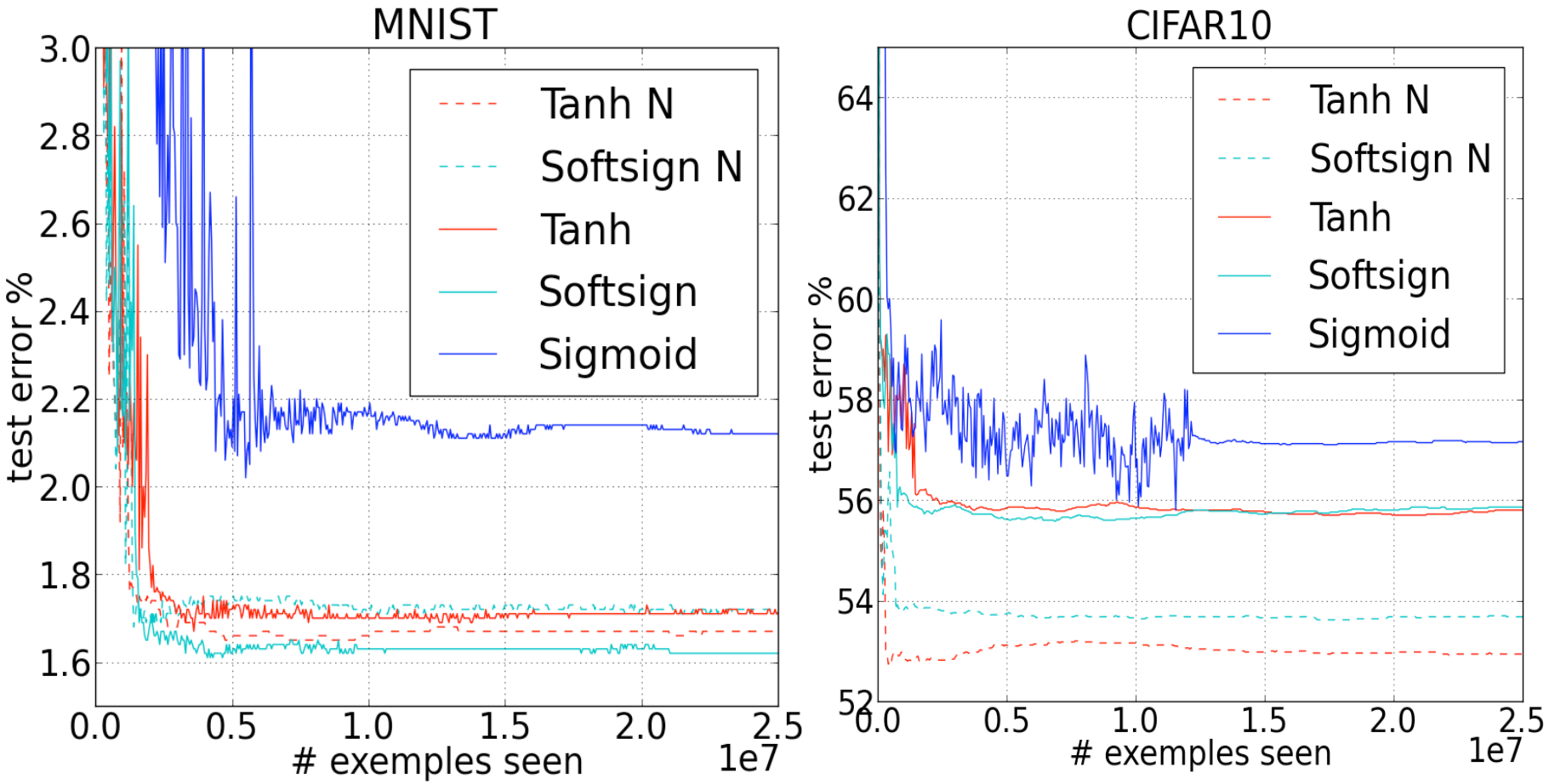
- 3 shape categories
- All 9 combinations of 1 or 2 shapes in an image = 9 target classes
- 32x32 inputs
- 1000 units/layer
- 1 to 5 layers
- All hyper-parameters separately optimized for each case shown



Gradual Saturation with Tanh

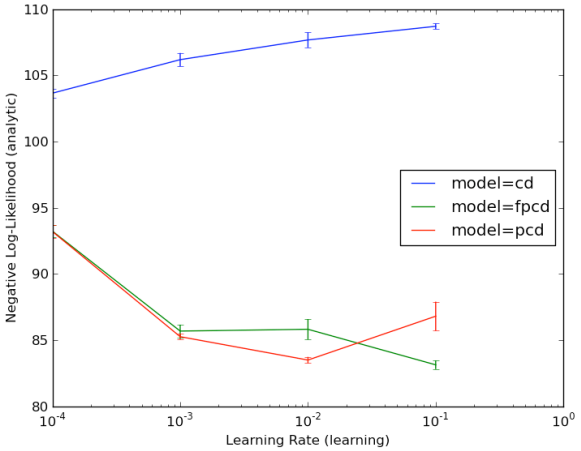


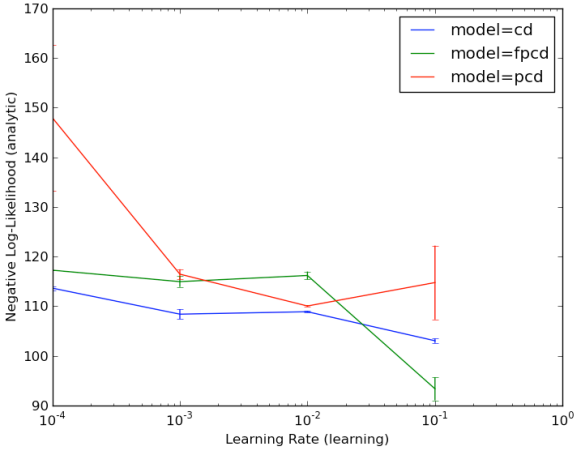
Normalized Initialization with Variance-Preserving Jacobians



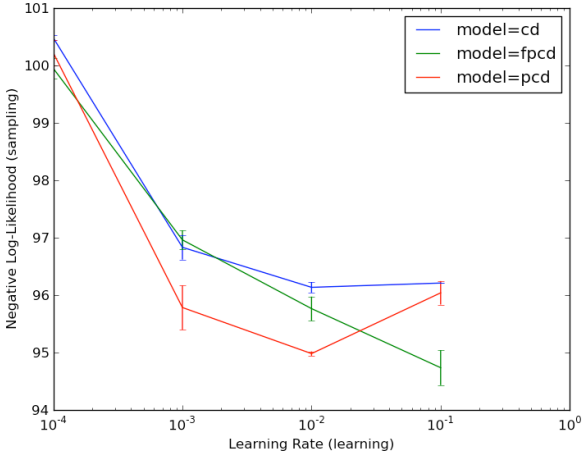
ISL vs Analytic Likelihood

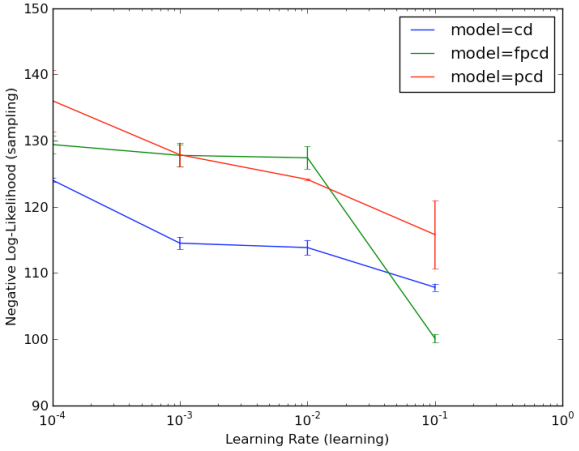
Analytic





ISL





0/1 units

-1/1 units

Note CD discrepency because sampler does not reflect model