

# Deep Learning of Representations

Canadian AI 2014 Tutorial  
Montreal, Canada

Yoshua Bengio

May 6, 2014



# Outline of the Tutorial

1. Representation Learning, motivations
2. Basic Algorithms
3. Practical Considerations and Applications
4. Challenges & Ongoing Work

Upcoming book: “Deep Learning” <http://www.iro.umontreal.ca/~bengioy/dlbook/>

# Ultimate Goals

- AI
- Needs **knowledge**
- Needs **learning**  
(involves priors + *optimization/search*)
- Needs **generalization**  
(guessing where probability mass concentrates)
- Needs ways to fight the curse of dimensionality  
(exponentially many configurations of the variables to consider)
- Needs disentangling the underlying explanatory factors  
(making sense of the data)

# What is Machine Learning?

Mathematical principles and computer algorithms exploiting data

- for extracting what is **GENERAL**
- so as to be able to say something meaningful **about new cases**
- to identify which **configurations** of variables are **plausible**
- to **generate** new plausible configurations
- to learn to **predict, classify, take decisions**

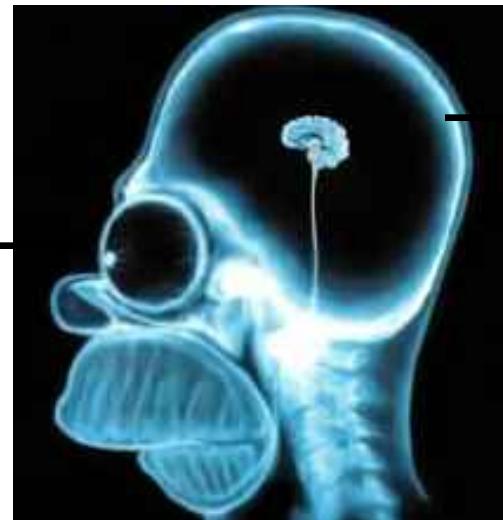
A hand is shown writing a complex mathematical equation on a chalkboard. The equation appears to be a polynomial or a system of equations involving variables like  $y$ ,  $x$ , and  $a$ . The handwriting is in white chalk on a dark background.

$$\begin{aligned} & 5a(y + z)^2 + 13y + 4z \\ & \frac{a^2}{29}(y + A)^{13} + \frac{2}{3}z^2 - 4(y + z) \end{aligned}$$

# Supervised Training Example

Input X

a



Output  
 $f(X)$

six

Target  
Y

two!



# Generalization vs Training Error

- Minimizing Training Error very well  
→ learning by heart
- Real objective: generalizing to new examples



- Mathematical guarantees about generalization if training error is small and predictor not too flexible (= priors)

# What is Generalizing?

- Capturing dependencies between random variables
- Spreading out the probability mass from the empirical distribution. Where???  
= making good guesses away from the training examples.
- Discovering underlying abstractions / explanatory factors

Part 1

# Intro & Motivations for Representation Learning and Deep Learning

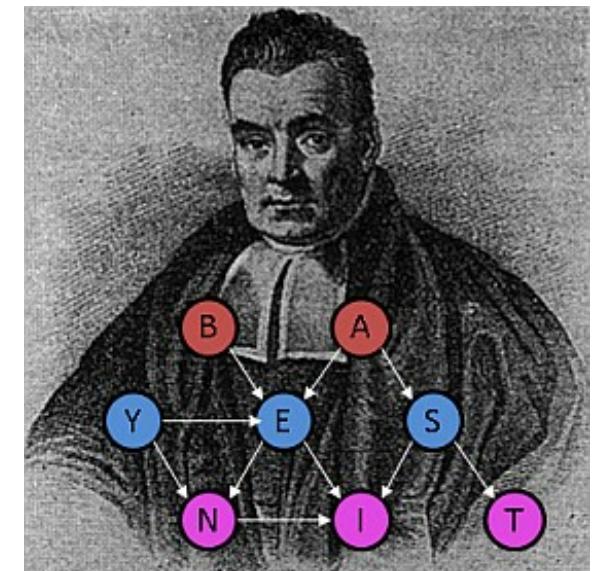
# Representation Learning

- Good **features** essential for successful ML: 90% of effort



- Handcrafting features vs learning them

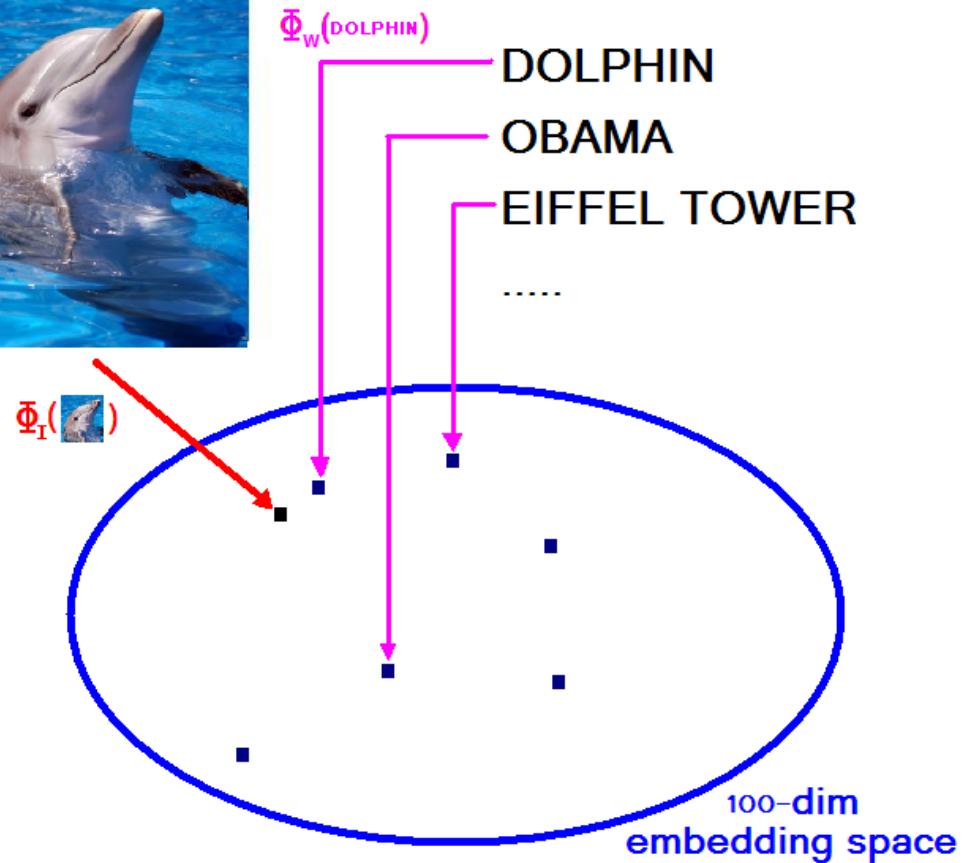
- Good representation?
- **guesses**  
the features / factors / causes



# Google Image Search: Different object types represented in the same space



Google:  
S. Bengio, J.  
Weston & N.  
Usunier  
(IJCAI 2011,  
NIPS'2010,  
JMLR 2010,  
MLJ 2010)



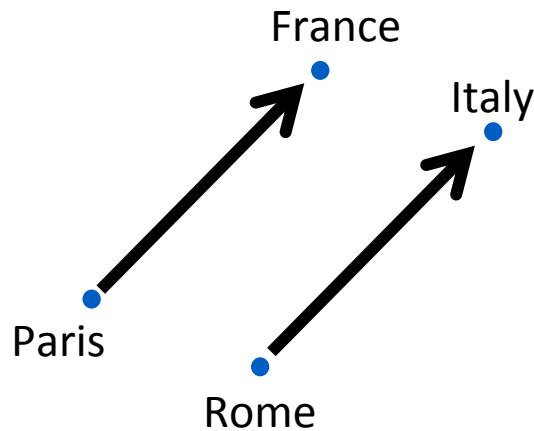
Learn  $\Phi_I(\cdot)$  and  $\Phi_w(\cdot)$  to optimize precision@k.

## Following up on (Bengio et al NIPS'2000) Neural word embeddings – visualization



## Analogical Representations for Free (Mikolov et al, ICLR 2013)

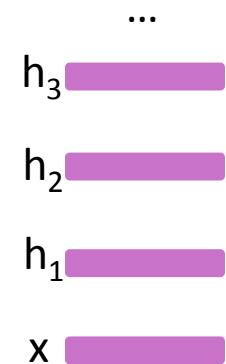
- Semantic relations appear as linear relationships in the space of learned representations
- King – Queen  $\approx$  Man – Woman
- Paris – France + Italy  $\approx$  Rome



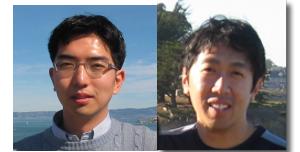
# Deep Representation Learning

Learn multiple levels of representation  
of increasing complexity/abstraction

- theory: exponential gain
- brains are deep
- cognition is compositional
- Better mixing (Bengio et al, ICML 2013)
- **They work! SOTA on industrial-scale AI tasks  
(object recognition, speech recognition,  
language modeling, music modeling)**

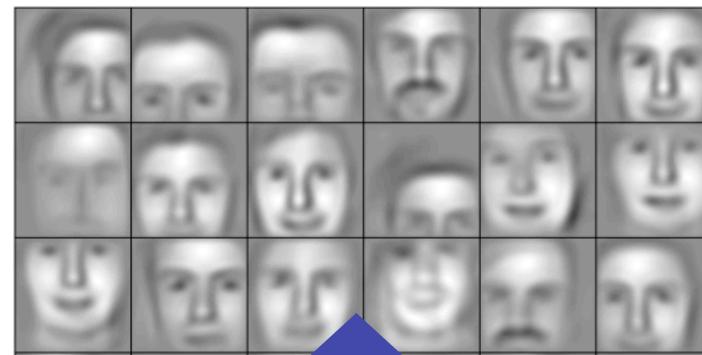


# Learning multiple levels of representation



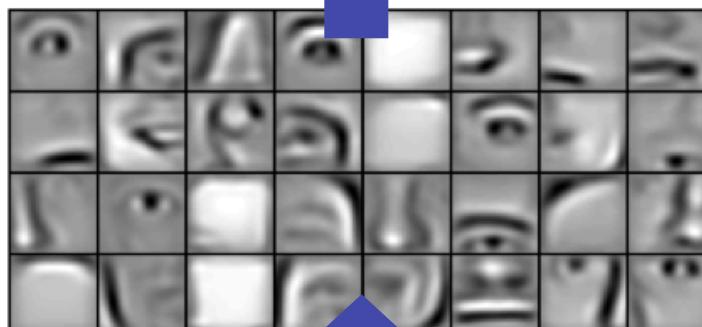
(Lee, Largman, Pham & Ng, NIPS 2009)  
(Lee, Grosse, Ranganath & Ng, ICML 2009)

Successive model layers learn deeper intermediate representations

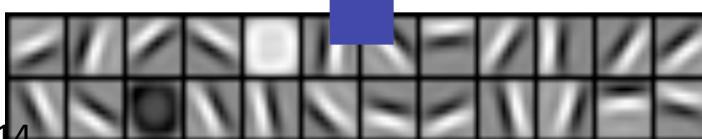


Layer 3

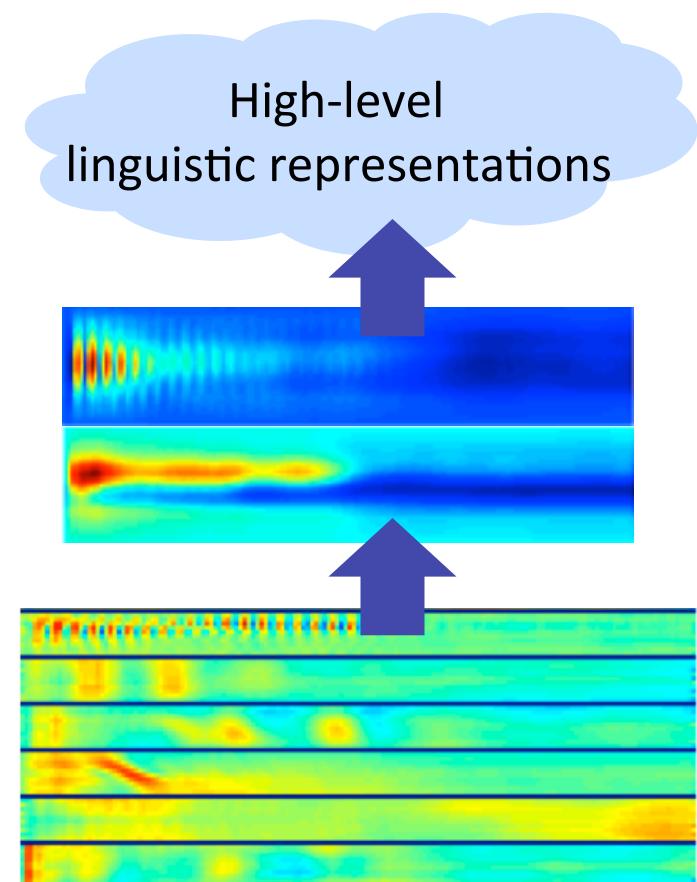
Parts combine  
to form objects



Layer 2

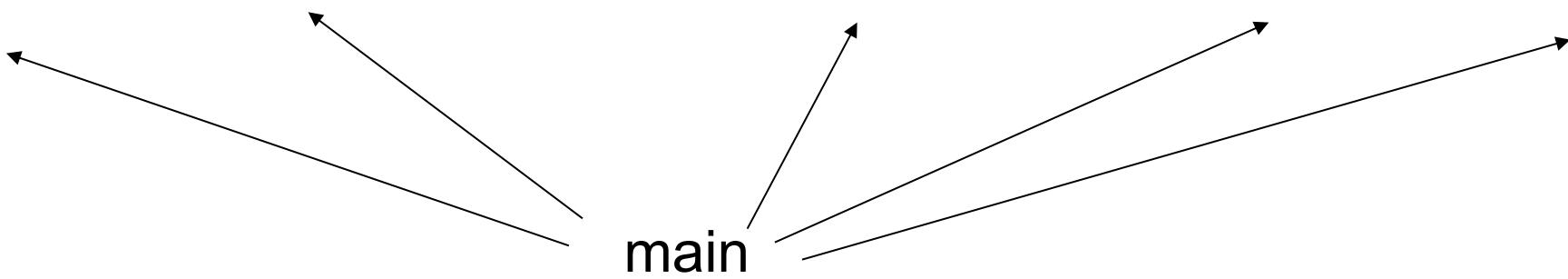


Layer 1

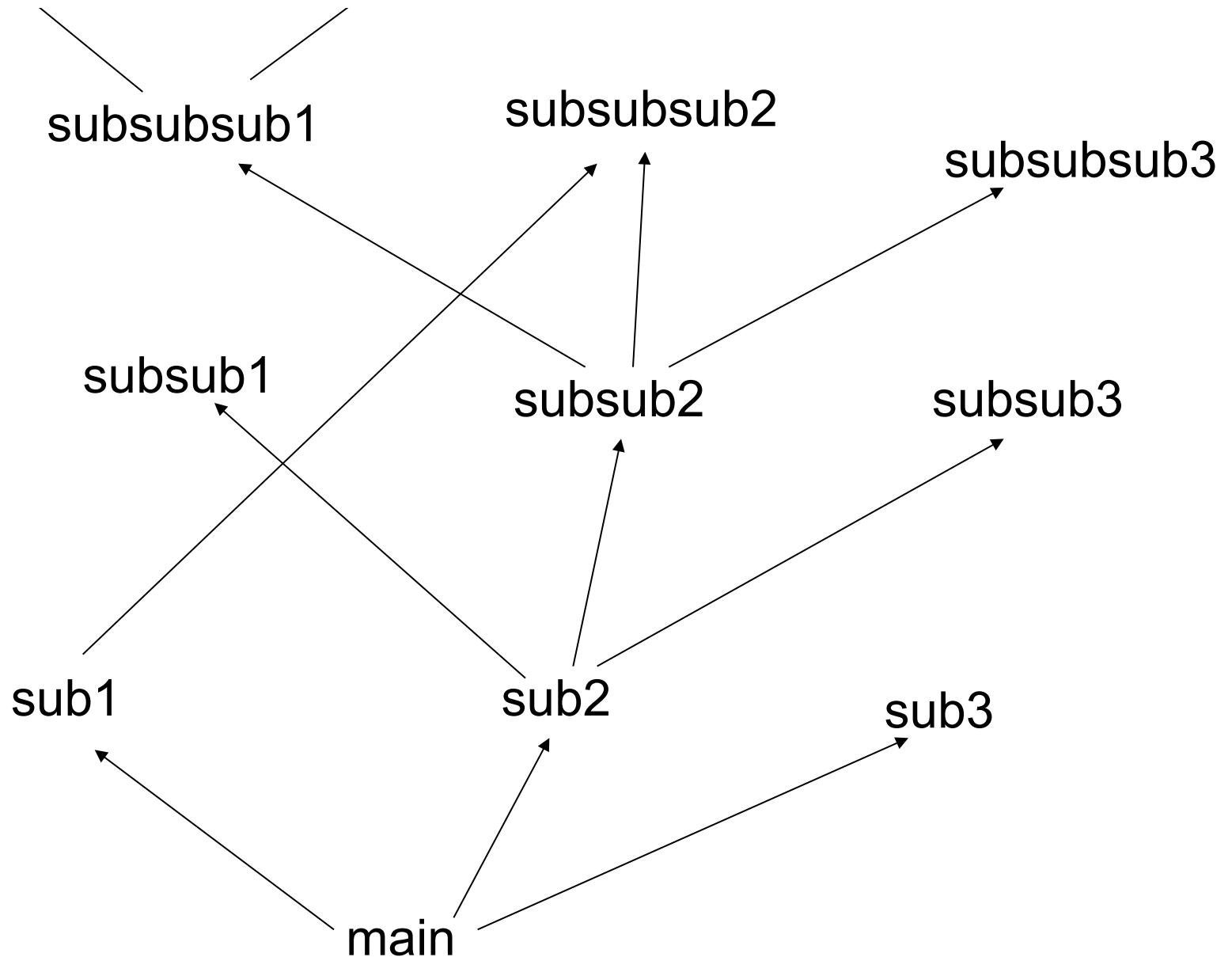


subroutine1 includes  
subsub1 code and  
subsub2 code and  
subsubsub1 code

subroutine2 includes  
subsub2 code and  
subsub3 code and  
subsubsub3 code and ...



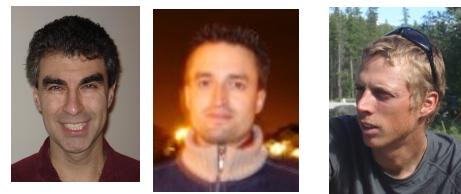
**“Shallow” computer program**



“Deep” computer program

# Deep Architectures are More Expressive

Theoretical arguments:



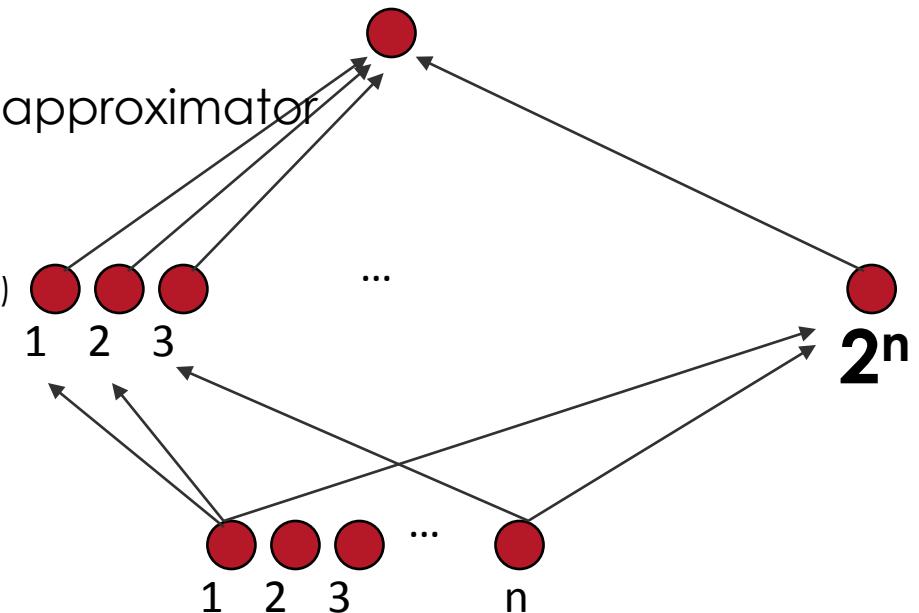
2 layers of Logic gates  
Formal neurons  
RBF units = universal approximator

RBMs & auto-encoders = universal approximator

Theorems on advantage of depth:

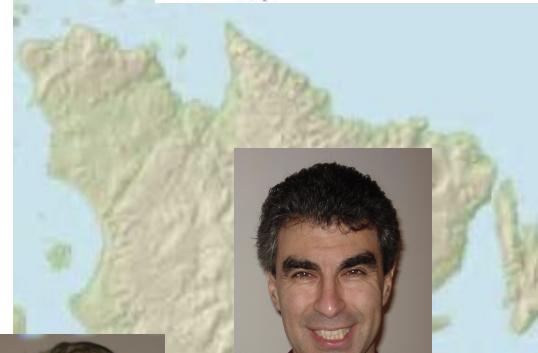
(Hastad et al 86 & 91, Bengio et al 2007, Bengio & Delalleau 2011, Braverman 2011, Pascanu et al 2014)

Some functions compactly represented with  $k$  layers may require exponential size with 2 layers



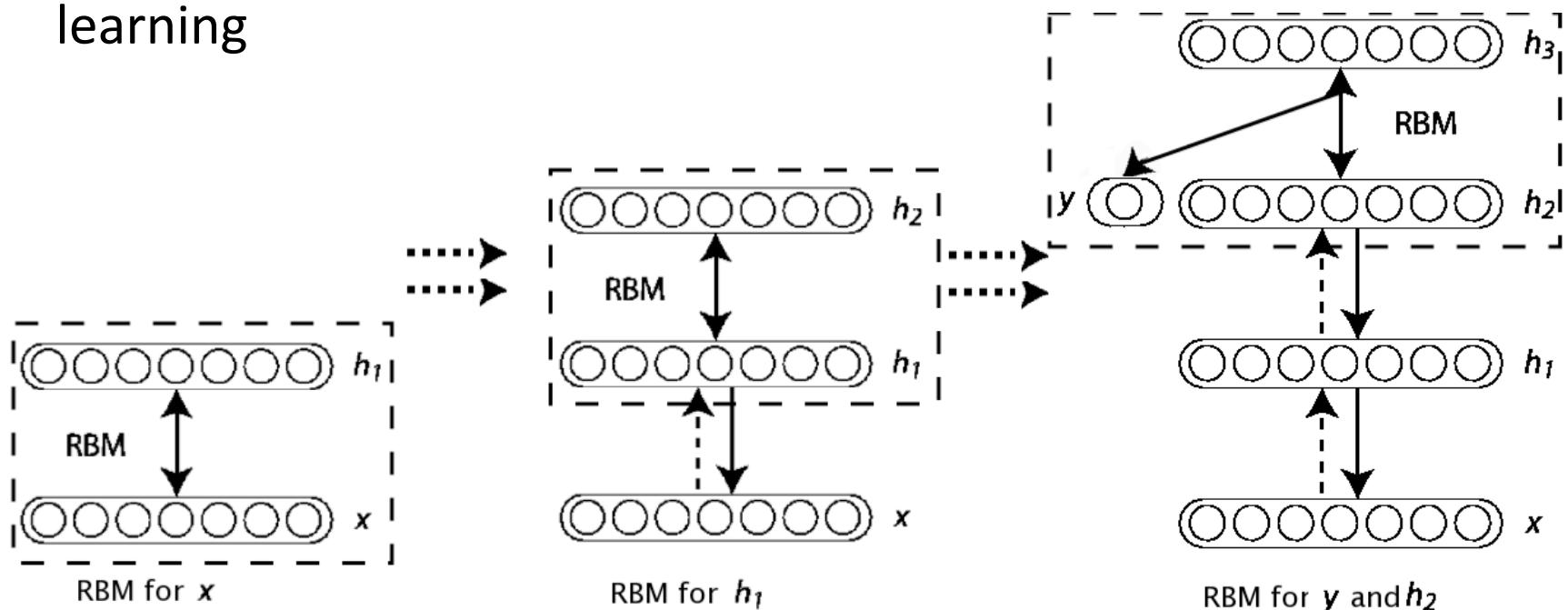
# Breakthrough in 2006

- Ability to train deep architectures by using layer-wise unsupervised learning, whereas previous purely supervised attempts had failed
- Unsupervised feature learners:
  - RBMs
  - Auto-encoder variants
  - Sparse coding variants



# Stacking Single-Layer Learners

- One of the big ideas from 2006: layer-wise unsupervised feature learning

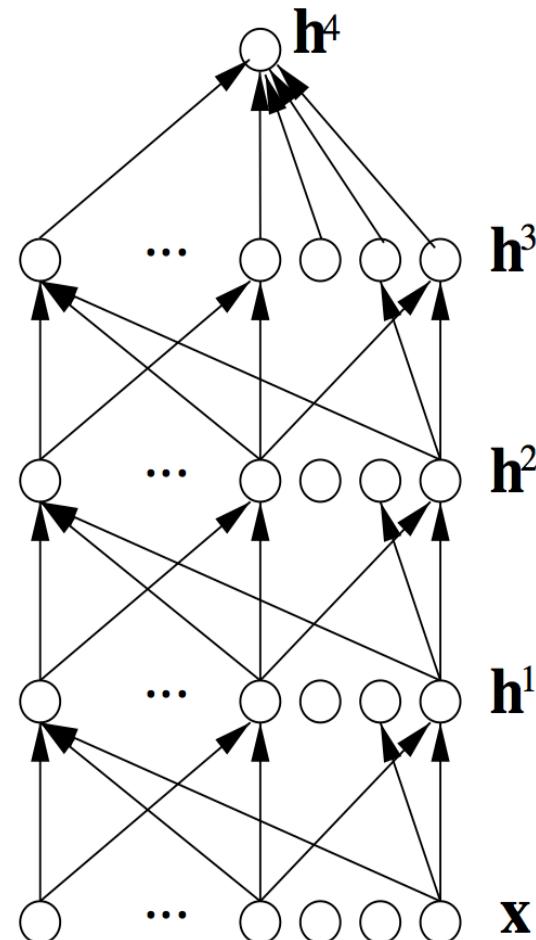


Stacking Restricted Boltzmann Machines (RBM) → Deep Belief Network (DBN)

Stacking regularized auto-encoders → deep neural nets

# Deep Supervised Neural Nets

- Now can train them even without unsupervised pre-training:  
**better initialization and non-linearities** (rectifiers, maxout), generalize well with large labeled sets and regularizers (dropout)
- **Unsupervised pre-training:**  
rare classes, transfer, smaller labeled sets, or as extra regularizer.



# Deep Learning in the News



# WIRED

Researcher Dreams Up Machines  
That Learn Without Humans  
06.27.13

Yoshua Bengio. Image: C

## The New York Times

Scientists See Promise in  
Deep-Learning Programs

John Markoff

November 23, 2012

## THE GLOBE AND MAIL

CANADA'S NATIONAL NEWSPAPER • FOUNDED 1844

Google taps U  
of T professor  
to teach  
context to  
computers  
03.11.13



# WIRED

The Man Behind the Google Brain: Andrew Ng  
and the Quest for the New AI

BY DANIELA HERNANDEZ 05.07.13 6:30 AM

21

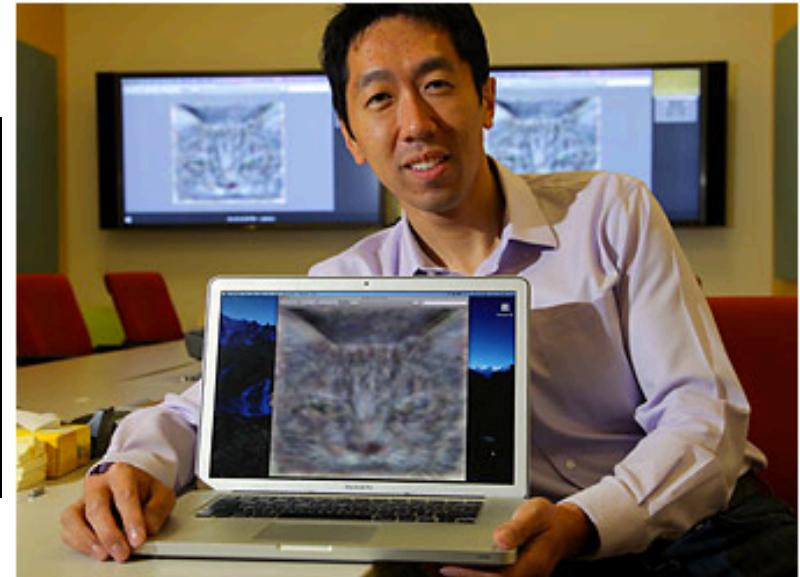
# The New York Times

Monday, June 25, 2012 Last Update: 11:50 PM ET

DIGITAL SUBSCRIPTION: 4 WEEKS

ING DIRECT

Follow Us [f](#) [t](#) [g](#)



Jim Wilson/The New York Times

### Despite Itself, a Simulated Brain Seeks Cats

By JOHN MARKOFF 12 minutes ago

A Google research team, led by Andrew Y. Ng, above, and Jeff Dean, created a neural network of 16,000 processors that reflected human obsession with Internet felines.

# 10 BREAKTHROUGH TECHNOLOGIES 2013

Intr

## Deep Learning

With massive amounts of computational power, machines can now recognize objects and translate speech in real time. Artificial intelligence is finally getting smart.

## Temporary Social Media

Messages that quickly self-destruct could enhance the privacy of online communications and make people freer to be spontaneous.

## Prenatal DNA Sequencing

Reading the DNA of fetuses will be the next frontier of the genomic revolution. But do you really want to know about the genetic problems or musical aptitude of your unborn child?

## Memory Implants

A maverick neuroscientist believes he has deciphered the code by which the brain

## Smart Watches

## Ultra-Efficient Solar Power

Doubling the efficiency of a solar cell would completely

## Add Ma

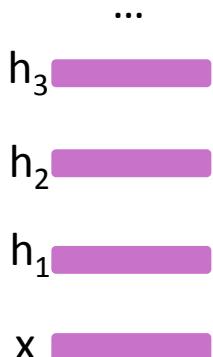
Ske  
prin  
wor  
mar  
the  
tech  
jet p

## Big Ph

Coll  
ana  
fron  
pho

# Deep Learning

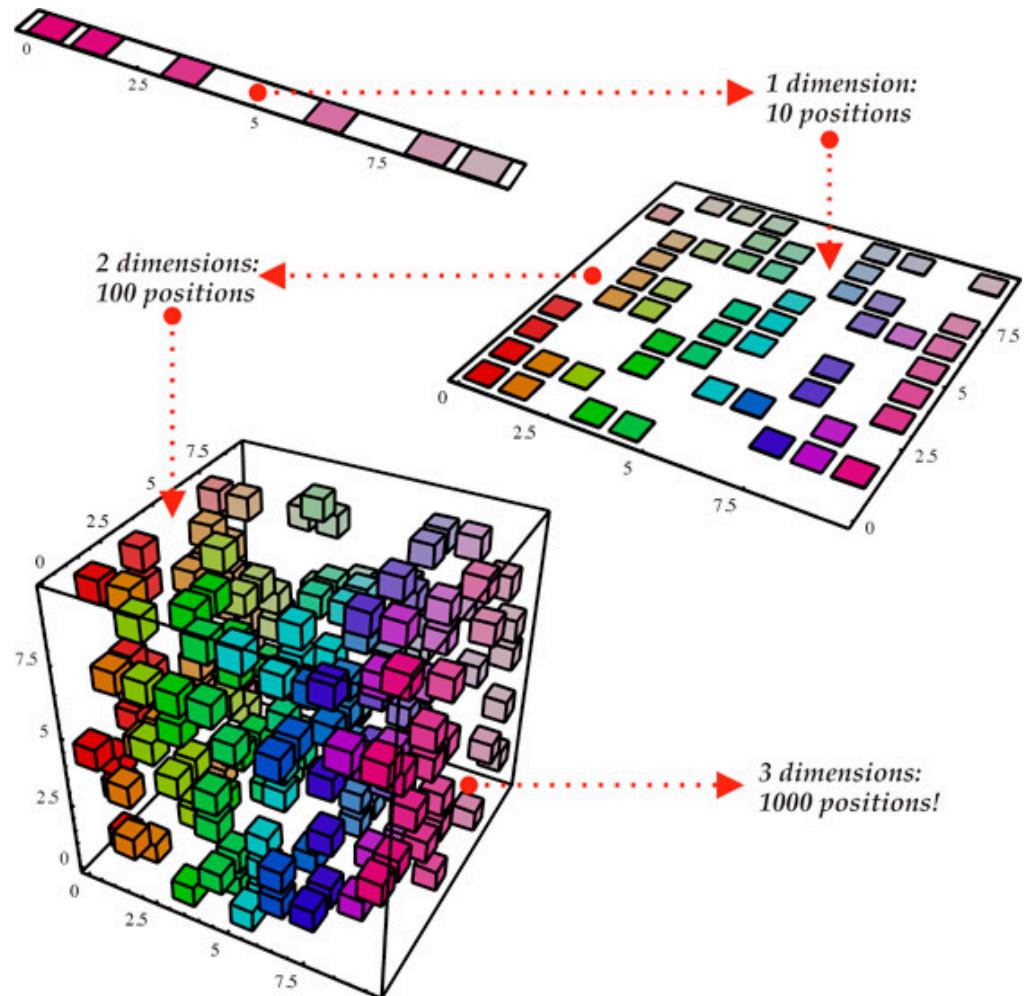
*When the number of levels can be data-selected, this is a **deep architecture***



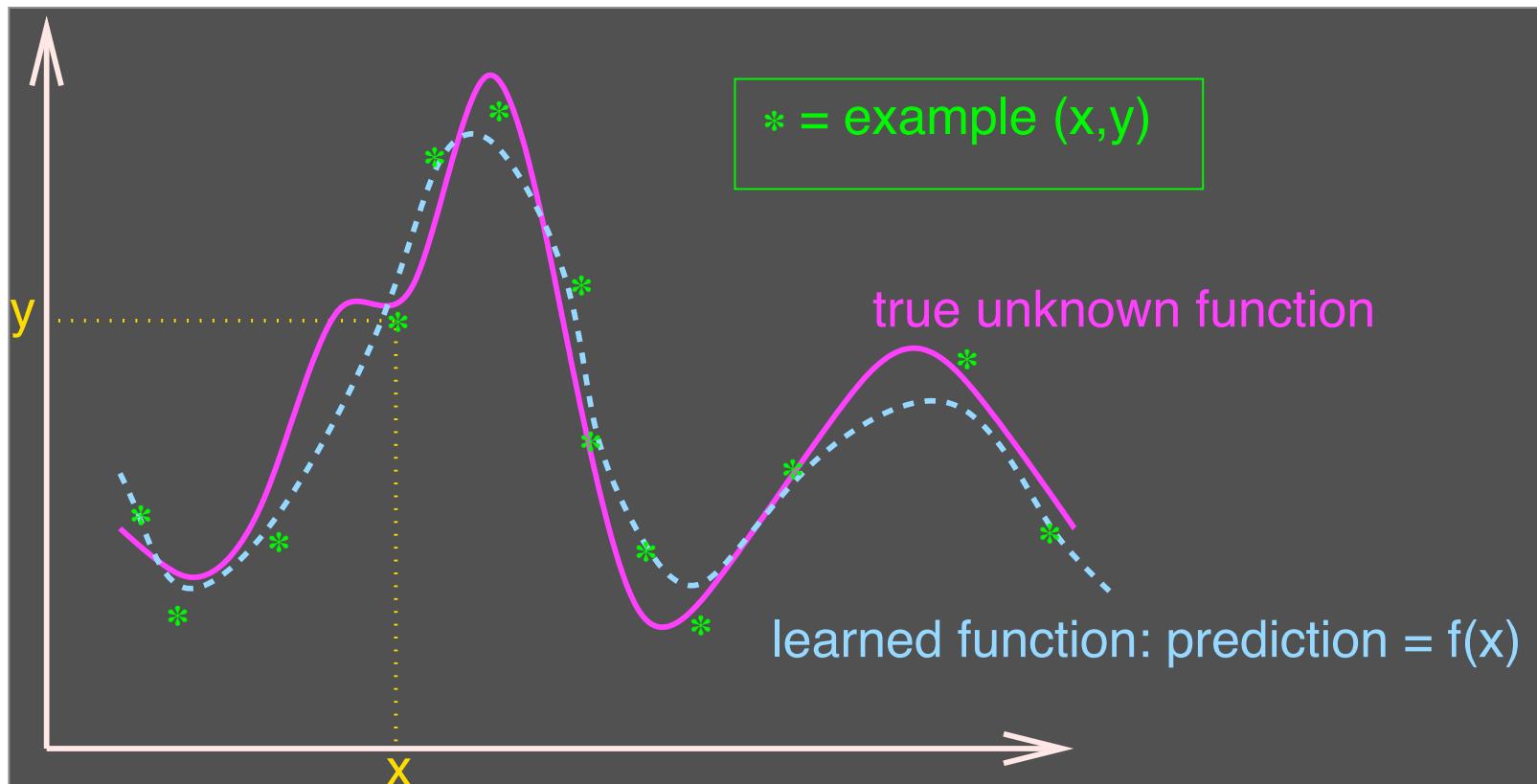
# ML 101. What We Are Fighting Against: The Curse of Dimensionality

To generalize locally,  
need representative  
examples for all  
relevant variations!

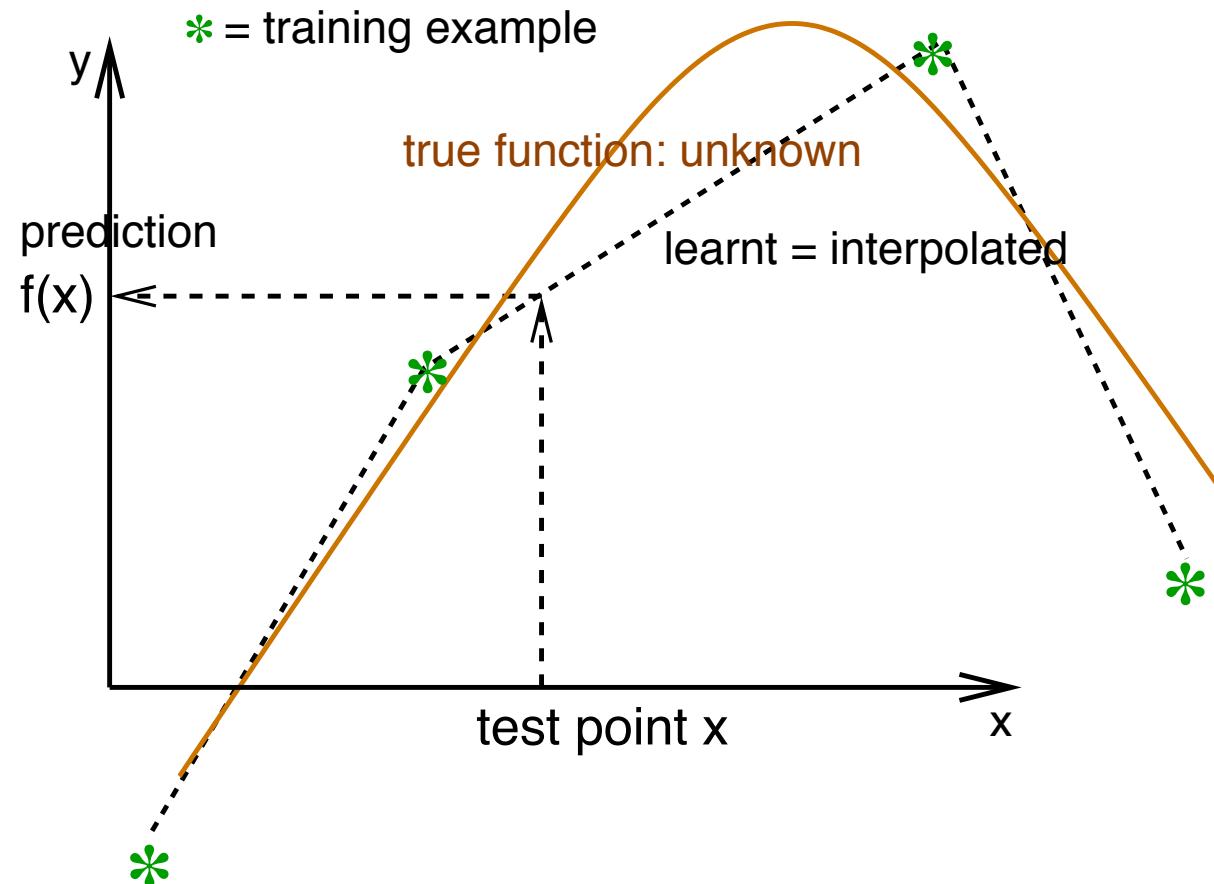
Classical solution: hope  
for a smooth enough  
target function, or  
make it smooth by  
handcrafting good  
features / kernel



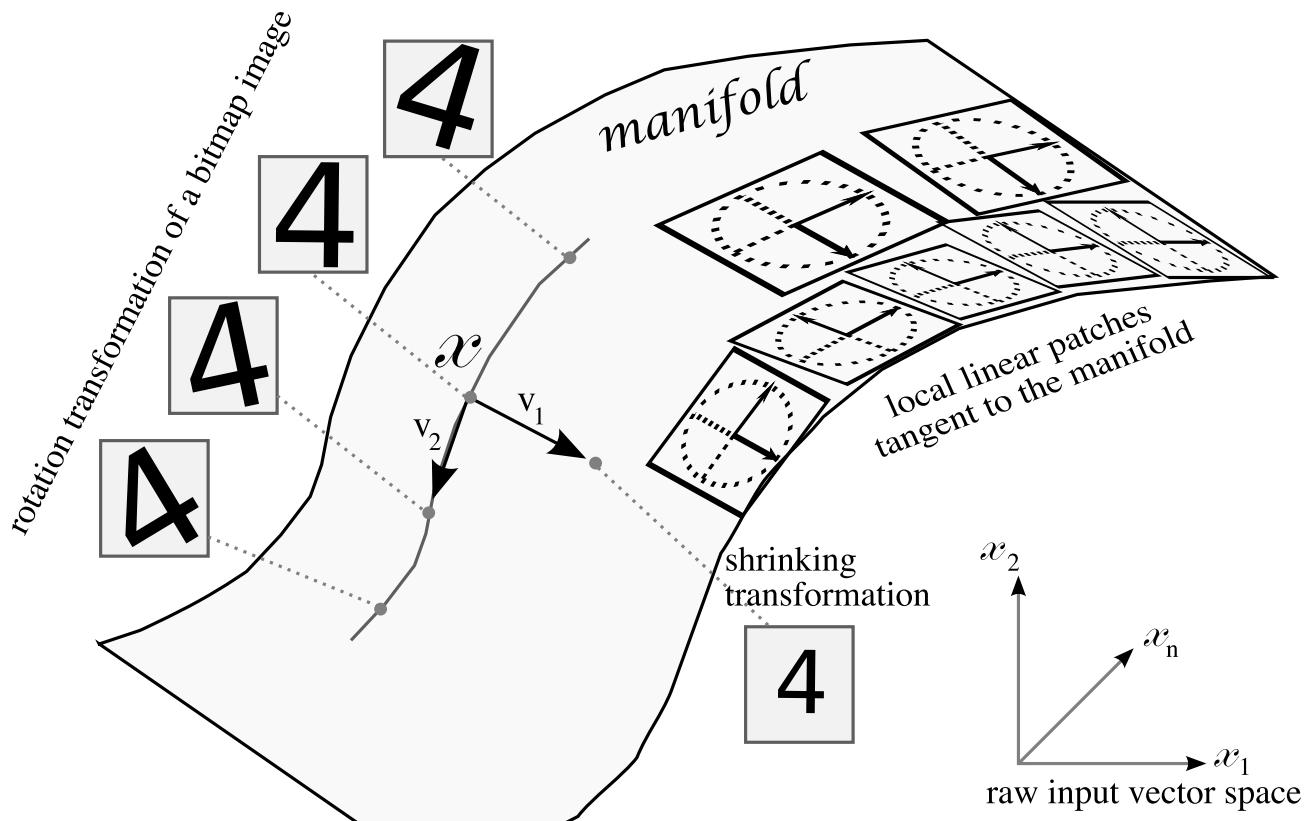
# Easy Learning



# Local Smoothness Prior: Locally Capture the Variations



# However, Real Data Are near Highly Curved Sub-Manifolds

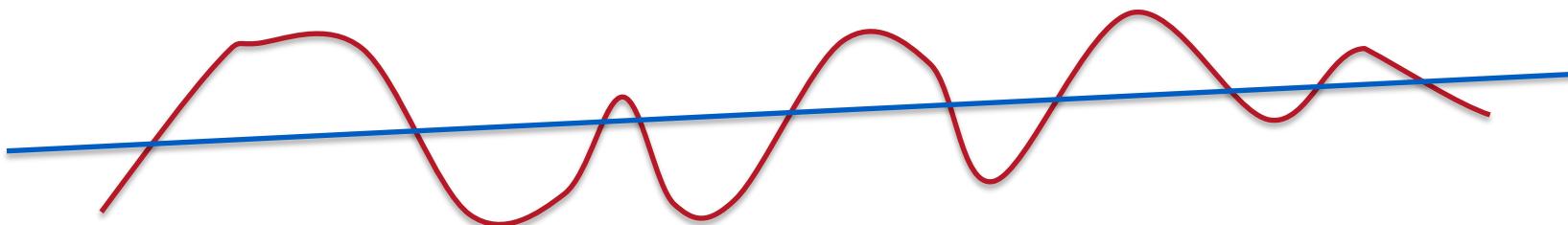


# Not Dimensionality so much as Number of Variations



(Bengio, Dellalleau & Le Roux 2007)

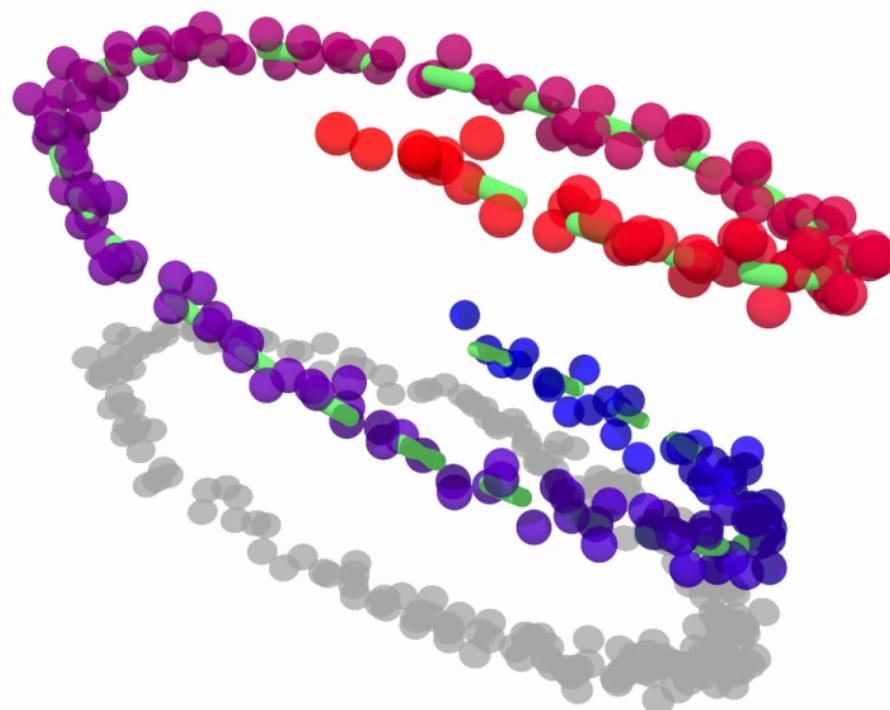
- **Theorem:** Gaussian kernel machines need at least  $k$  examples to learn a function that has  $2k$  zero-crossings along some line



- **Theorem:** For a Gaussian kernel machine to learn some maximally varying functions over  $d$  inputs requires  $O(2^d)$  examples

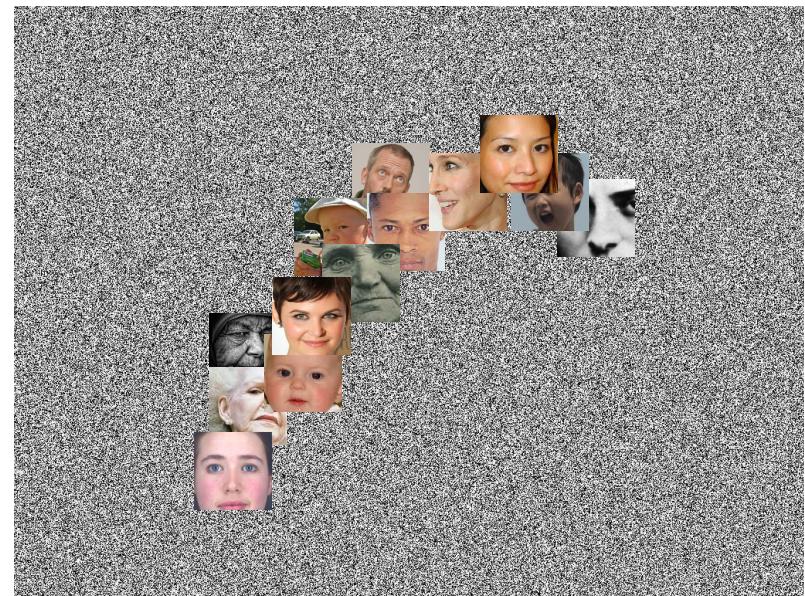
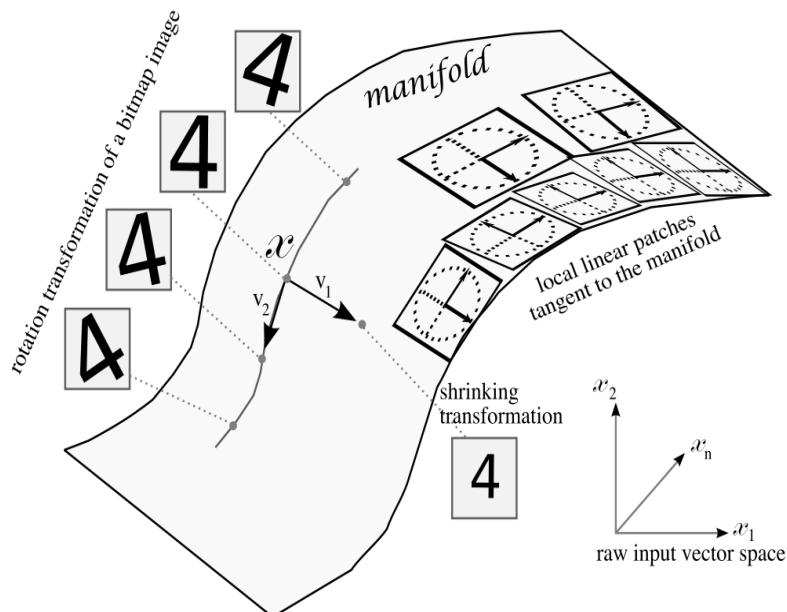
# Geometrical view on machine Learning

- Generalization: guessing **where** *probability* mass concentrates
- Challenge: the curse of dimensionality (exponentially many configurations of the variables to consider)

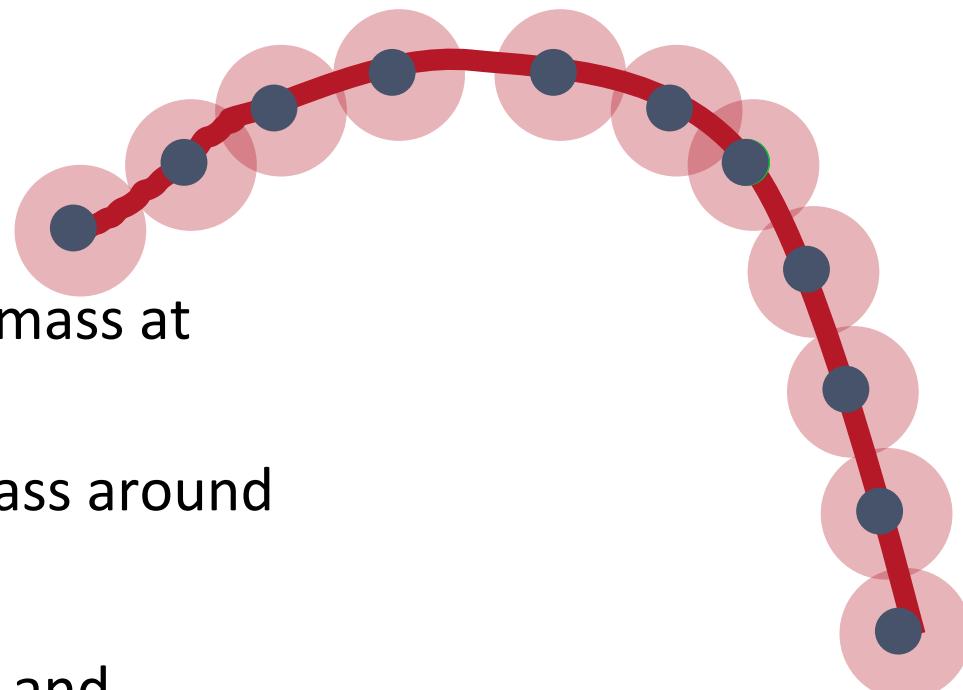


# For AI Tasks: Manifold structure

- examples **concentrate** near a lower dimensional “manifold”
- **Evidence: most input configurations are unlikely**



# Putting Probability Mass where Structure is Plausible



- Empirical distribution: mass at training examples
- Smoothness: spread mass around
- Insufficient
- Guess some ‘structure’ and generalize accordingly

Is there any hope to  
generalize non-locally?

Yes! Need good priors!

# Bypassing the curse

We need to build **compositionality** into our ML models

Just as human languages exploit compositionality to give representations and meanings to complex ideas

Exploiting compositionality gives an exponential gain in representational power

Distributed representations / embeddings: **feature learning**

Deep architecture: **multiple levels of feature learning**

Prior: compositionality is useful to describe the world around us efficiently

# Six Good Reasons to Explore Representation Learning

# #1 Learning features, not just handcrafting them

Most ML systems use very carefully hand-designed features and representations

Many practitioners are very experienced – and good – at such feature design (or kernel design)

“Machine learning” often reduces to linear models (including CRFs) and nearest-neighbor-like features/models (including n-grams, kernel SVMs, etc.)

**Hand-crafting features is time-consuming, brittle, incomplete**

## #2 Distributed Representations

Many neurons active simultaneously in the brain: around 1%

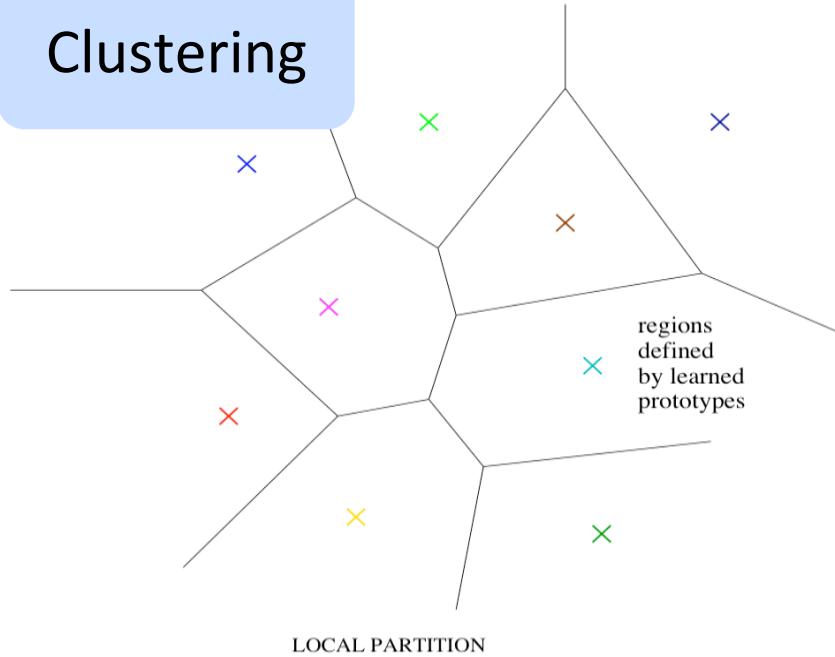
The input is represented by the activation of a set of features that are not mutually exclusive.

Can be exponentially more efficient than local representations



# #2 Non-distributed representations

## Clustering

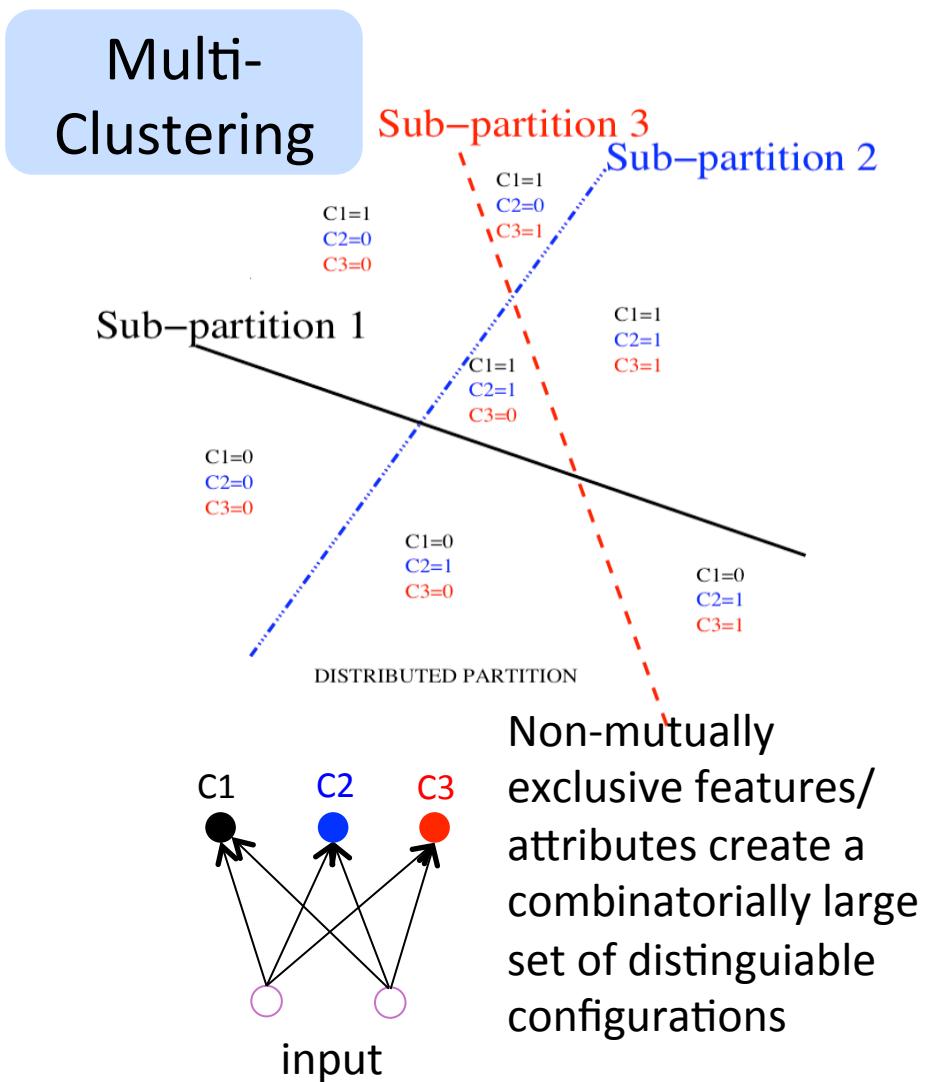


- Clustering, Nearest-Neighbors, RBF SVMs, local non-parametric density estimation & prediction, decision trees, etc.
- Parameters for each distinguishable region
- **# of distinguishable regions is linear in # of parameters**

→ No non-trivial generalization to regions without examples

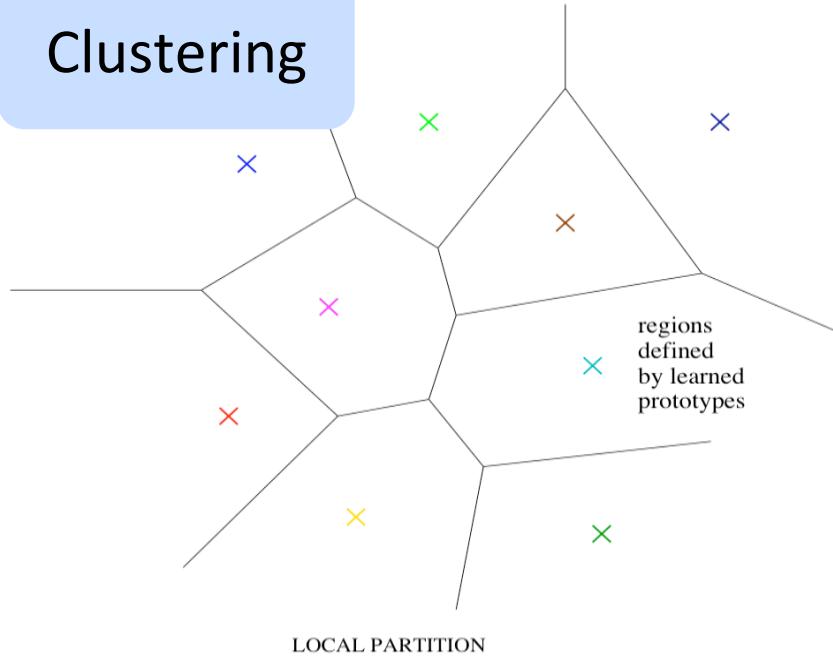
# #2 The need for distributed representations

- Factor models, PCA, RBMs, Neural Nets, Sparse Coding, Deep Learning, etc.
- Each parameter influences many regions, not just local neighbors
- **# of distinguishable regions grows almost exponentially with # of parameters**
- **GENERALIZE NON-LOCALLY TO NEVER-SEEN REGIONS**

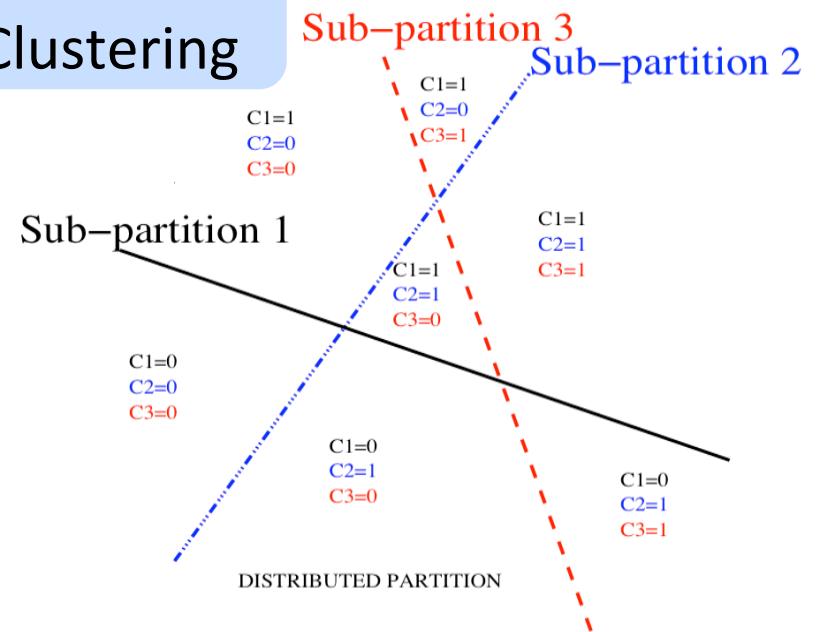


# #2 The need for distributed representations

Clustering



Multi-Clustering



Learning a **set of features** that are not mutually exclusive can be **exponentially more statistically efficient** than having nearest-neighbor-like or clustering-like models

## #3 Unsupervised feature Learning

Today, most practical ML applications require (lots of) labeled training data

But almost all **data is unlabeled**, e.g. text, images on the web

The brain needs to learn about  $10^{14}$  synaptic strengths

... in about  $10^9$  seconds

Labels cannot possibly provide enough information

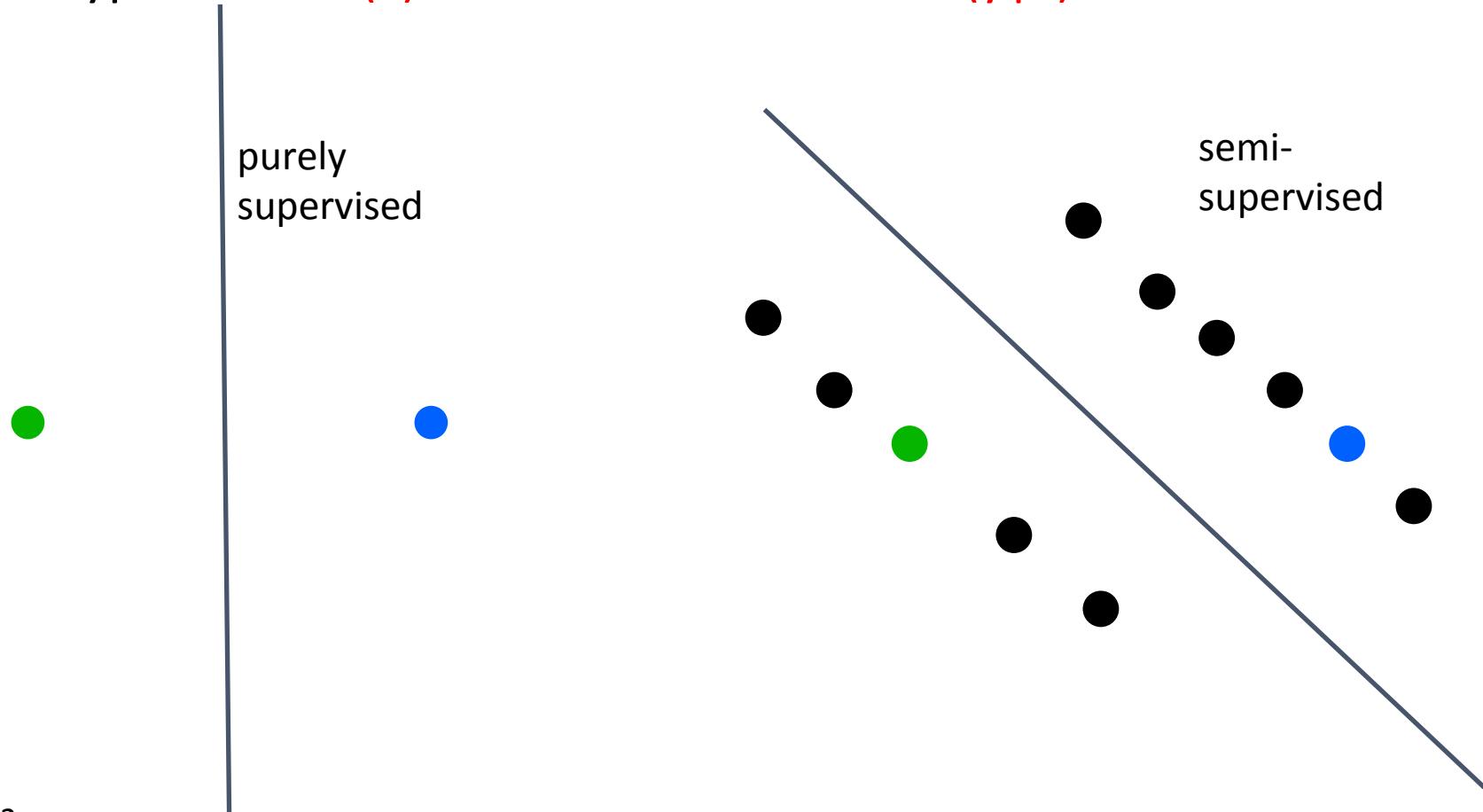
Most information acquired in an **unsupervised** fashion

# #3 How do humans generalize from very few examples?

- They **transfer** knowledge from previous learning:
  - Representations
  - Explanatory factors
- Previous learning from: unlabeled data
  - + labels for other tasks
- **Prior: shared underlying explanatory factors, in particular between  $P(x)$  and  $P(Y|x)$**

## #3 Sharing Statistical Strength by Semi-Supervised Learning

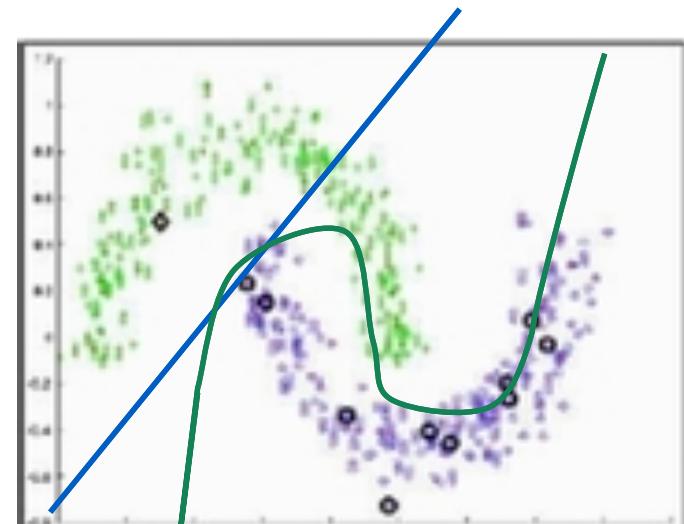
- Hypothesis:  $P(x)$  shares structure with  $P(y|x)$



# Why Semi-Supervised Learning Works

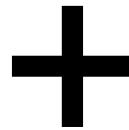
- The labeled examples (circles) help to identify the class of each cluster of unlabeled examples.
- The unlabeled examples (colored dots) help to identify the shape of each cluster.

Without unlabeled examples



With unlabeled examples

few labeled  
examples

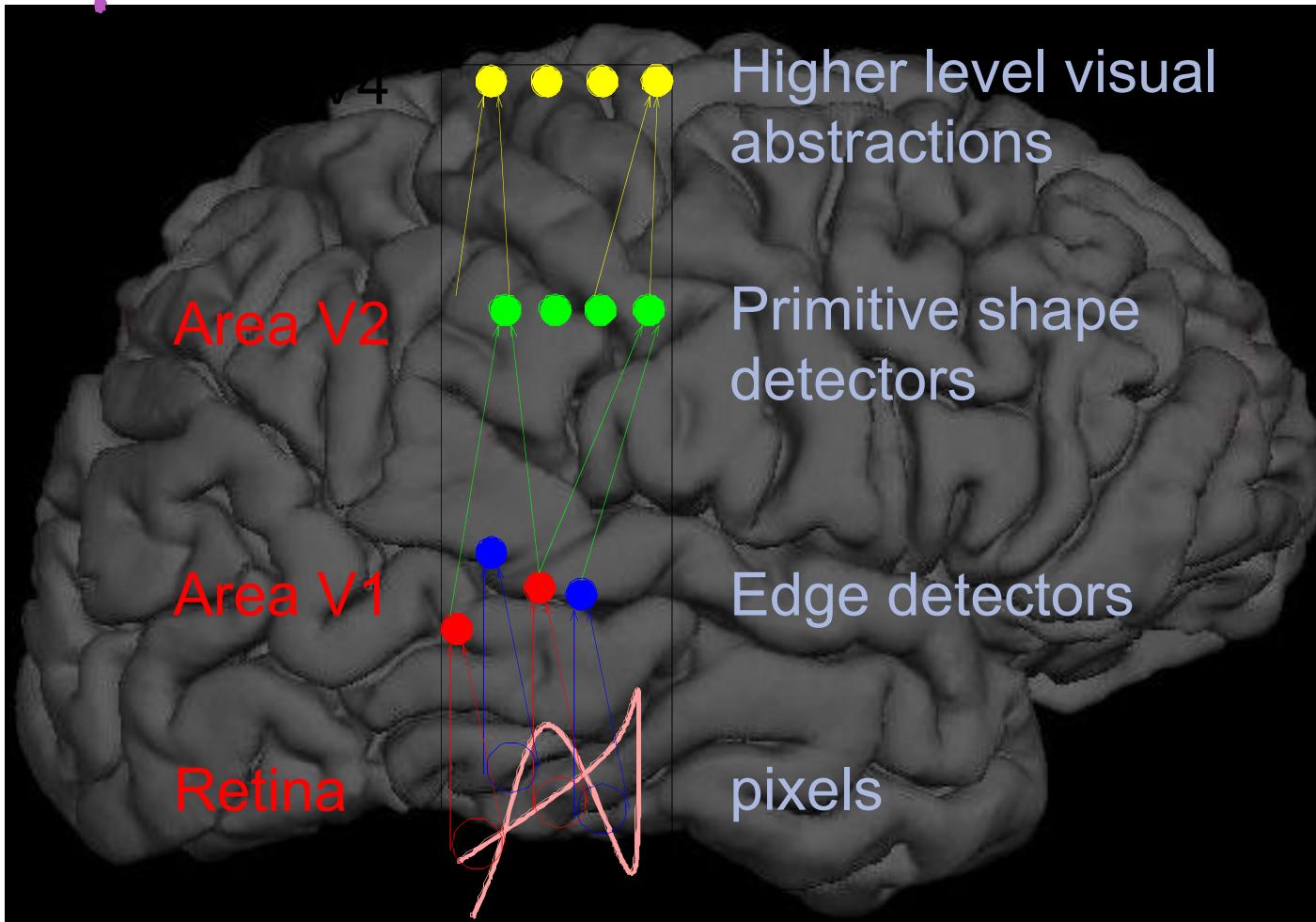


“happiness”

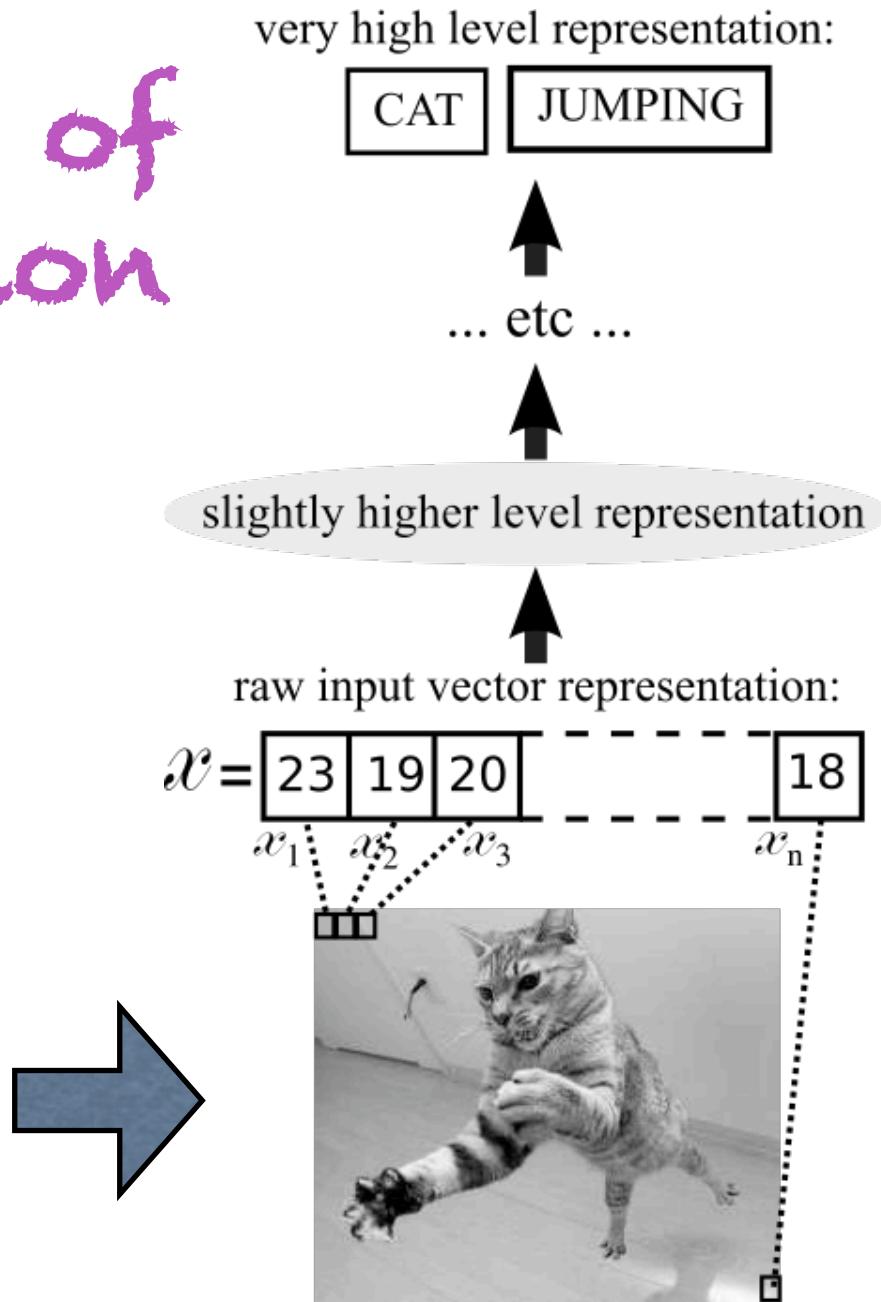


many unlabeled examples

# #4 Depth Deep Architecture in the Brain



# #4 Levels of Representation

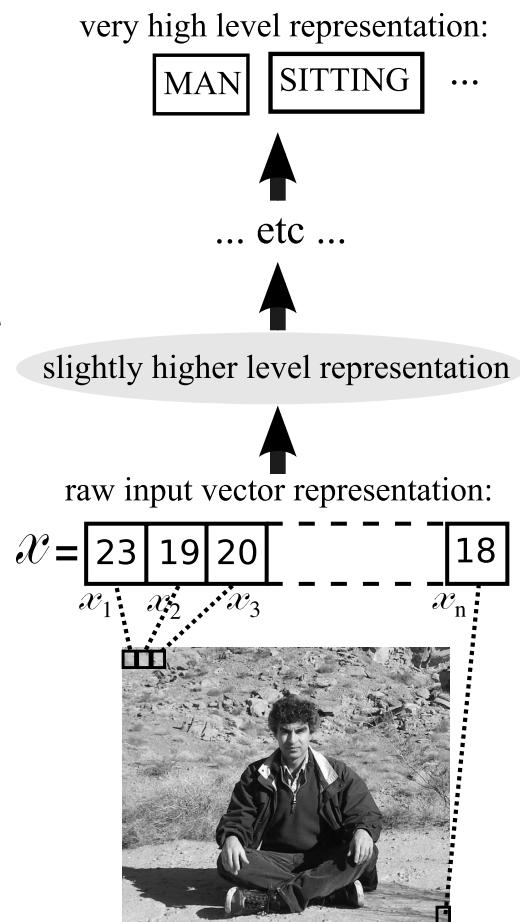


# #4 Deep Architecture in our Mind

- 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 good to automatically learn / discover these concepts**

(knowledge engineering failed because of superficial introspection?)



# #4 Learning multiple levels of representation

There is theoretical and empirical evidence in favor of multiple levels of representation

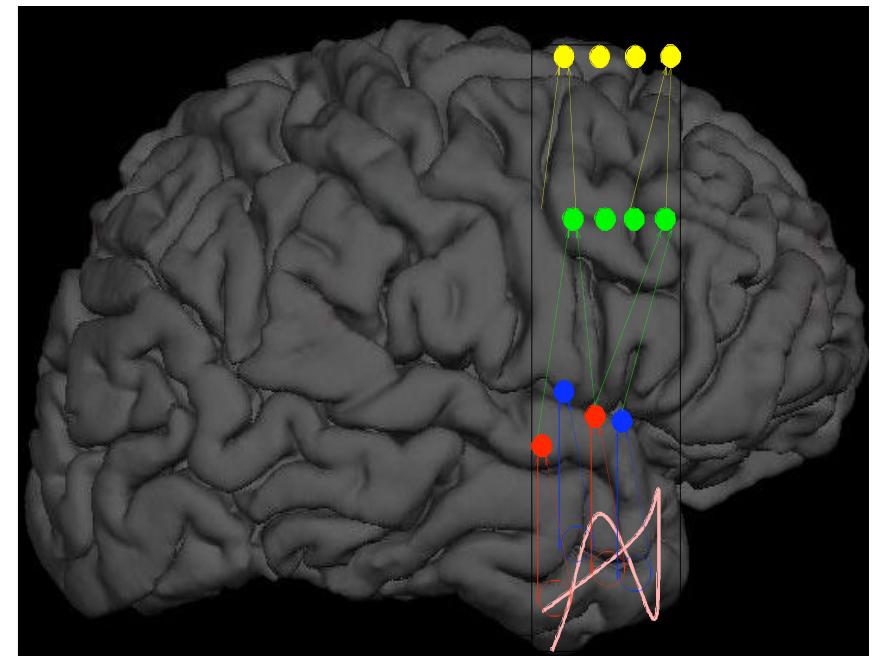
**Exponential gain for some families of functions**

Biologically inspired learning

Brain has a deep architecture

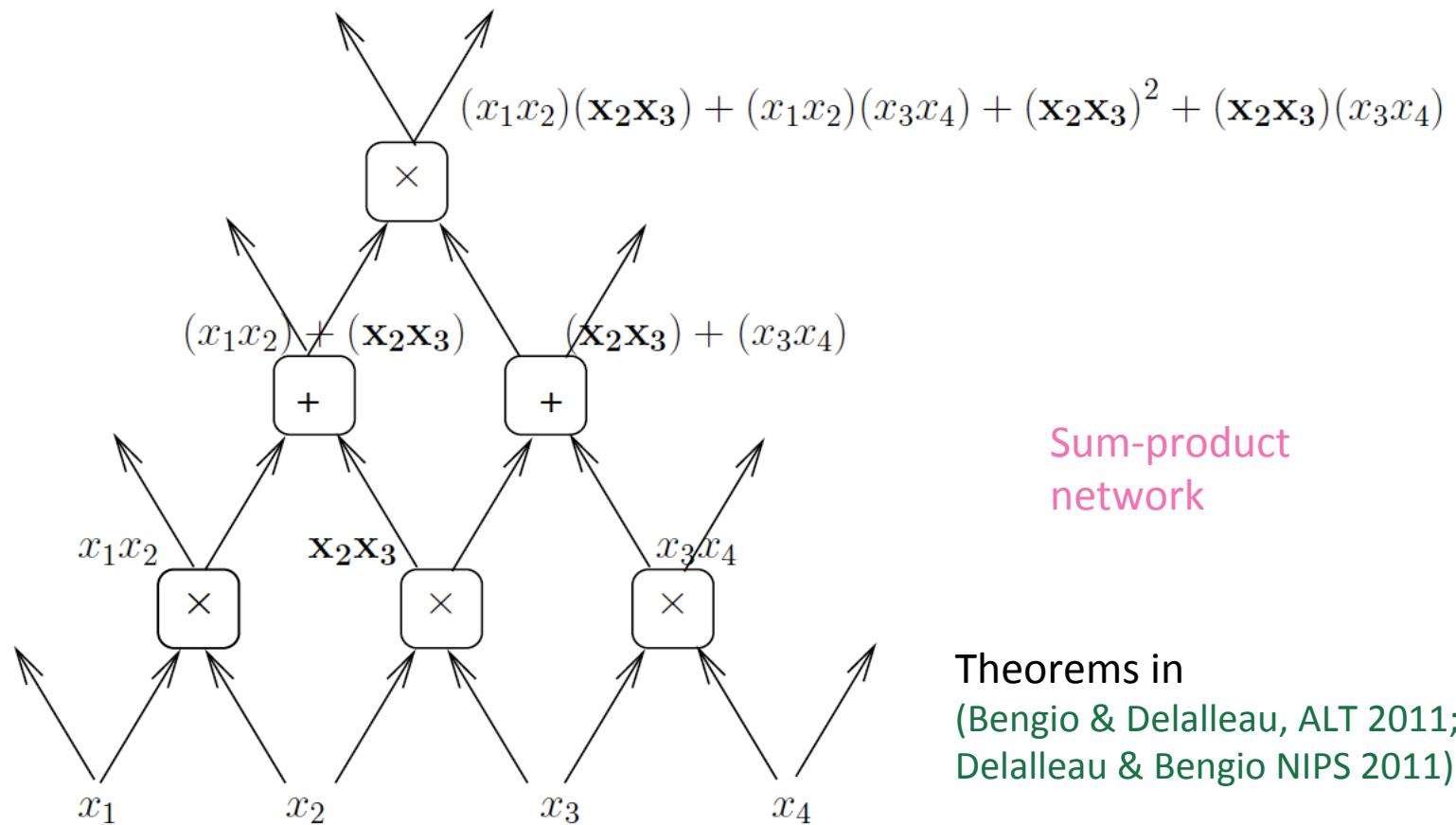
Cortex seems to have a generic learning algorithm

**Humans first learn simpler concepts and then compose them into more complex ones**



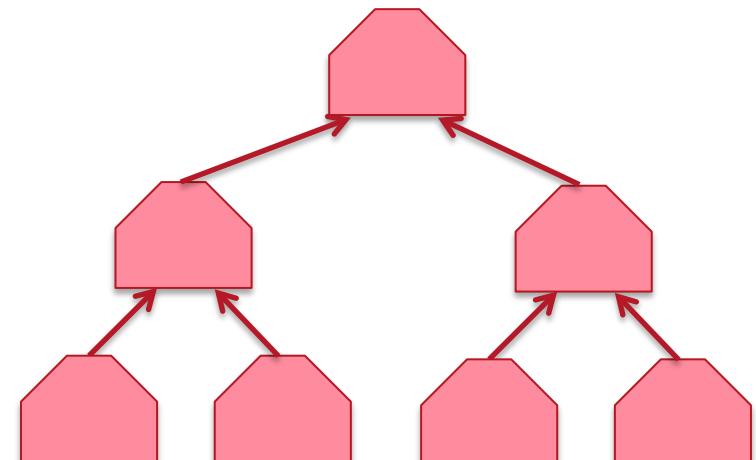
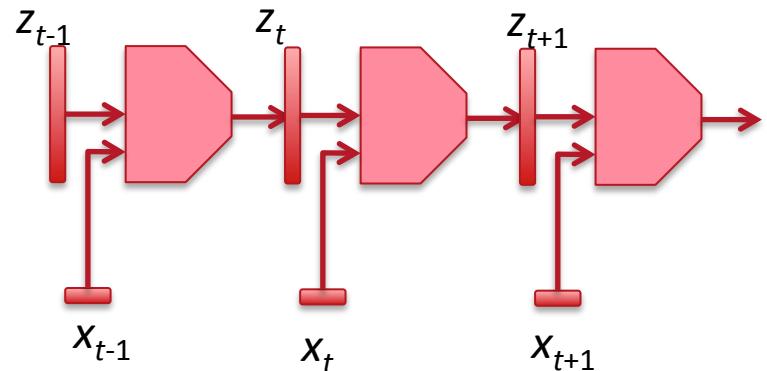
## #4 Sharing Components in a Deep Architecture

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



# #4 Handling the compositionality of human language and thought

- Human languages, ideas, and artifacts are composed from simpler components
- **Recursion:** the same operator (same parameters) is applied repeatedly on different states/components of the computation
- Result after unfolding = deep computation / representation

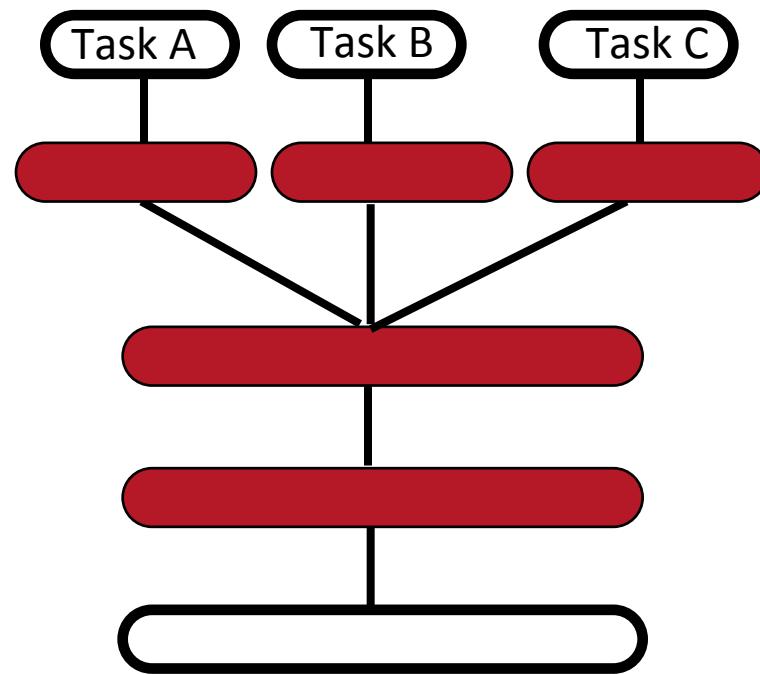


(Bottou 2011, Socher et al 2011)



# #5 Multi-Task Learning

- Generalizing better to new tasks (tens of thousands!) is crucial to approach AI
- Deep architectures learn good intermediate representations that can be shared across tasks  
*(Collobert & Weston ICML 2008, Bengio et al AISTATS 2011)*
- Good representations that disentangle underlying factors of variation make sense for many tasks because **each task concerns a subset of the factors**

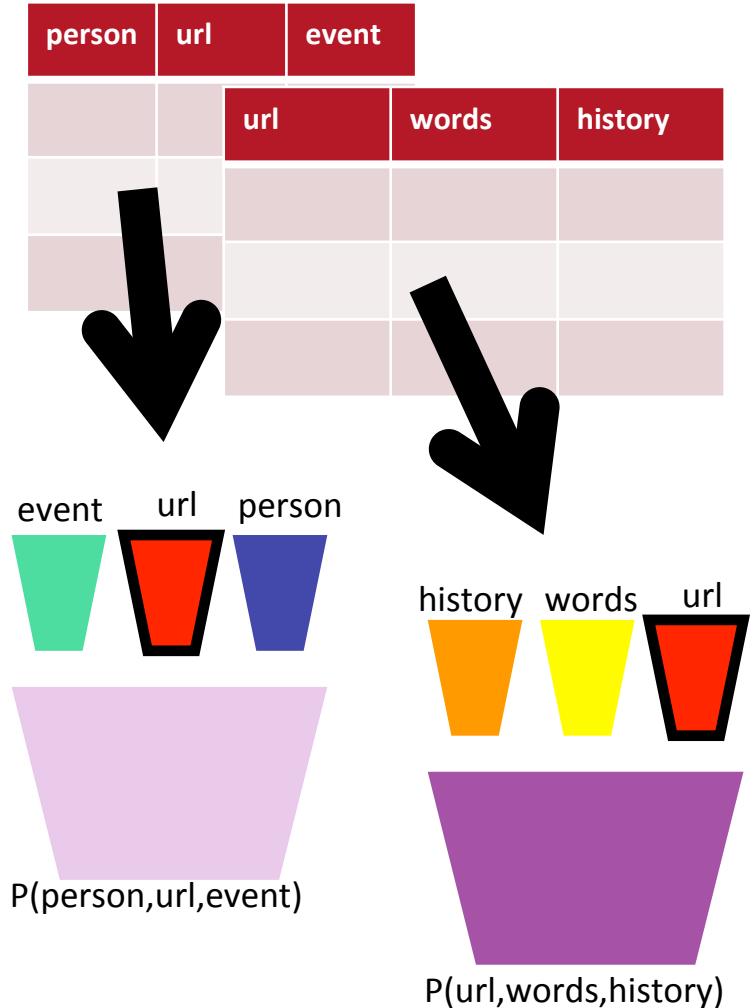


E.g. dictionary, with intermediate concepts re-used across many definitions

**Prior: shared underlying explanatory factors between tasks**

# #5 Combining Multiple Sources of Evidence with Shared Representations

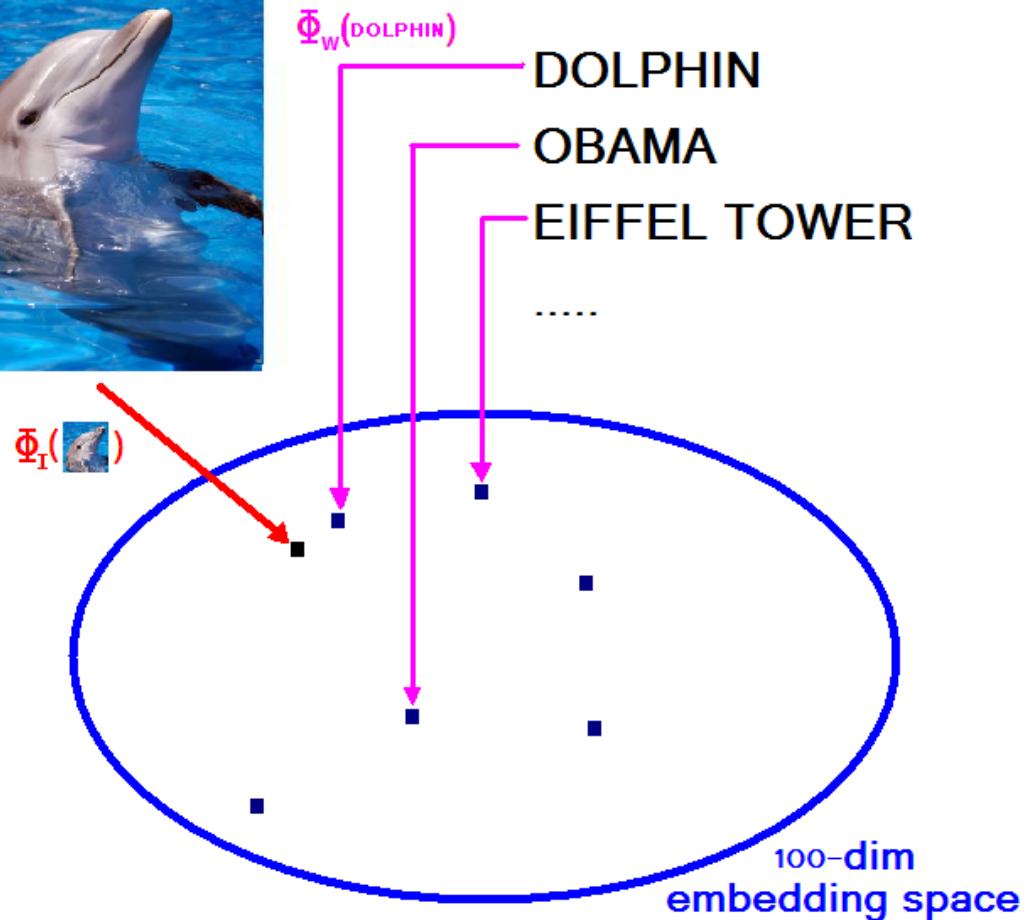
- Traditional ML: data = matrix
- Relational learning: multiple sources, different tuples of variables
- Share representations of same types across data sources
- Shared learned representations help propagate information among data sources: e.g., WordNet, XWN, Wikipedia, **FreeBase**, ImageNet...  
(Bordes et al AISTATS 2012, ML J. 2013)
- **FACTS = DATA**
- **Deduction = Generalization**



# #5 Different object types represented in same space



Google:  
S. Bengio, J.  
Weston & N.  
Usunier  
(IJCAI 2011,  
NIPS'2010,  
JMLR 2010,  
ML J. 2010)



Learn  $\Phi_T(\cdot)$  and  $\Phi_w(\cdot)$  to optimize precision@k.

# #6 Invariance and Disentangling

- Invariant features
- Which invariances?
- Alternative: learning to disentangle factors
- Good disentangling →  
  avoid the curse of dimensionality



# #6 Emergence of Disentangling

- (Goodfellow et al. 2009): sparse auto-encoders trained on images
  - some higher-level features more invariant to geometric factors of variation
- (Glorot et al. 2011): sparse rectified denoising auto-encoders trained on bags of words for sentiment analysis
  - different features specialize on different aspects (domain, sentiment)



**WHY?**

# Broad Priors as Hints to Disentangle the Factors of Variation

- *Multiple factors*: distributed representations
- Multiple levels of abstraction: *depth*
- *Semi-supervised* learning: Y is one of the factors explaining X
- *Multi-task* learning: different tasks share some factors
- *Manifold* hypothesis: probability mass concentration
- Natural *clustering*: class = manifold, well-separated manifolds
- Temporal and spatial *coherence*
- *Sparsity*: most factors irrelevant for particular X
- *Simplicity* of factor dependencies (in the right representation)

# #6 Sparse Representations

- Just add a sparsifying penalty on learned representation  
(prefer 0s in the representation)
- Information disentangling (compare to dense compression)
- More likely to be linearly separable (high-dimensional space)
- Locally low-dimensional representation = local chart
- Hi-dim. sparse = efficient variable size representation  
= data structure

Few bits of information



Many bits of information



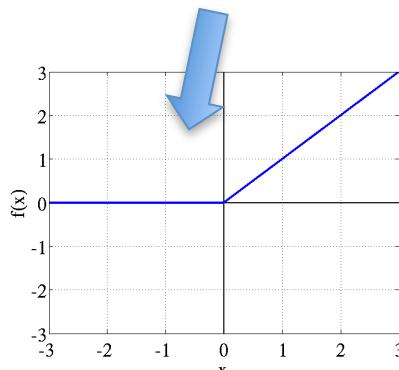
Prior: only few concepts and attributes relevant per example

# Deep Sparse Rectifier Neural Networks

(Glorot, Bordes and Bengio AISTATS 2011), following up on (Nair & Hinton 2010) softplus RBMs

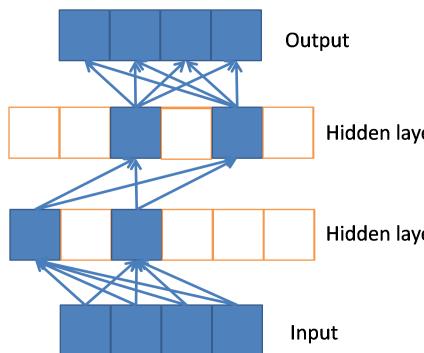
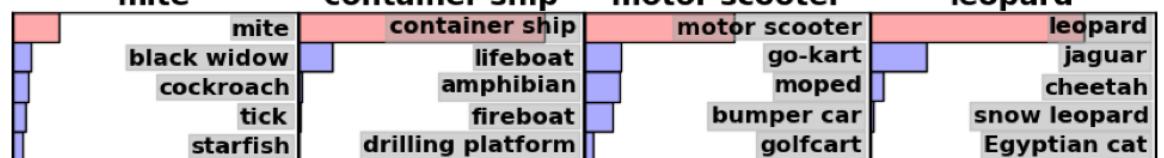
## Neuroscience motivations

Leaky integrate-and-fire model



## Machine learning motivations

- Sparse representations
- Sparse gradients
- Trains deep nets even w/o pretraining



**Outstanding results** by Krizhevsky et al 2012  
killing the state-of-the-art on ImageNet 1000:

	1 <sup>st</sup> choice	Top-5
2 <sup>nd</sup> best		27% err
Previous SOTA	45% err	26% err
Krizhevsky et al	37% err	15% err

# Temporal Coherence and Scales

- Hints from nature about different explanatory factors:
  - Rapidly changing factors (often noise)
  - Slowly changing (generally more abstract)
  - Different factors at different time scales
- Exploit those **hints** to **disentangle** better!
- (Becker & Hinton 1993, Wiskott & Sejnowski 2002, Hurri & Hyvarinen 2003, Berkes & Wiskott 2005, Mobahi et al 2009, Bergstra & Bengio 2009)

Part 2

# Representation Learning Algorithms

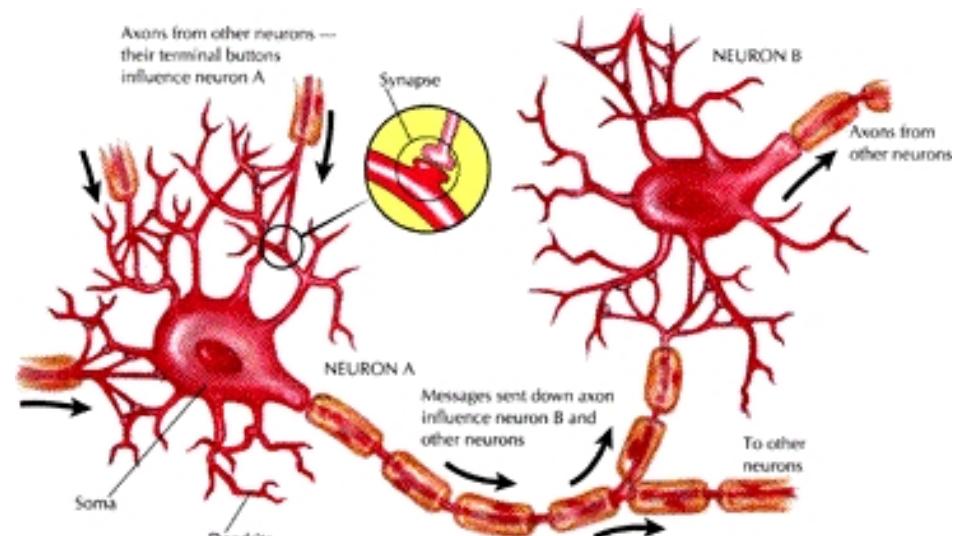
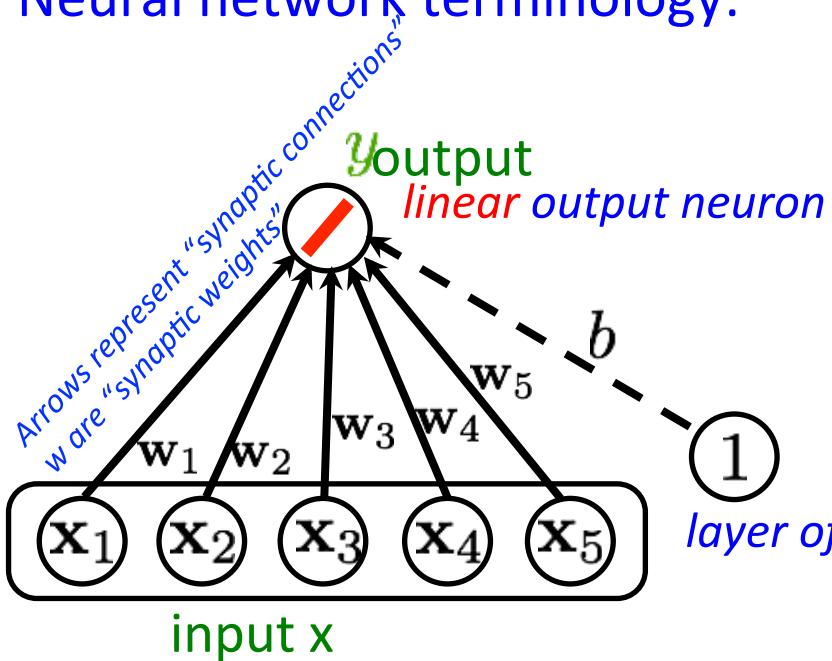
# Linear Regression

## Neural network view

Intuitive understanding of the dot product:  
each component of  $x$  weighs differently on the response.

$$y = f_{\theta}(\mathbf{x}) = \mathbf{w}_1 \mathbf{x}_1 + \mathbf{w}_2 \mathbf{x}_2 + \dots + \mathbf{w}_d \mathbf{x}_d + b$$

Neural network terminology:

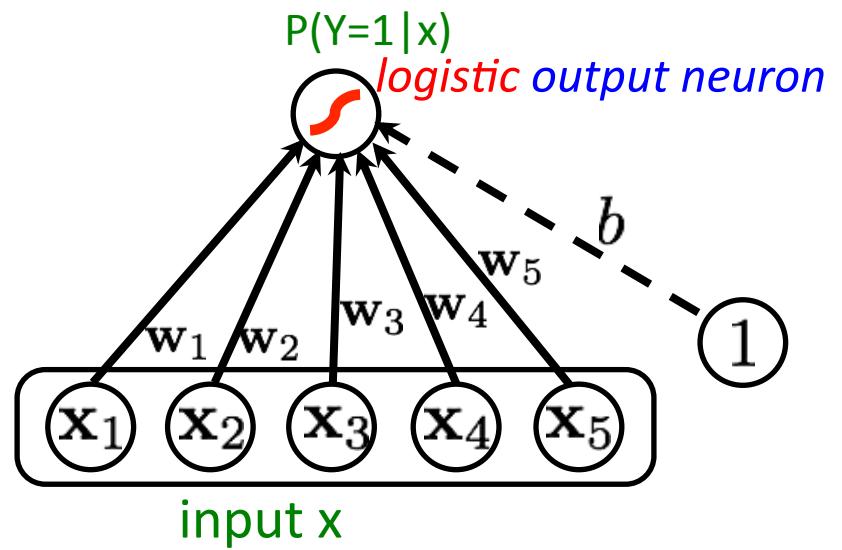
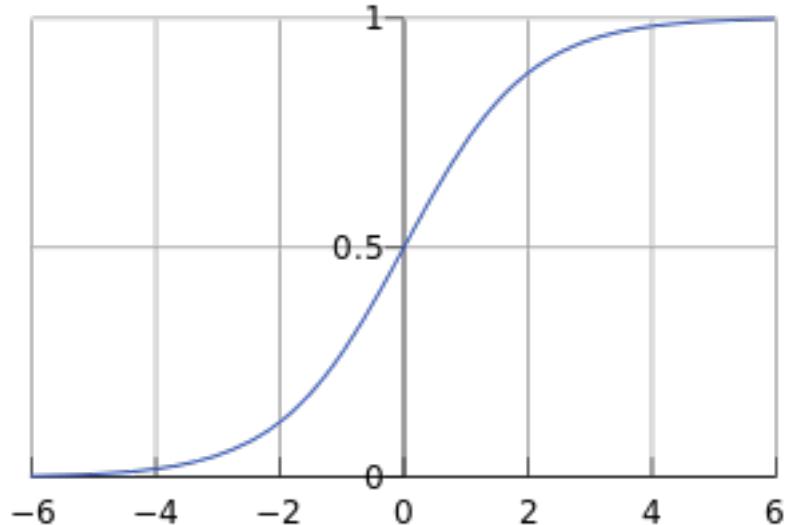


# Logistic Regression

- Predict the probability of a category  $y$ , given input  $x$ 
  - $P(Y=y | X=x)$
- Simple extension of linear regression (binary case):
  - $P(Y=1 | X=x) = \text{sigmoid}(b + w \cdot x)$
- Train by tuning  $(b, w)$  to maximize average log-likelihood
  - Average(  $\log P(Y=y | X=x)$  )

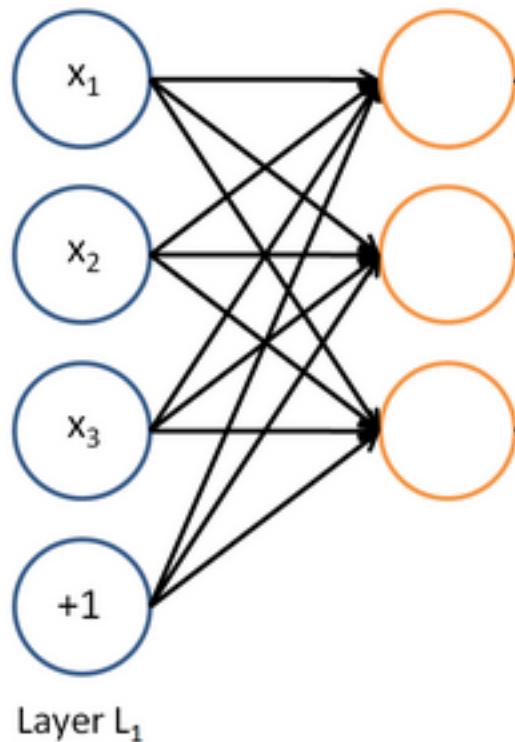
over training pairs  $(x, y)$ , by gradient-based optimization

- This is a very shallow neural network (no hidden layer)



# A neural network = running several logistic regressions at the same time

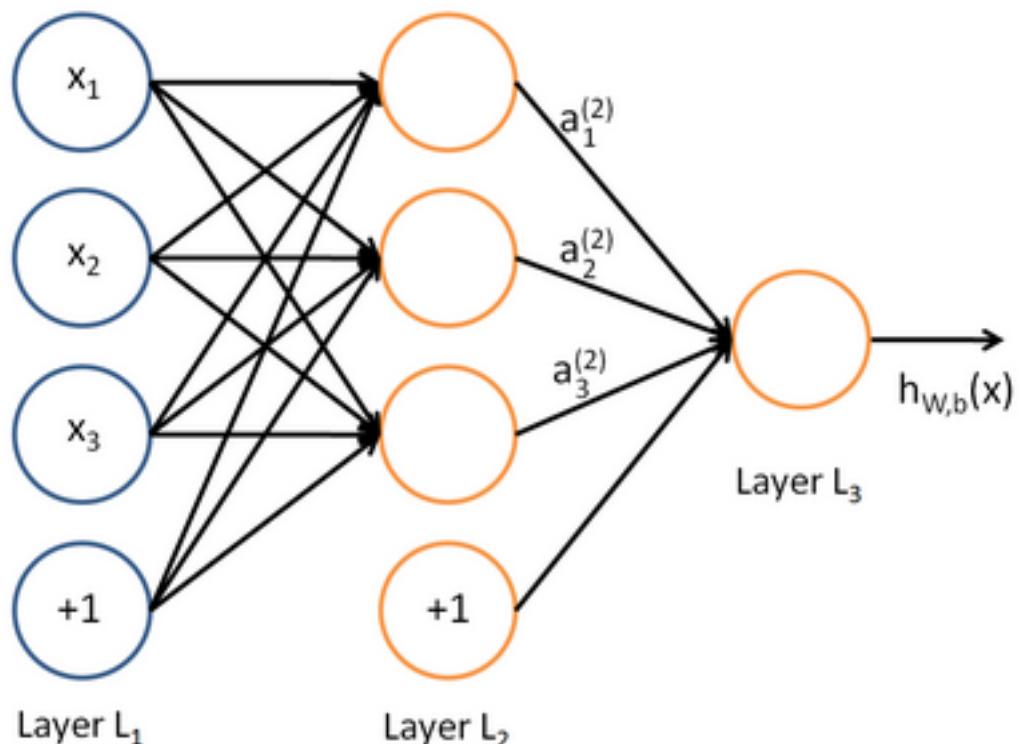
If we feed a vector of inputs through a bunch of logistic regression functions, then we get a vector of outputs



But we don't have to decide ahead of time what variables these logistic regressions are trying to predict!

A neural network = running several logistic regressions at the same time

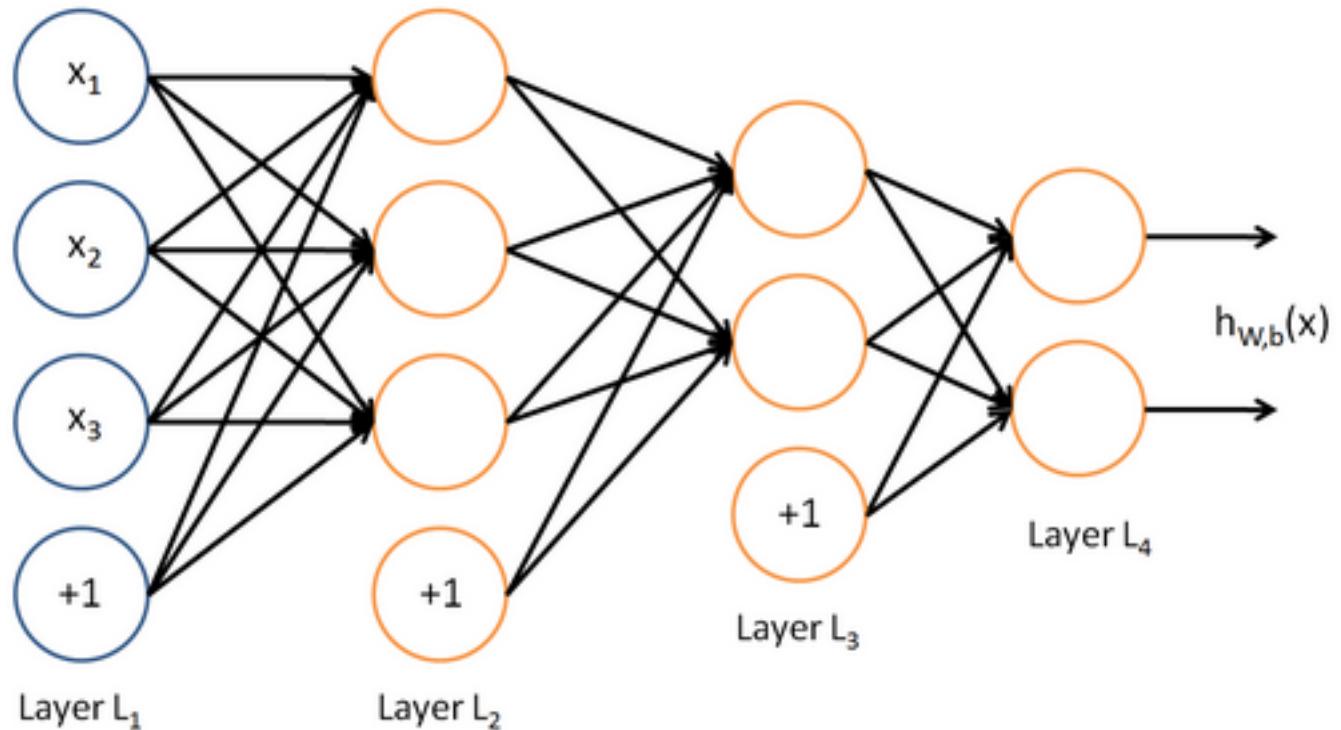
... which we can feed into another logistic regression function



and it is the training criterion that will decide what those intermediate binary target variables should be, so as to make a good job of predicting the targets for the next layer, etc.

A neural network = running several logistic regressions at the same time

- Before we know it, we have a multilayer neural network....



# Back-Prop

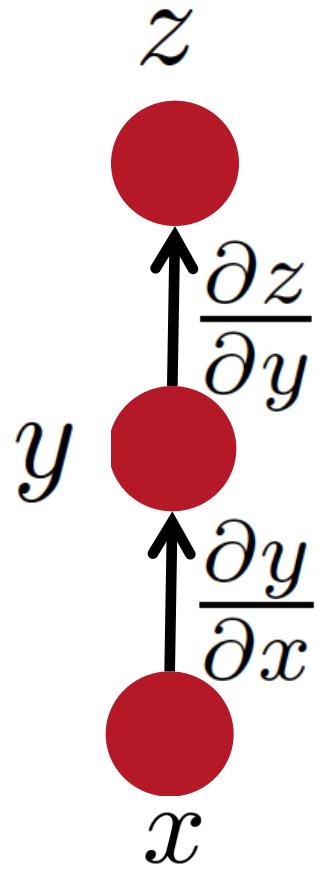
- Compute gradient of example-wise loss wrt parameters

- Simply applying the derivative chain rule wisely

$$z = f(y) \quad y = g(x) \quad \frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}$$

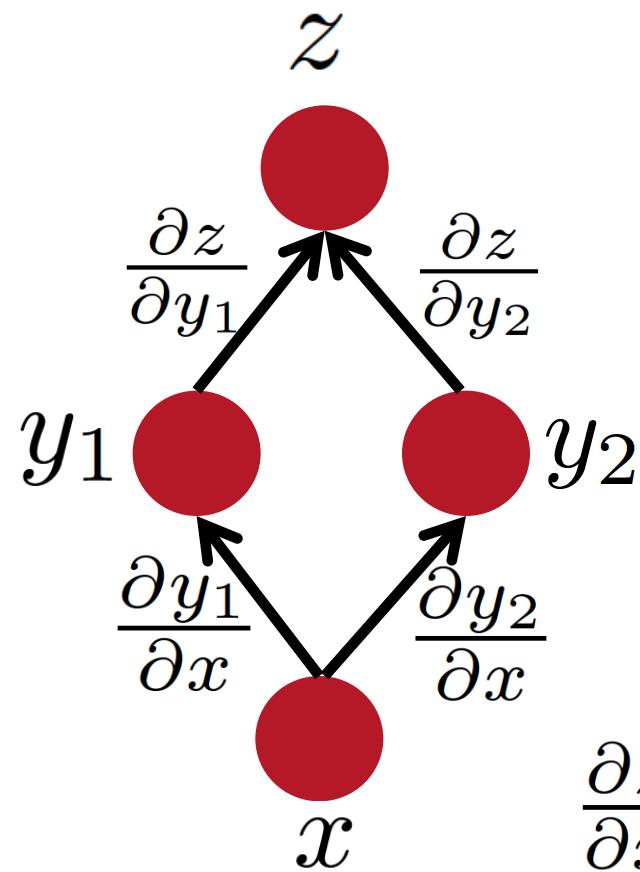
- *If computing the loss(example, parameters) is  $O(n)$  computation, then so is computing the gradient*

## Simple Chain Rule



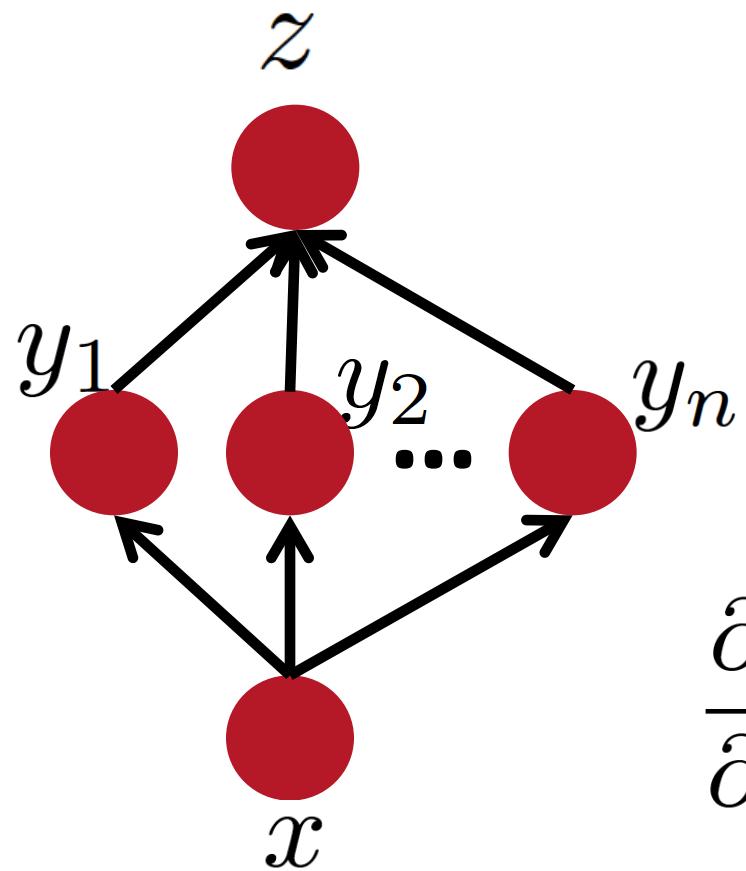
$$\Delta z = \frac{\partial z}{\partial y} \Delta y$$
$$\Delta y = \frac{\partial y}{\partial x} \Delta x$$
$$\Delta z = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x} \Delta x$$
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}$$

## Multiple Paths Chain Rule



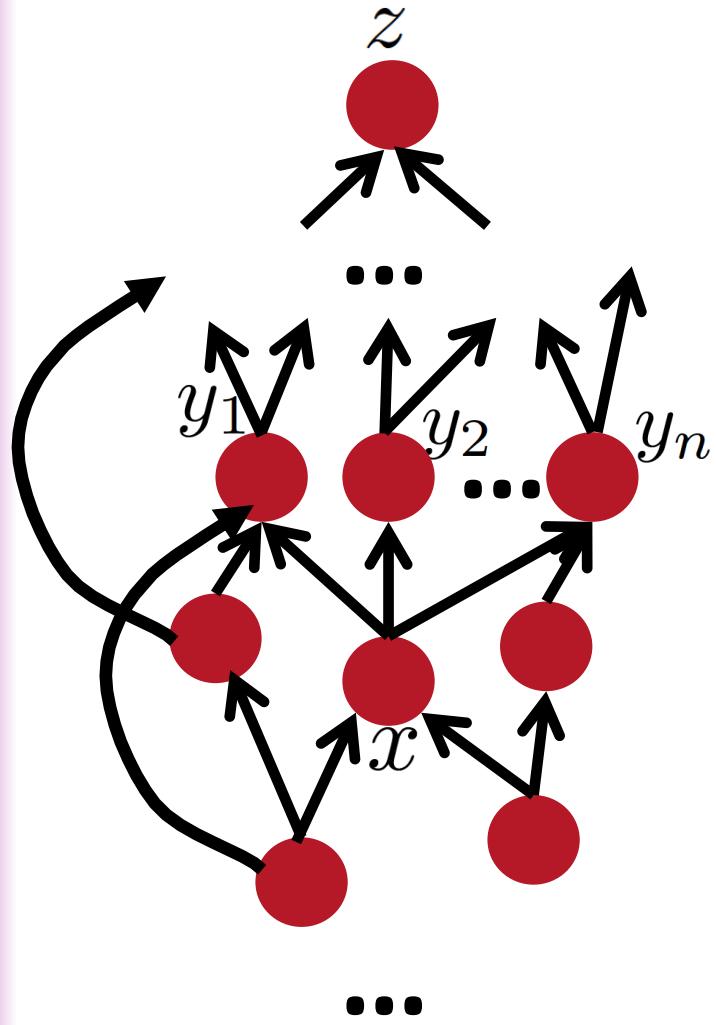
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1} \frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2} \frac{\partial y_2}{\partial x}$$

## Multiple Paths Chain Rule - General



$$\frac{\partial z}{\partial x} = \sum_{i=1}^n \frac{\partial z}{\partial y_i} \frac{\partial y_i}{\partial x}$$

## Chain Rule in Flow Graph



Flow graph: any directed acyclic graph

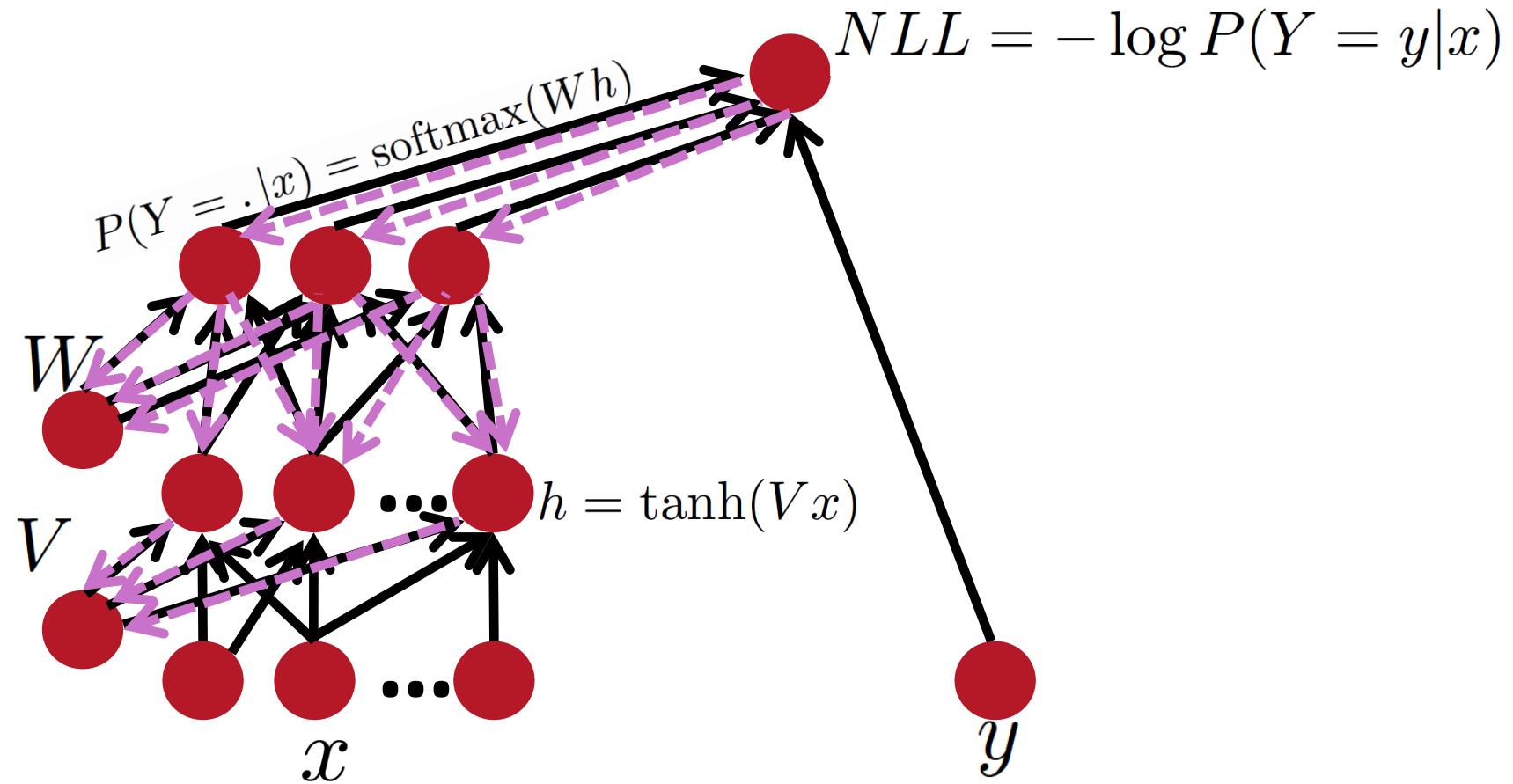
node = computation result

arc = computation dependency

$\{y_1, y_2, \dots, y_n\}$  = successors of  $x$

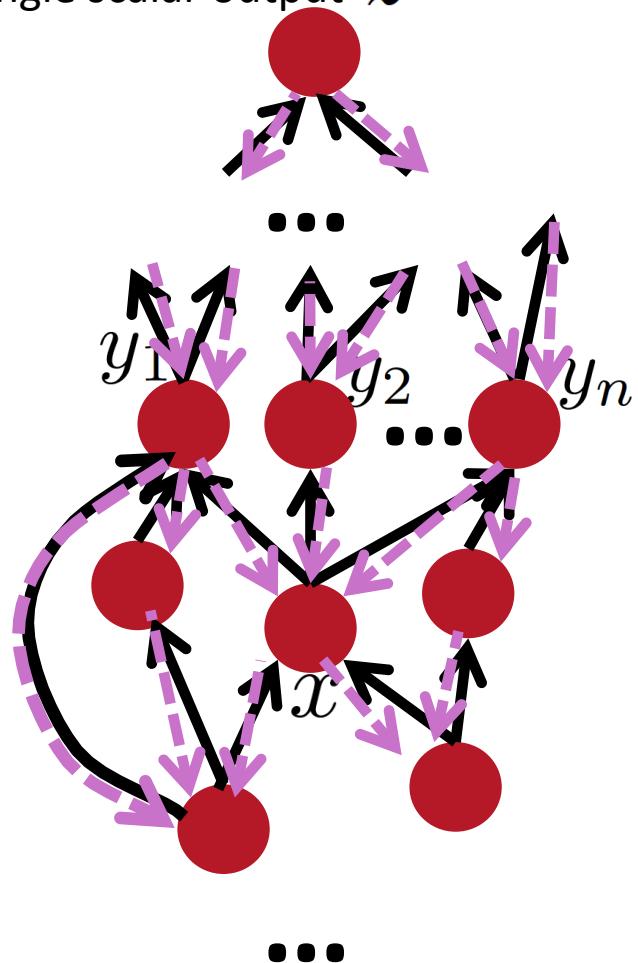
$$\frac{\partial z}{\partial x} = \sum_{i=1}^n \frac{\partial z}{\partial y_i} \frac{\partial y_i}{\partial x}$$

# Back-Prop in Multi-Layer Net



# Back-Prop in General Flow Graph

Single scalar output  $z$



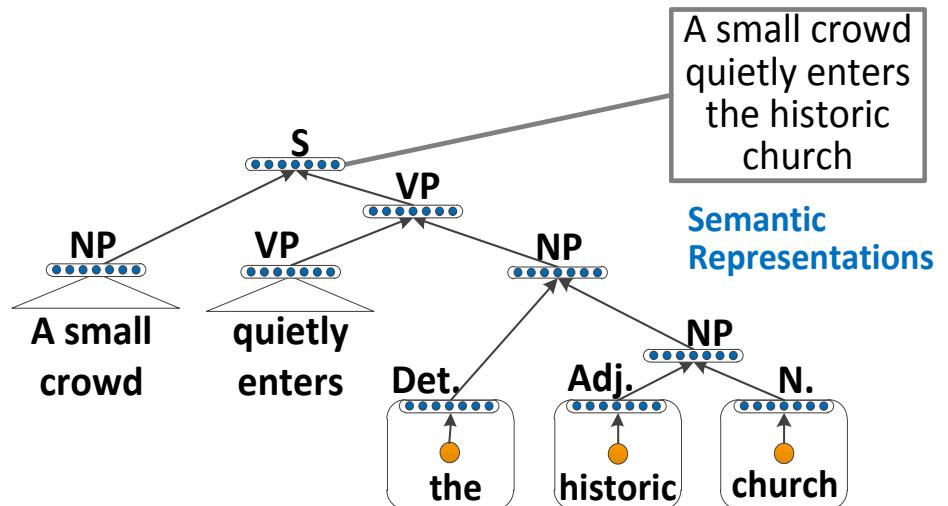
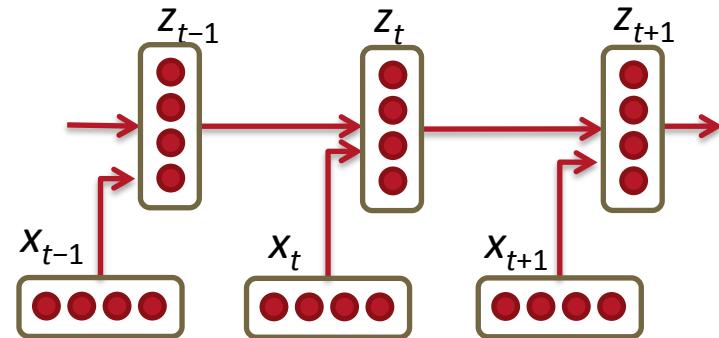
1. Fprop: visit nodes in topo-sort order
  - Compute value of node given predecessors
2. Bprop:
  - initialize output gradient = 1
  - visit nodes in reverse order:  
Compute gradient wrt each node using gradient wrt successors

$\{y_1, y_2, \dots, y_n\}$  = successors of  $x$

$$\frac{\partial z}{\partial x} = \sum_{i=1}^n \frac{\partial z}{\partial y_i} \frac{\partial y_i}{\partial x}$$

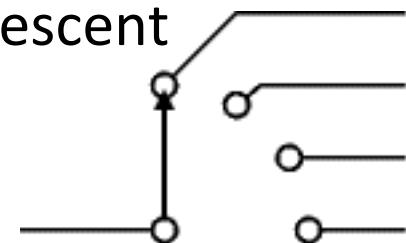
# Back-Prop in Recurrent & Recursive Nets

- Replicate a parameterized function over different time steps or nodes of a DAG
- Output state at one time-step / node is used as input for another time-step / node

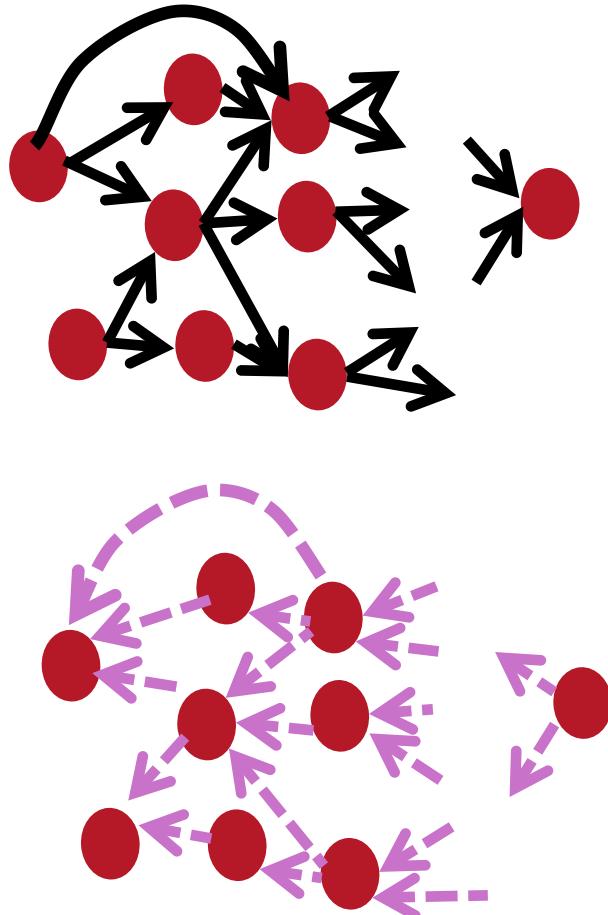


# Backpropagation Through Structure

- Inference → discrete choices
  - (e.g., shortest path in HMM, best output configuration in CRF)
- E.g. Max over configurations or sum weighted by posterior
- The loss to be optimized depends on these choices
- The inference operations are flow graph nodes
- If continuous, can perform stochastic gradient descent
  - $\text{Max}(a,b)$  is continuous.



# Automatic Differentiation

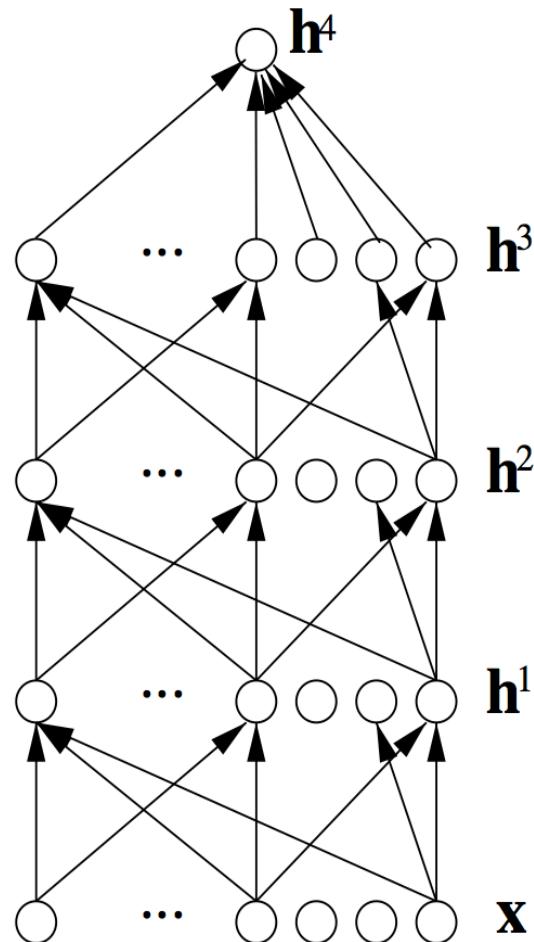


- The gradient computation can be automatically inferred from the symbolic expression of the fprop.
- Each node type needs to know how to compute its output and how to compute the gradient wrt its inputs given the gradient wrt its output.
- Easy and fast prototyping

theano

# Deep Supervised Neural Nets

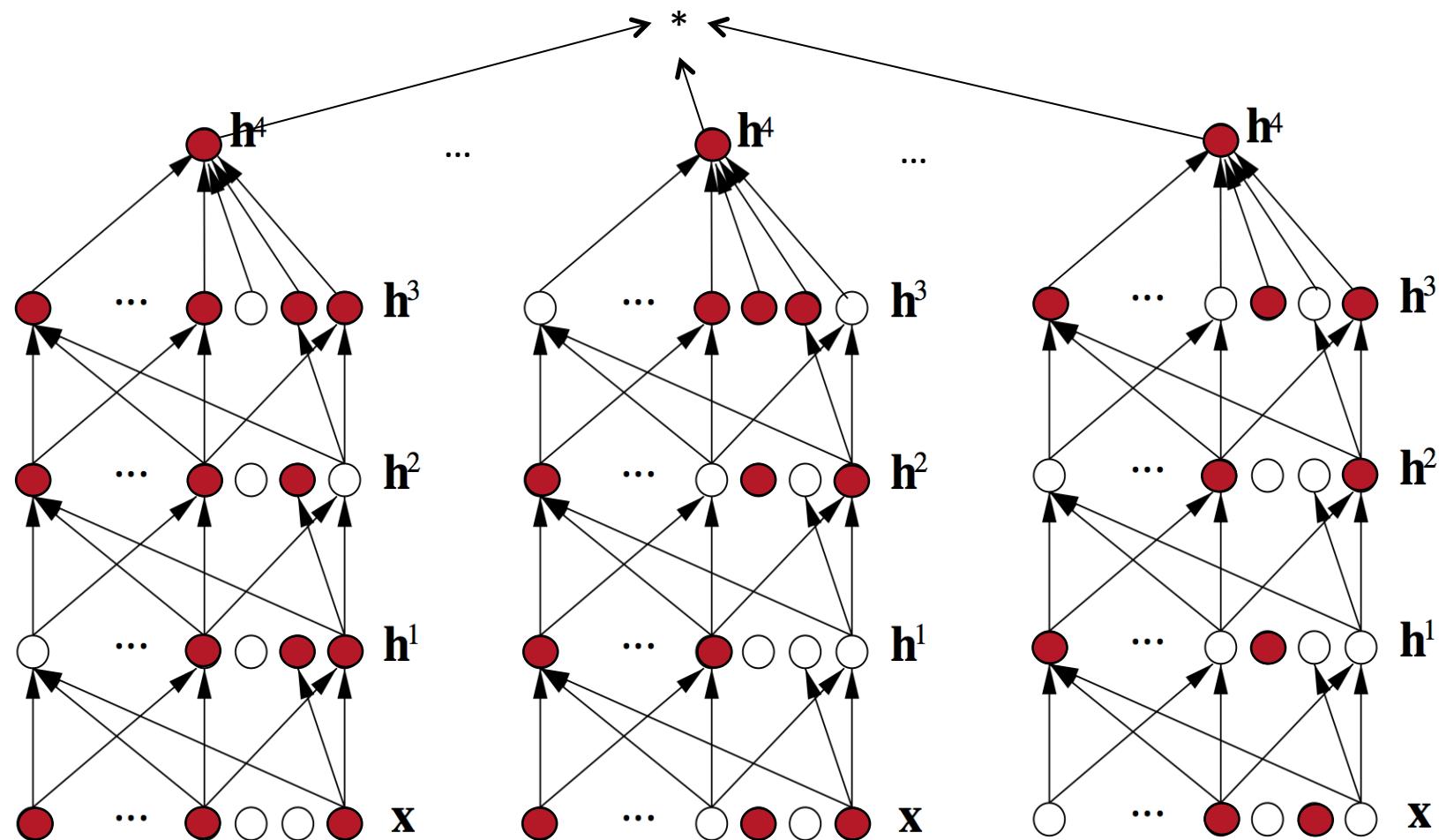
- We can now train them even without unsupervised pre-training, thanks to better initialization and non-linearities (rectifiers, maxout) and they can generalize well with large labeled sets and dropout.
- Unsupervised pre-training still useful for rare classes, transfer, smaller labeled sets, or as an extra regularizer.



# **Stochastic Neurons as Regularizer:** Improving neural networks by preventing co-adaptation of feature detectors (**Hinton et al 2012, arXiv**)

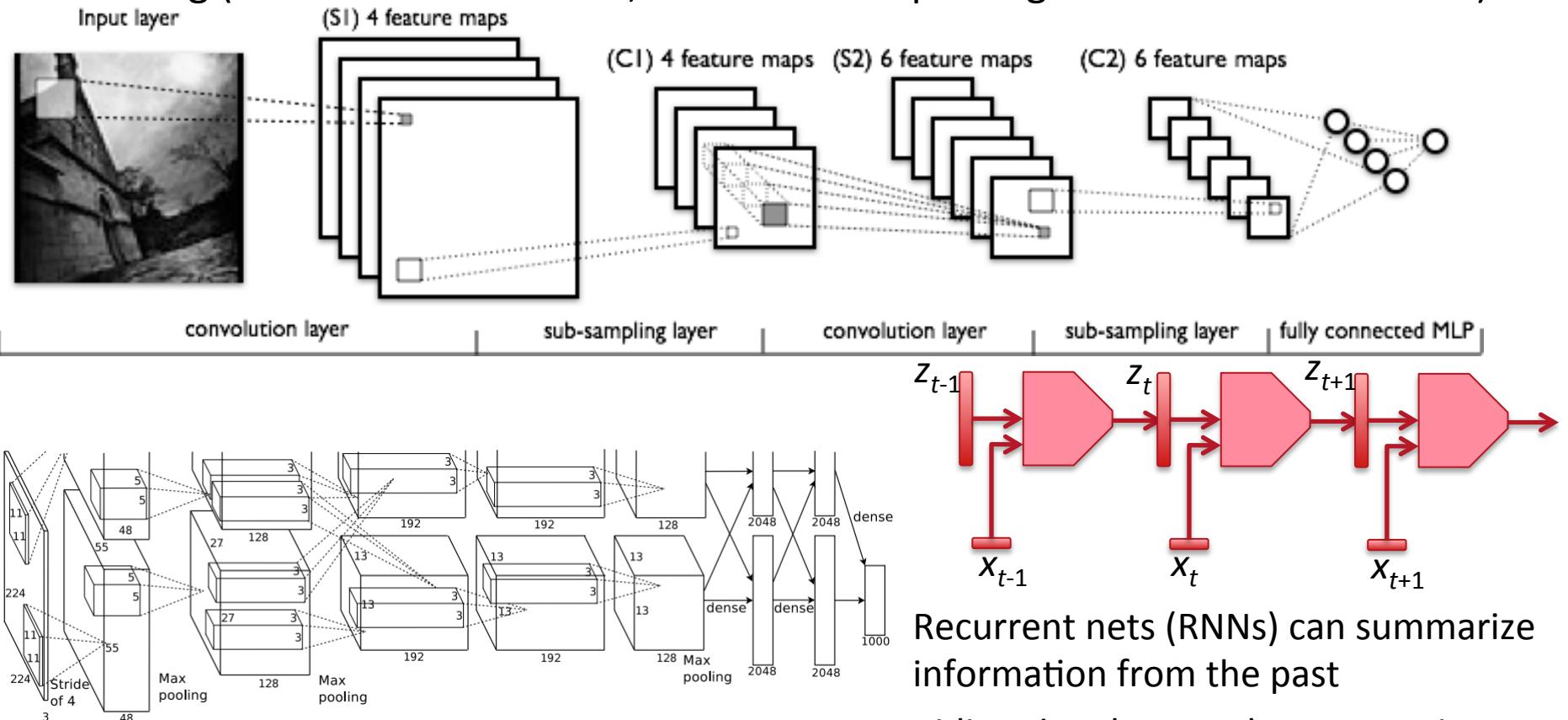
- **Dropouts** trick: during training multiply neuron output by random bit ( $p=0.5$ ), during test by 0.5
- Used in deep supervised networks
- Similar to denoising auto-encoder, but corrupting every layer
- Works better with some non-linearities (rectifiers, maxout)  
(Goodfellow et al. ICML 2013)
- Equivalent to averaging over exponentially many architectures
  - Used by Krizhevsky et al to break through ImageNet SOTA
  - Also improves SOTA on CIFAR-10 ( $18 \rightarrow 16\%$  err)
  - Knowledge-free MNIST with DBMs ( $.95 \rightarrow .79\%$  err)
  - TIMIT phoneme classification ( $22.7 \rightarrow 19.7\%$  err)

# Dropout Regularizer: Super-Efficient Bagging



# Temporal & Spatial Inputs: Convolutional & Recurrent Nets

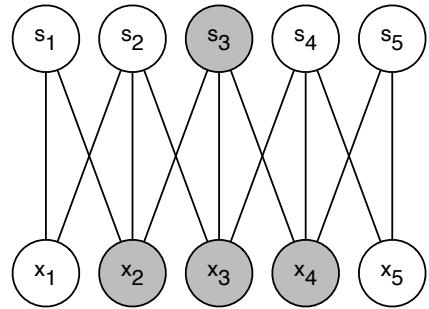
- Local connectivity across time/space
- Sharing weights across time/space (translation equivariance)
- Pooling (translation invariance, cross-channel pooling for learned invariances)



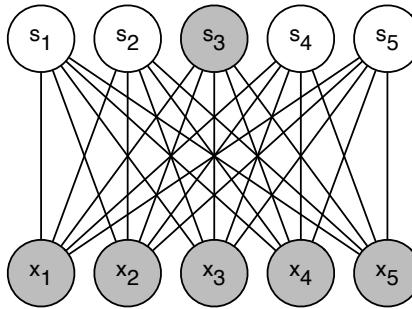
Recurrent nets (RNNs) can summarize information from the past

Bidirectional RNNs also summarize information from the future

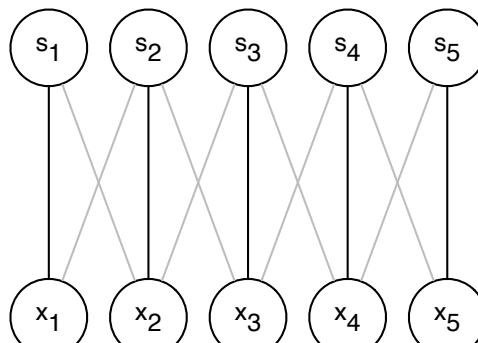
**Convolution = sparse connectivity + parameter sharing**

$$s[t] = (x * w)(t) = \sum_{a=-\infty}^{\infty} x[a]w[t-a]$$


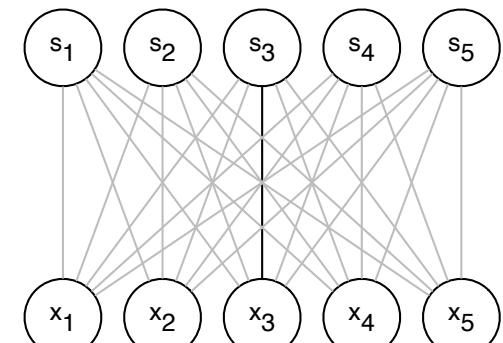
sparse



dense



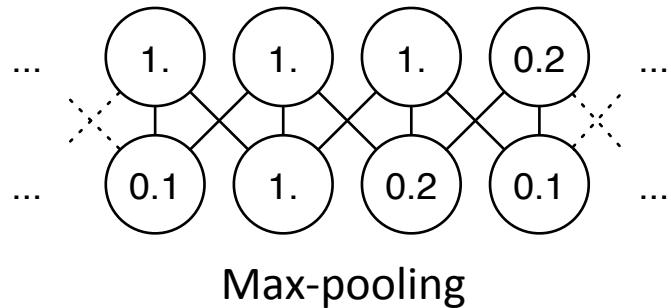
shared



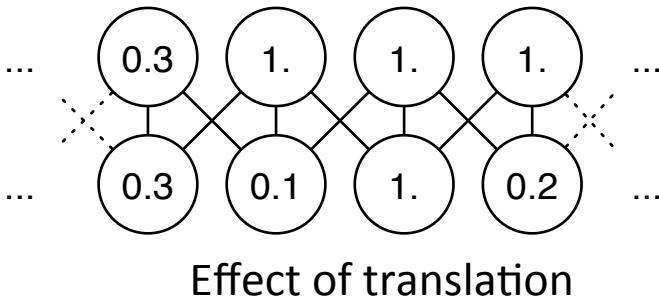
not shared

# Pooling Layers

- Aggregate to achieve local invariance

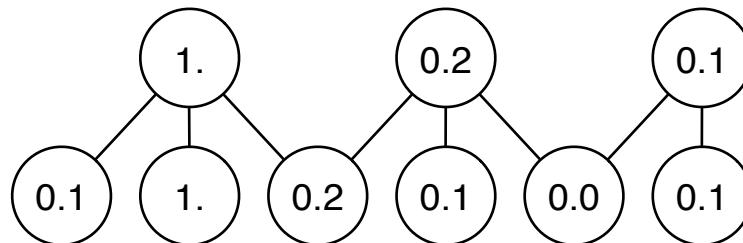


Max-pooling

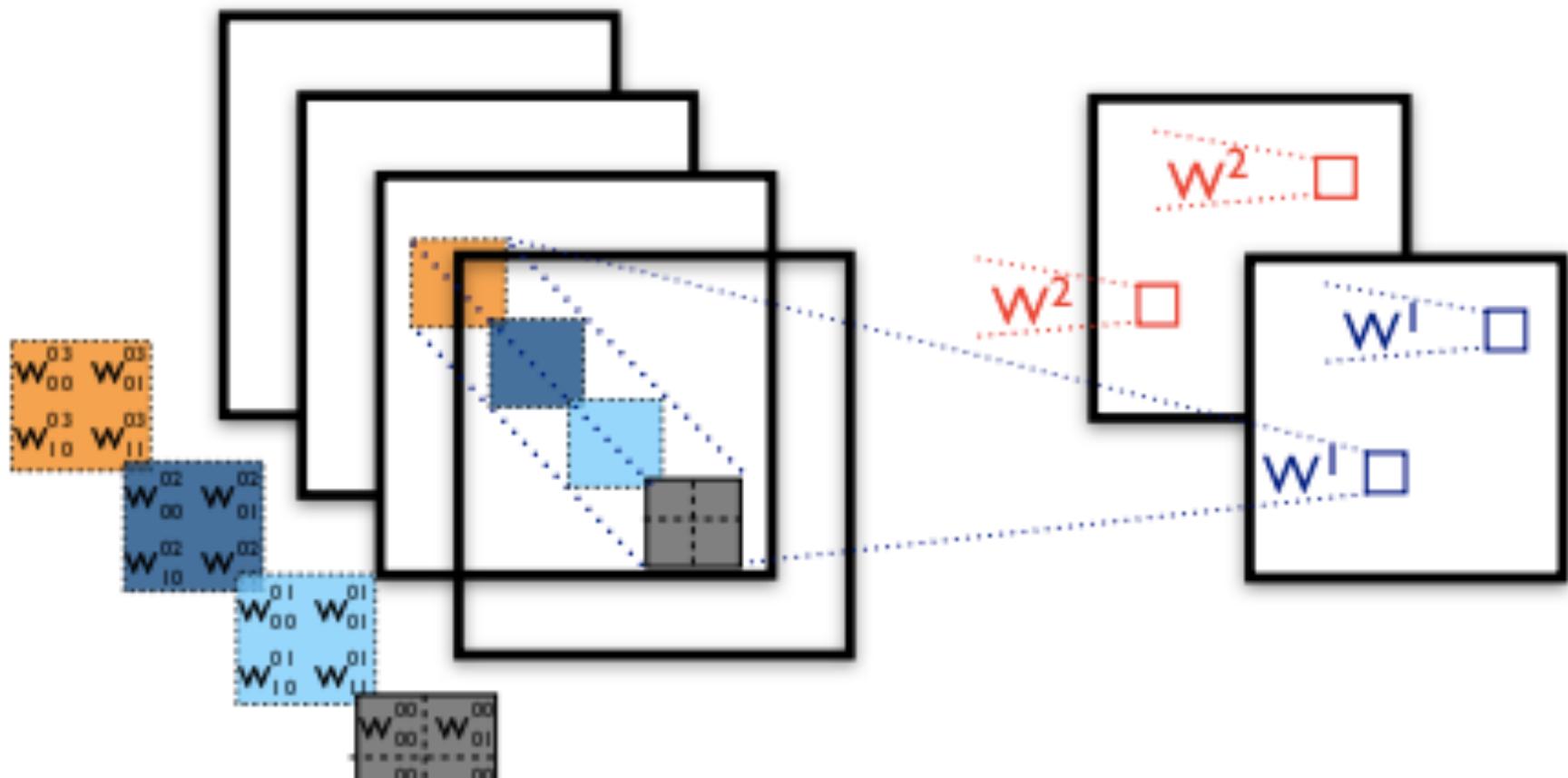


Effect of translation

- Subsampling to reduce temporal/spatial scale and computation

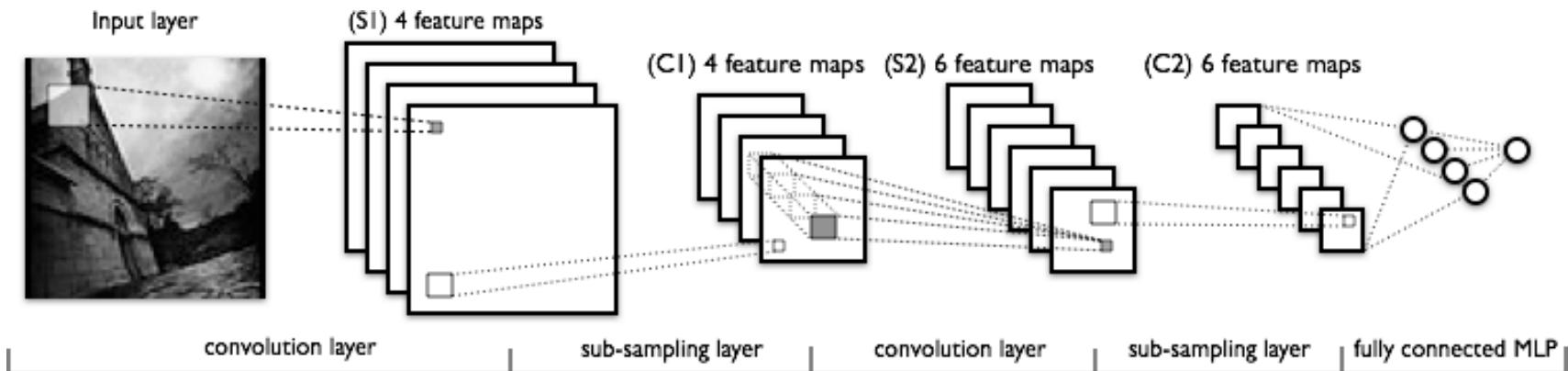


# Multiple Convolutions: Feature Maps



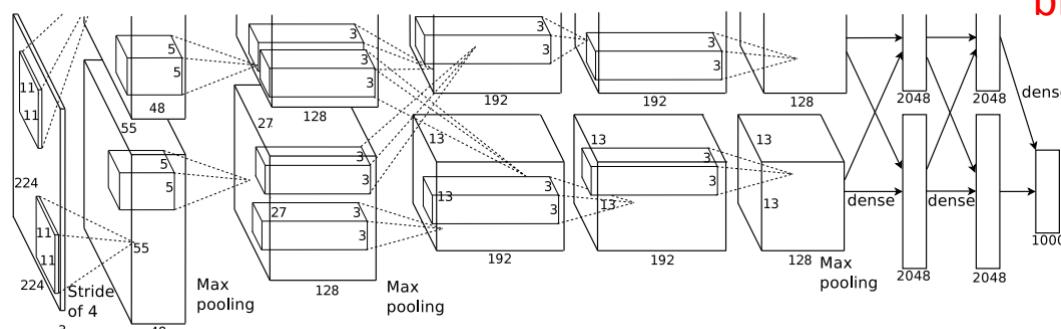
# Alternating convolutions & pooling

- Inspired by visual cortex, idea from Fukushima's Neocognitron, combined with back-prop and developed by **LeCun** since 1989



- Increasing number of features, decreasing spatial resolution
- Top layers are fully connected

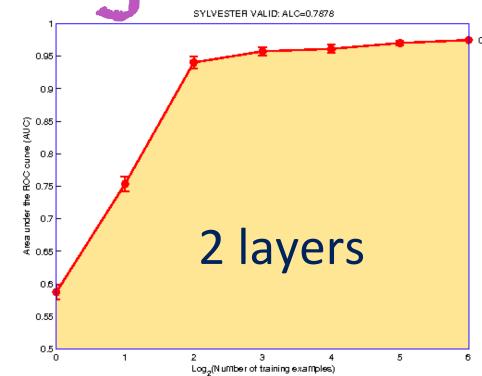
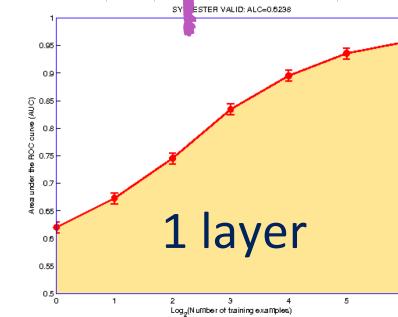
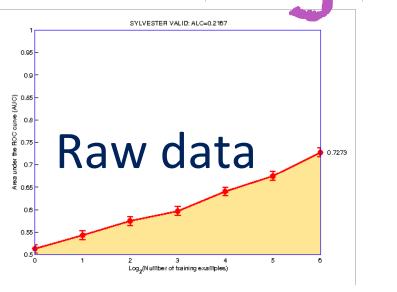
Krizhevsky, Sutskever & Hinton 2012  
breakthrough in object recognition



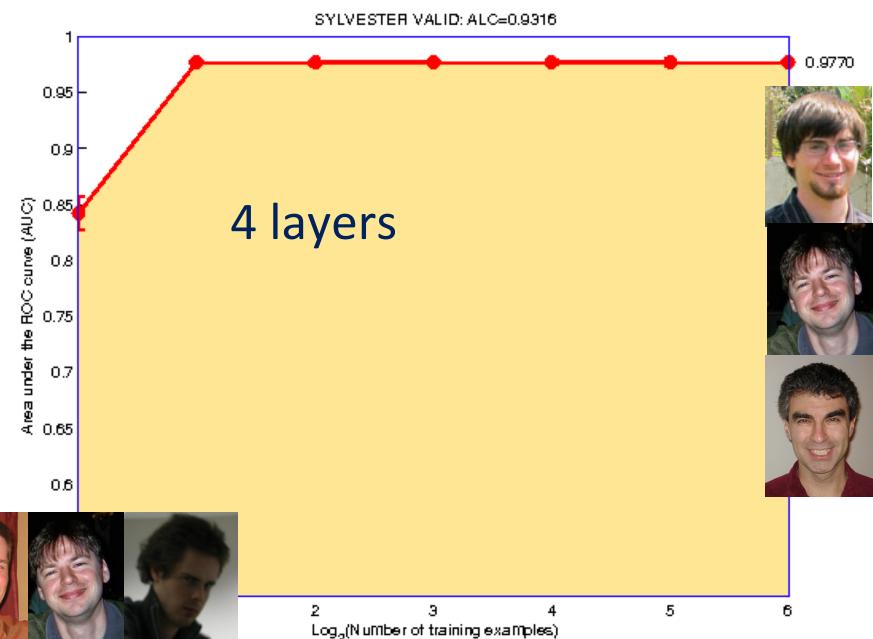
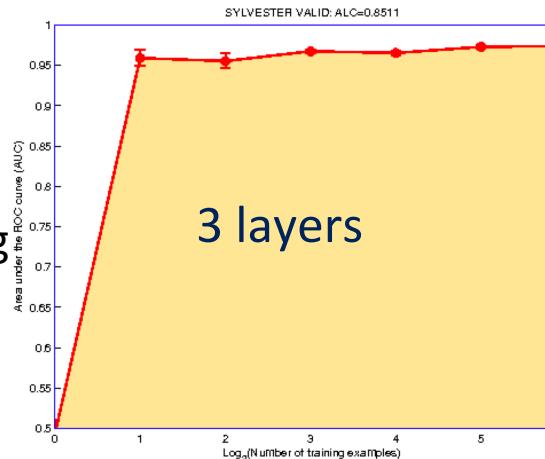
Distributed Representations  
of Neural Nets:

How to do unsupervised  
training?

# Unsupervised and Transfer Learning Challenge + Transfer Learning Challenge: Deep Learning 1st Place

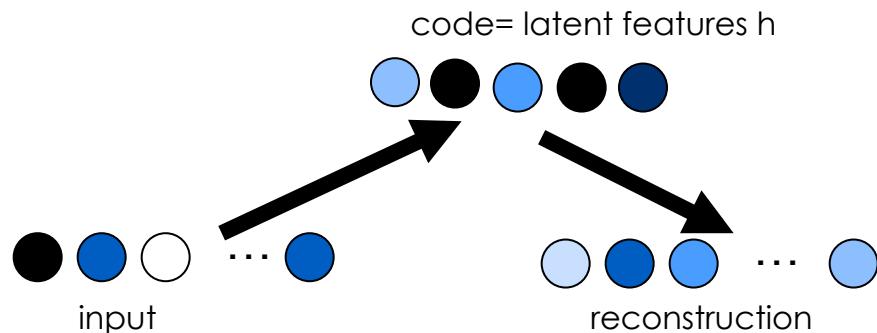


# ICML'2011 workshop on Unsup. & Transfer Learning



# PCA

= Linear Manifold  
 = Linear Auto-Encoder  
 = Linear Gaussian Factors

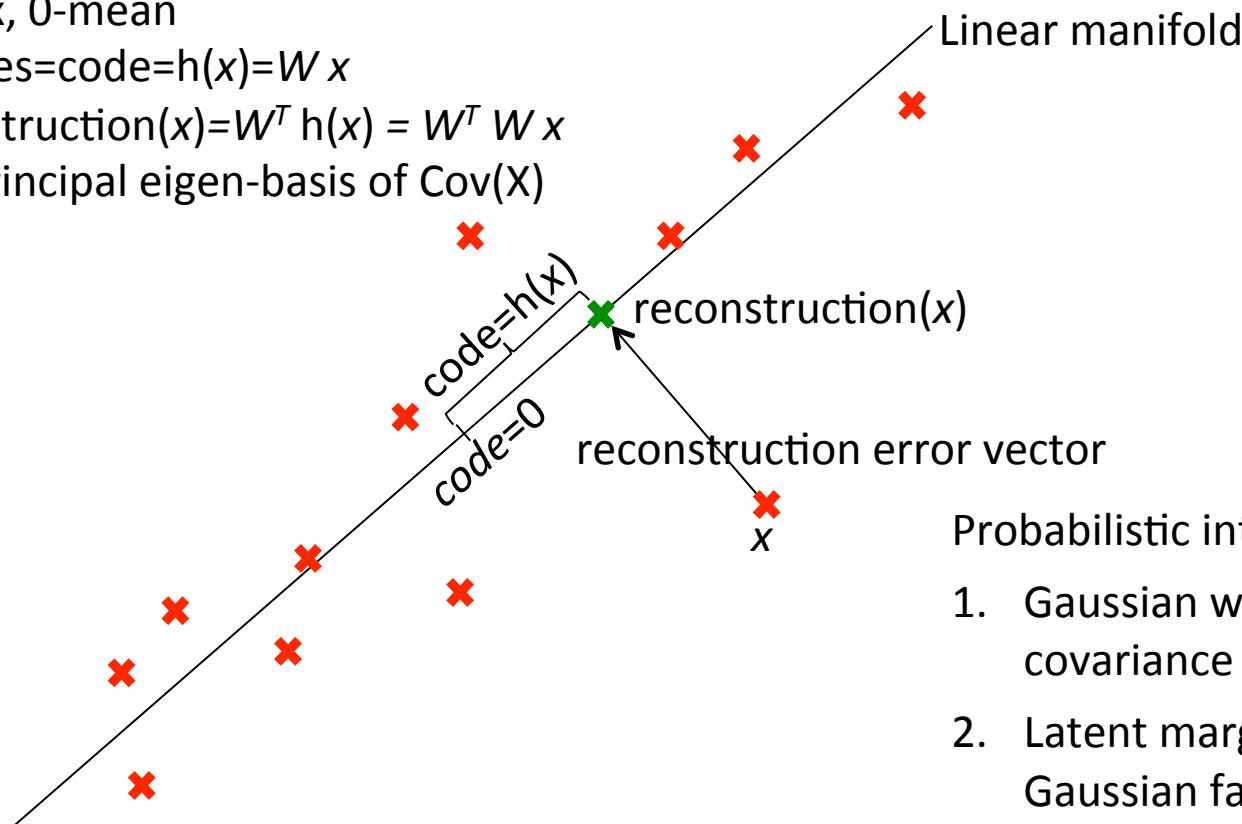


input  $x$ , 0-mean

features=code= $h(x)=Wx$

reconstruction( $x$ )= $W^T h(x) = W^T W x$

$W$  = principal eigen-basis of  $\text{Cov}(X)$



Probabilistic interpretations:

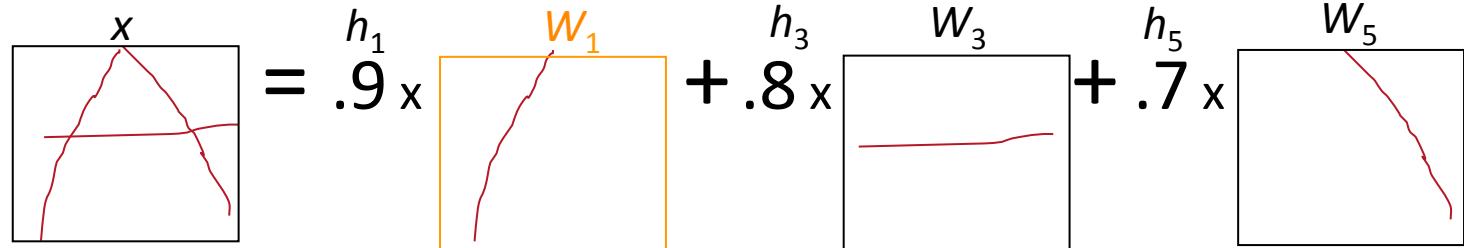
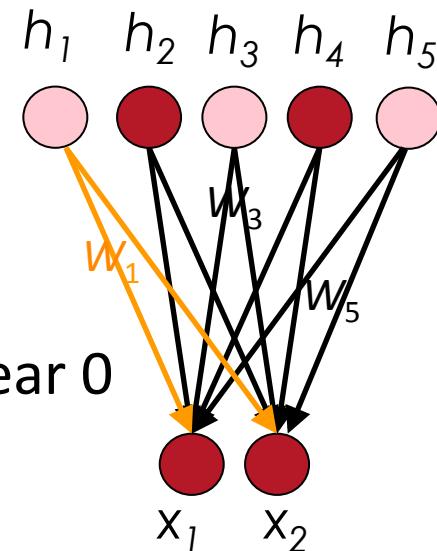
1. Gaussian with full covariance  $W^T W + \lambda I$
2. Latent marginally iid Gaussian factors  $h$  with  $x = W^T h + \text{noise}$

# Directed Factor Models:

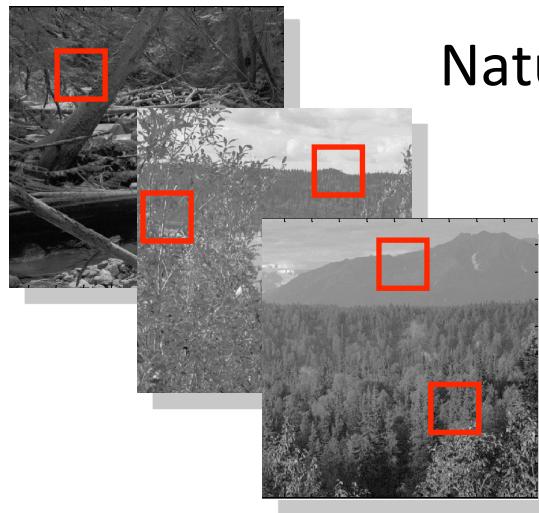
$$P(x, h) = P(h)P(x|h)$$

factors prior      likelihood

- $P(h)$  factorizes into  $P(h_1) P(h_2) \dots$
- Different priors:
  - PCA:  $P(h_i)$  is Gaussian
  - ICA:  $P(h_i)$  is non-parametric
  - **Sparse coding**:  $P(h_i)$  is concentrated near 0
- Likelihood is typically Gaussian  $x | h$   
with mean given by  $W^T h$
- **Inference** procedures (predicting  $h$ , given  $x$ ) differ
- Sparse  $h$ :  $x$  is explained by the weighted addition of selected filters  $h_i$

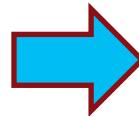


# Sparse autoencoder illustration for images

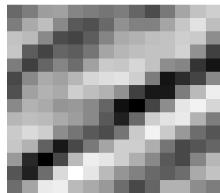


Natural Images

Learned bases:



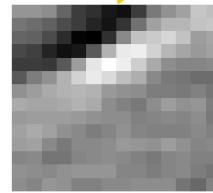
Test example



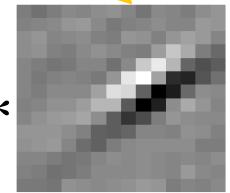
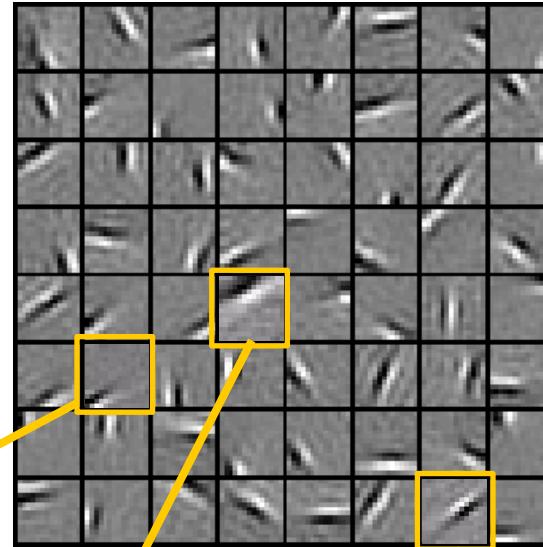
$$\approx 0.8 * \quad$$



$$+ 0.3 * \quad$$



$$+ 0.5 * \quad$$

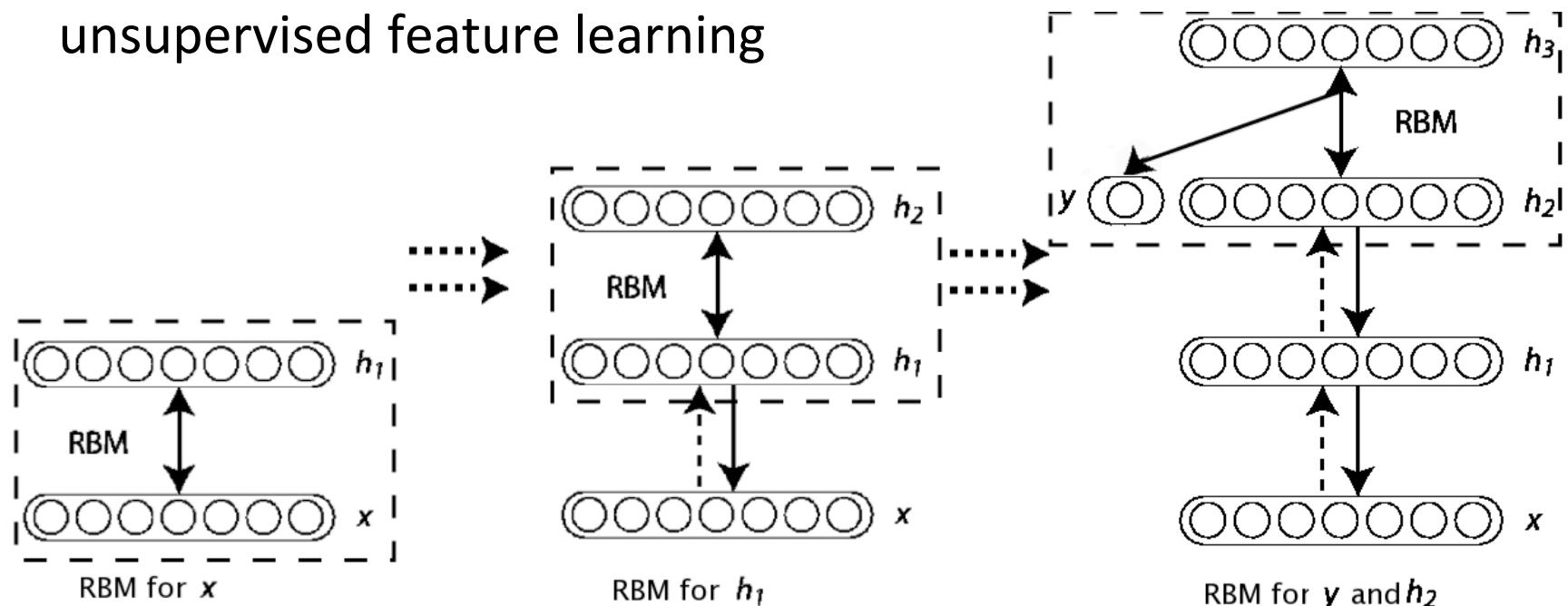


$$[h_1, \dots, h_{64}] = [0, 0, \dots, 0, \mathbf{0.8}, 0, \dots, 0, \mathbf{0.3}, 0, \dots, 0, \mathbf{0.5}, 0]$$

(feature representation)

# Stacking Single-Layer Learners

- PCA is great but can't be stacked into deeper more abstract representations (linear  $\times$  linear = linear)
- One of the big ideas from Hinton et al. 2006: layer-wise unsupervised feature learning



Stacking Restricted Boltzmann Machines (RBM)  $\rightarrow$  Deep Belief Network (DBN)

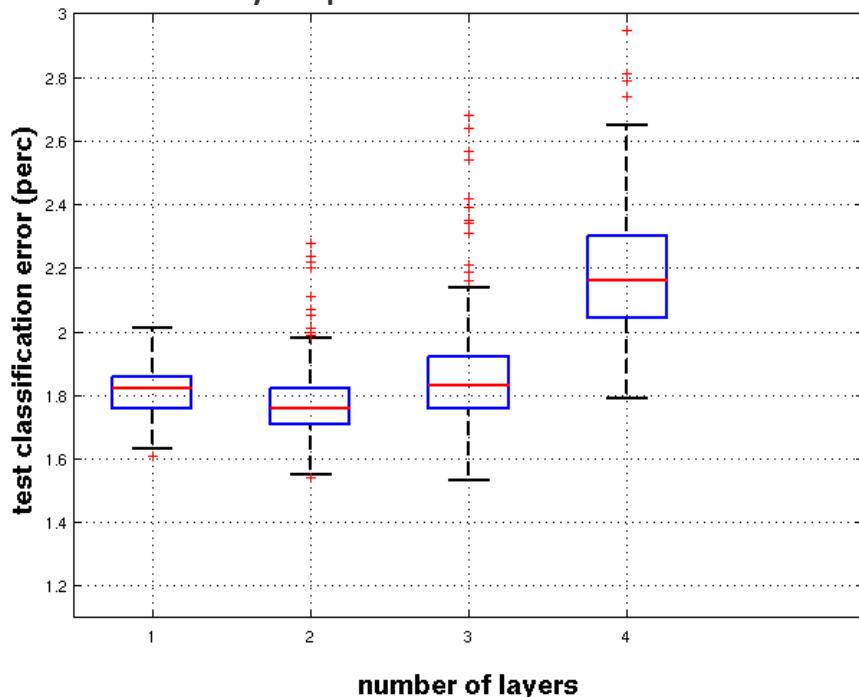
# Effective deep Learning first became possible with unsupervised pre-training

[Erhan et al., JMLR 2010]

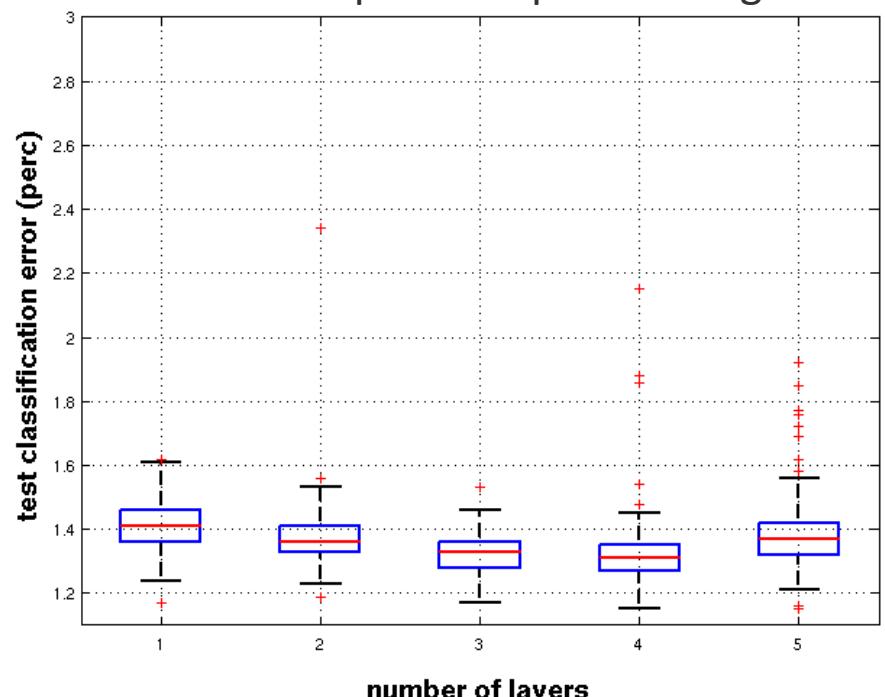


(with RBMs and Denoising Auto-Encoders)

Purely supervised neural net



With unsupervised pre-training

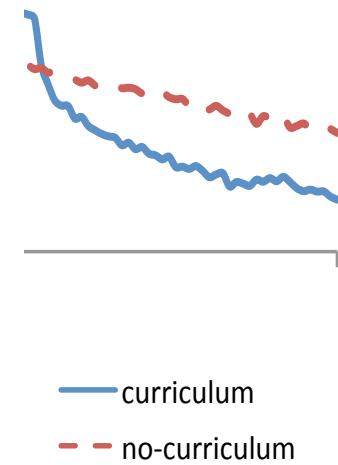


# Order & Selection of Examples Matters

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



- Curriculum learning
  - (Bengio et al 2009, Krueger & Dayan 2009)
  - Start with easier examples
- Faster convergence to a better local minimum in deep architectures

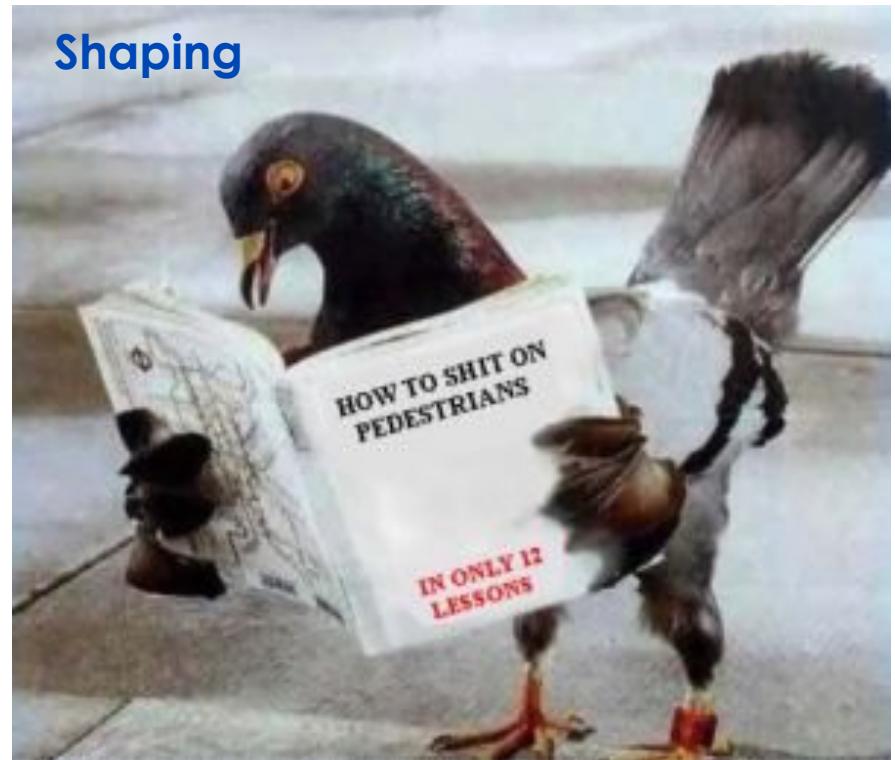


# Curriculum Learning

Guided learning helps training humans and animals

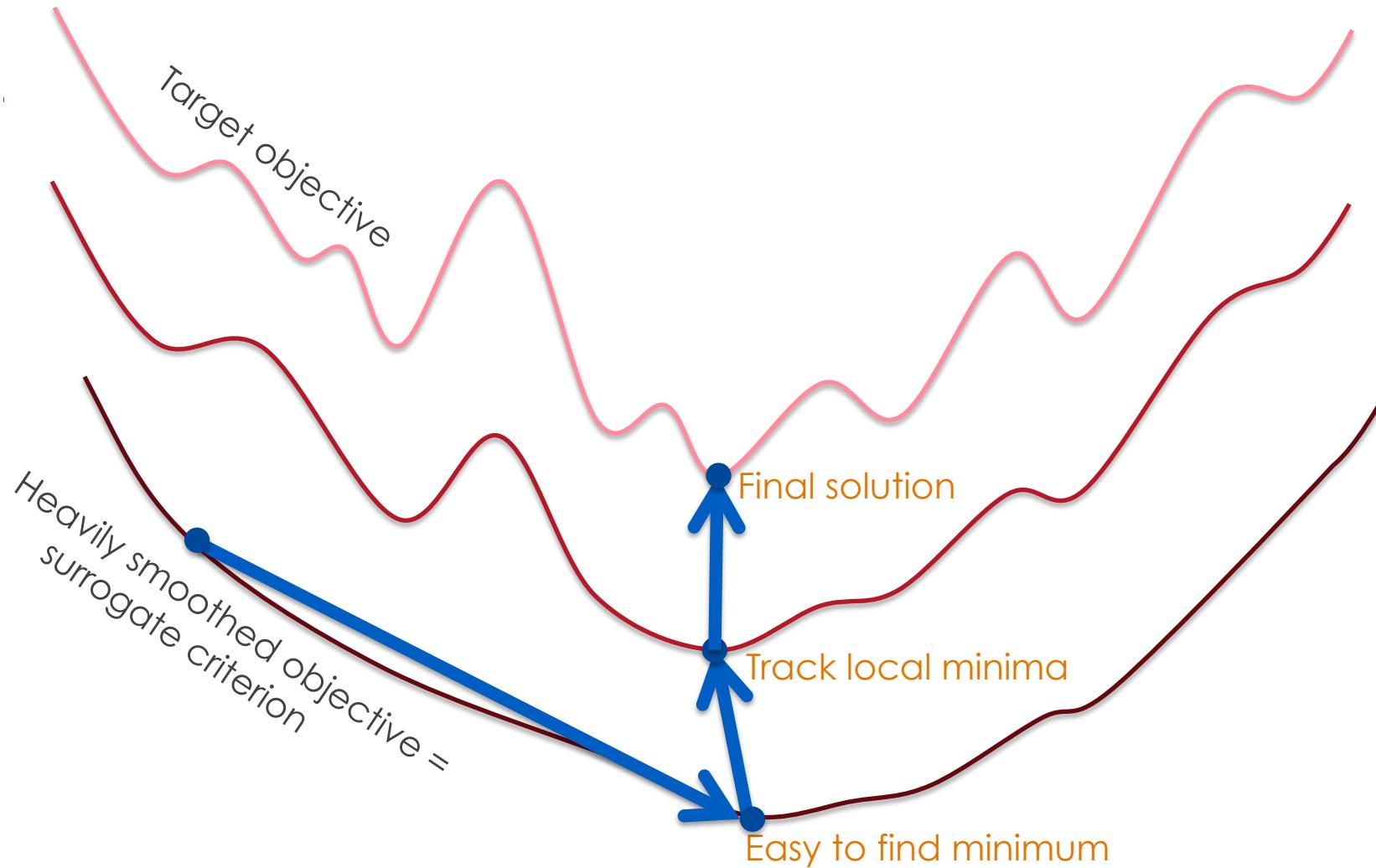


**Education**



Start from simpler examples / easier tasks (Piaget 1952, Skinner 1958)

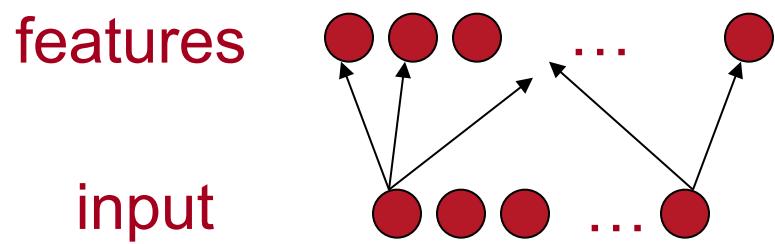
# Continuation Methods



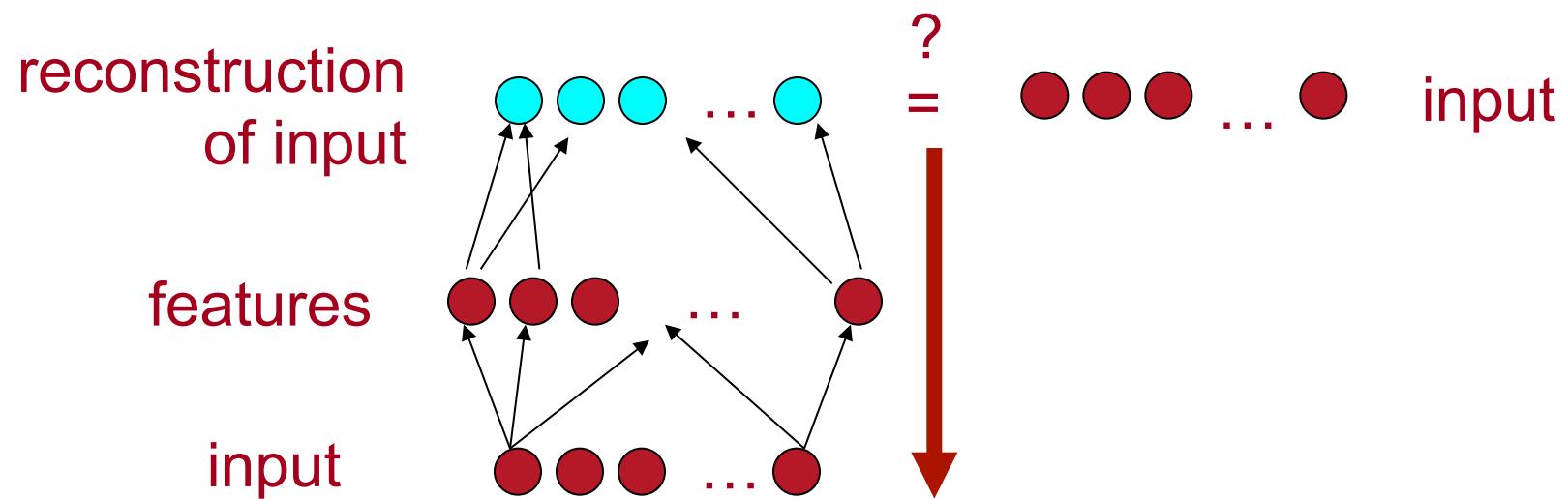
# Layer-wise Unsupervised Learning

input      

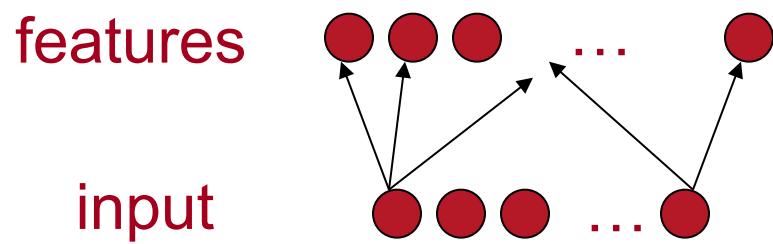
# Layer-Wise Unsupervised Pre-training



# Layer-Wise Unsupervised Pre-training

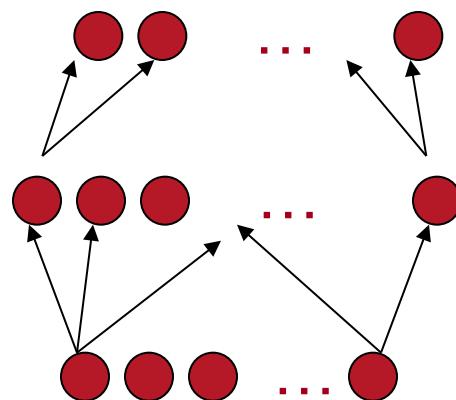


# Layer-Wise Unsupervised Pre-training

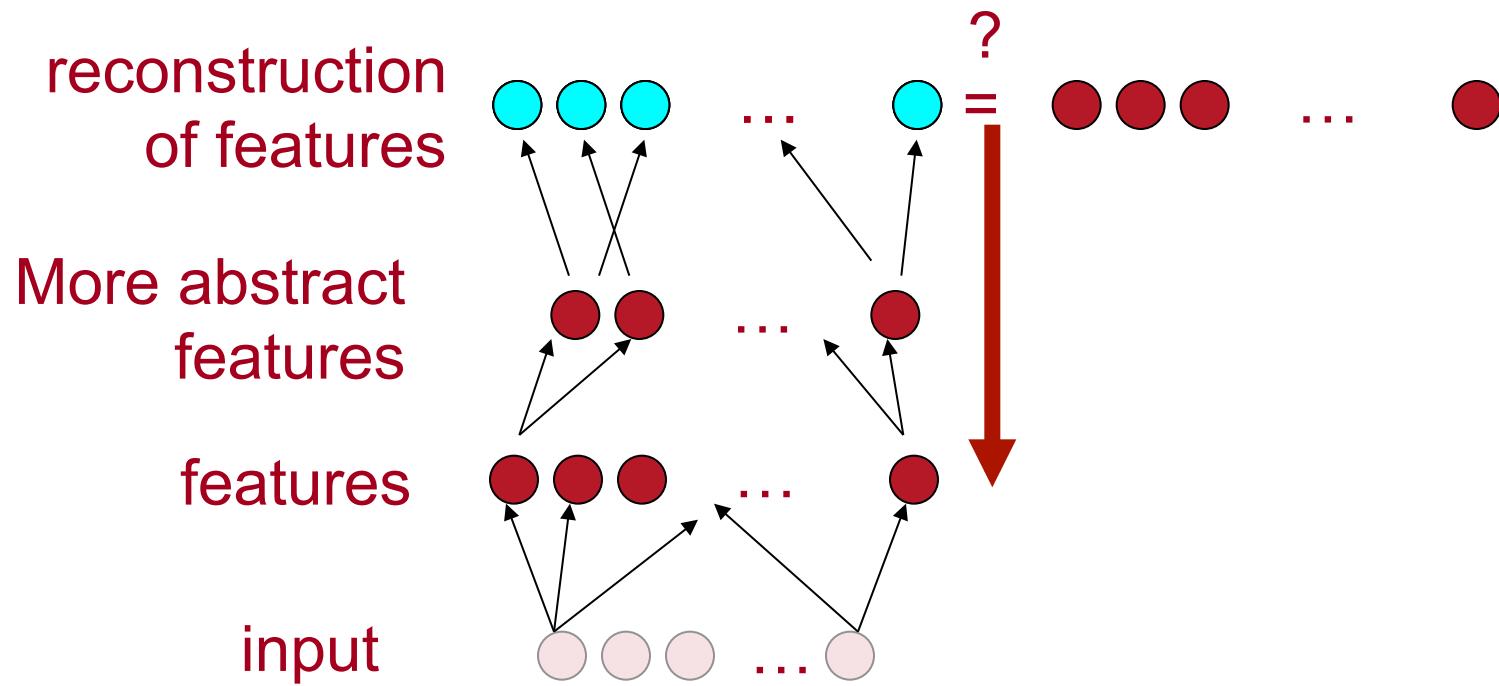


# Layer-Wise Unsupervised Pre-training

More abstract  
features  
features  
input

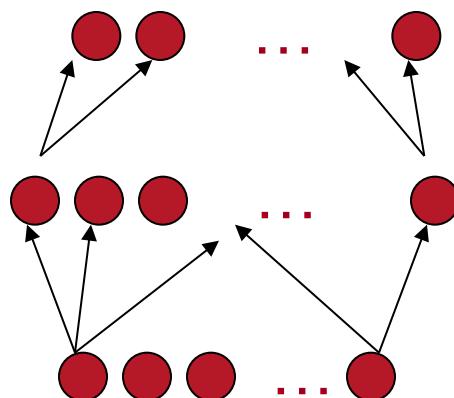


# Layer-wise Unsupervised Learning



# Layer-Wise Unsupervised Pre-training

More abstract  
features  
features  
input



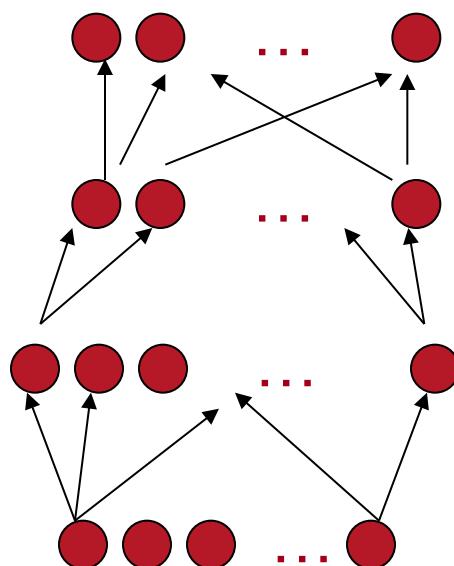
# Layer-wise Unsupervised Learning

Even more abstract  
features

More abstract  
features

features

input



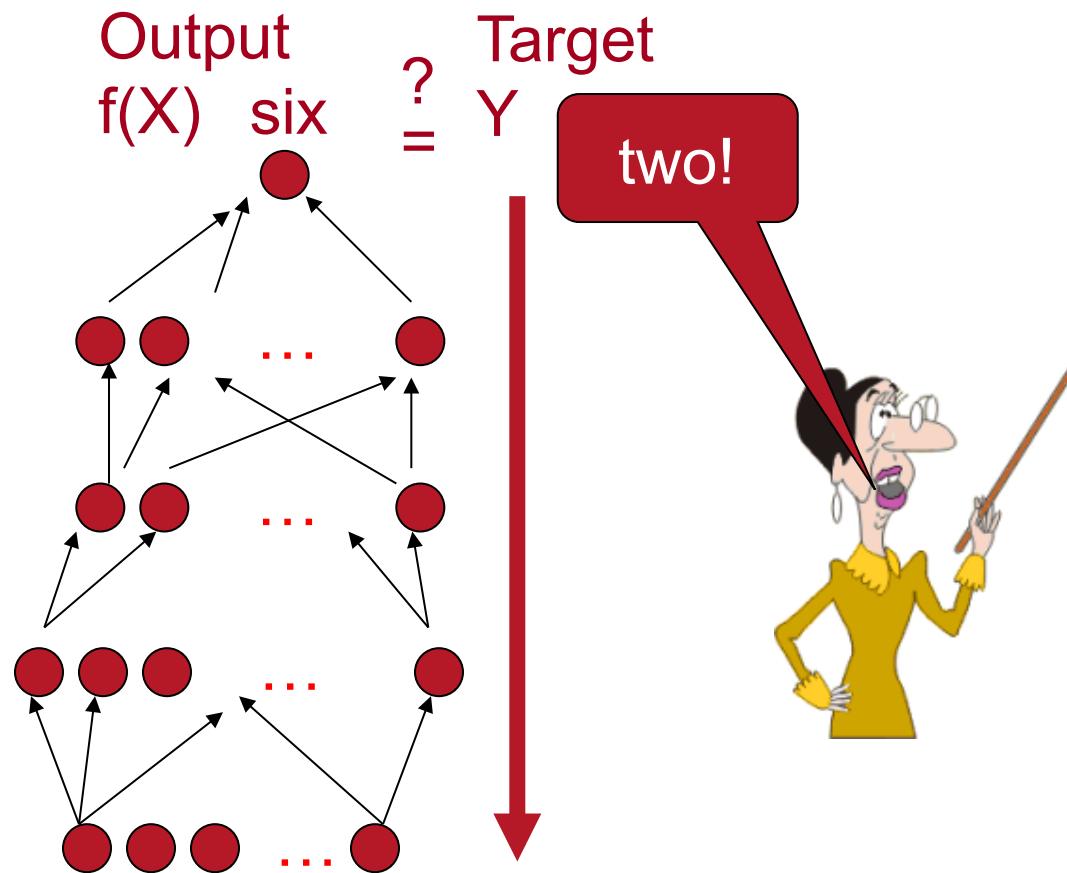
# Supervised Fine-Tuning

Even more abstract features

More abstract features

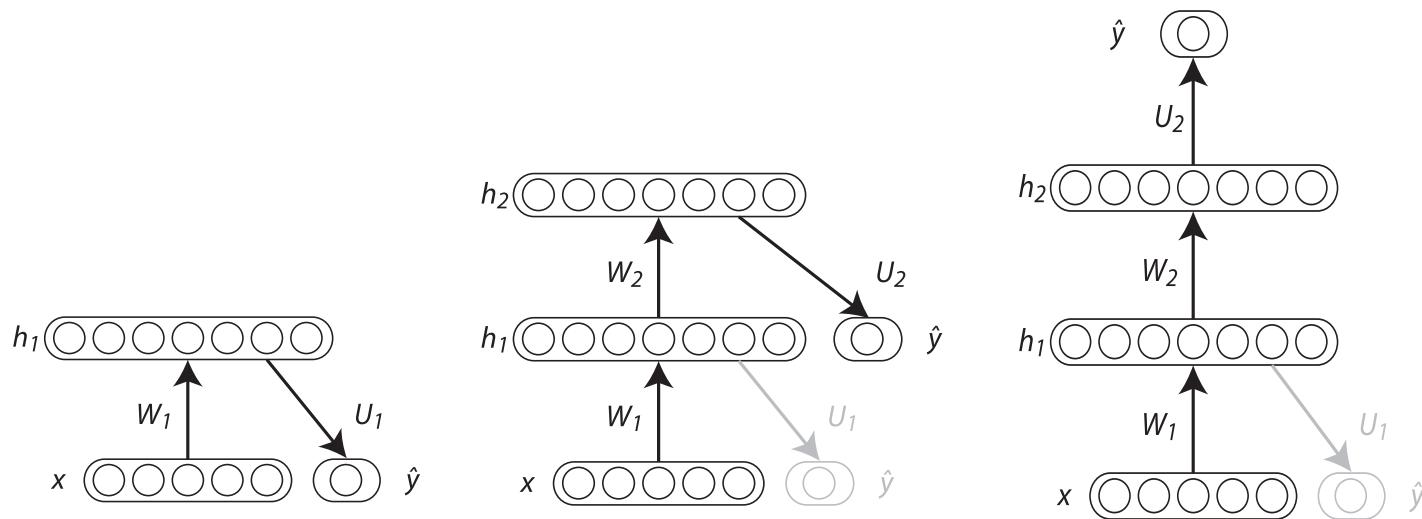
features

input



- Additional hypothesis: features good for  $P(x)$  good for  $P(y|x)$

# Greedy Layerwise Supervised Training



Generally worse than unsupervised pre-training but better than ordinary training of a deep neural network (Bengio et al. NIPS'2006). Has been used successfully on large labeled datasets, where unsupervised pre-training did not make as much of an impact.

# Why Unsupervised Learning?

- Recent progress mostly in supervised DL
- $\exists$  real challenges for unsupervised DL
- Potential benefits:
  - Exploit tons of unlabeled data
  - Answer new questions about the variables observed
  - Regularizer – transfer learning – domain adaptation
  - Easier optimization (local training signal)
  - Structured outputs

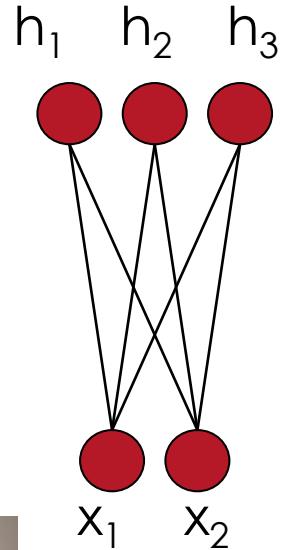
# Restricted Boltzmann Machines

# Undirected Models: the Restricted Boltzmann Machine

[Hinton et al 2006]



- Probabilistic model of the joint distribution of the observed variables (inputs alone or inputs and targets)  $x$
- Latent (hidden) variables  $h$  model high-order dependencies
- Inference is easy,  $P(h|x)$  factorizes into product of  $P(h_i | x)$
- See Bengio (2009) detailed monograph/review:  
*“Learning Deep Architectures for AI”*.
- See Hinton (2010)  
*“A practical guide to training Restricted Boltzmann Machines”*



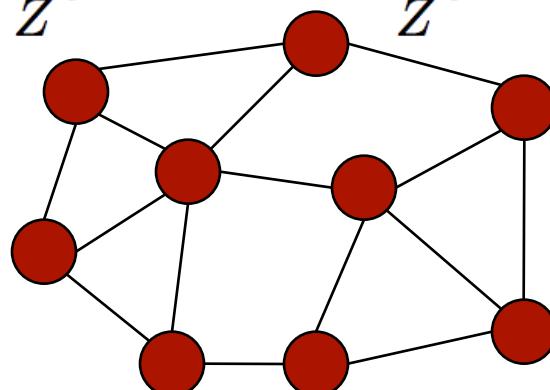
# Boltzmann Machines & MRFs

- Boltzmann machines:

$$(\text{Hinton 84}) \quad P(x) = \frac{1}{Z} e^{-\text{Energy}(x)} = \frac{1}{Z} e^{c^T x + x^T W x} = \frac{1}{Z} e^{\sum_i c_i x_i + \sum_{i,j} x_i W_{ij} x_j}$$

- Markov Random Fields:

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



Undirected  
graphical  
models

Soft constraint / probabilistic statement

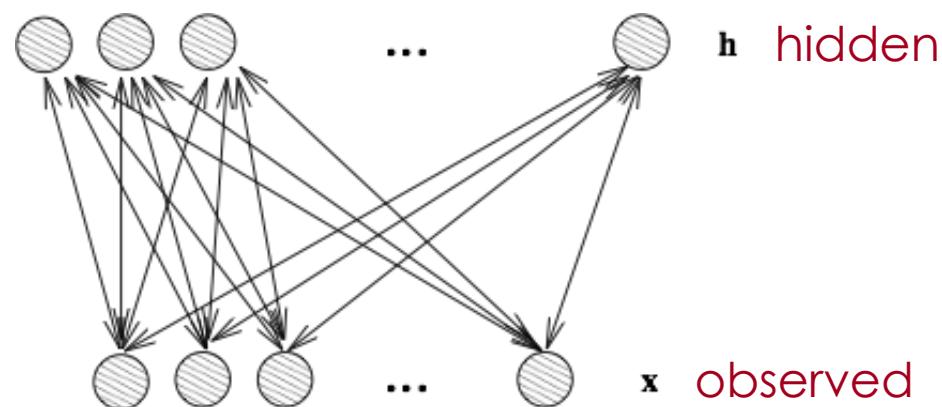
- More interesting with latent variables!

# Restricted Boltzmann Machine (RBM)

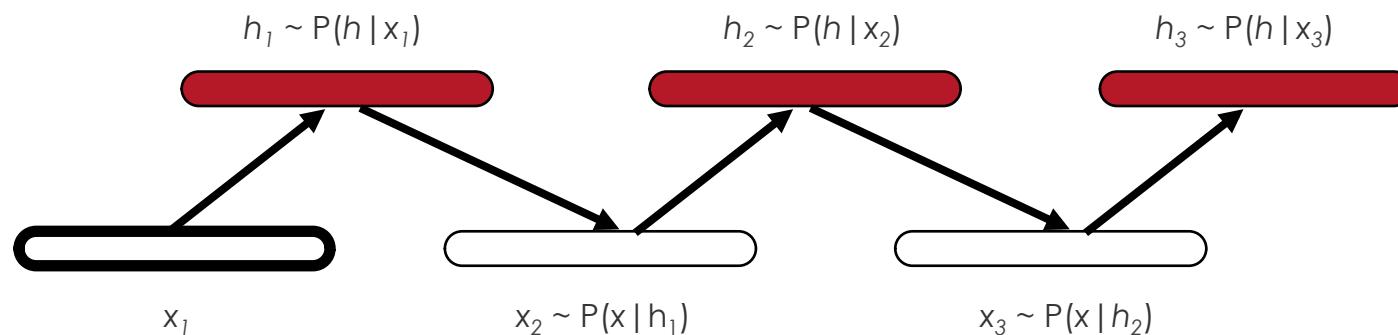
$$P(x, h) = \frac{1}{Z} e^{b^T h + c^T x + h^T W x} = \frac{1}{Z} e^{\sum_i b_i h_i + \sum_j c_j x_j + \sum_{i,j} h_i W_{ij} x_j}$$

- A popular building block for deep architectures

- **Bipartite** undirected graphical model



# Block Gibbs Sampling in RBMs



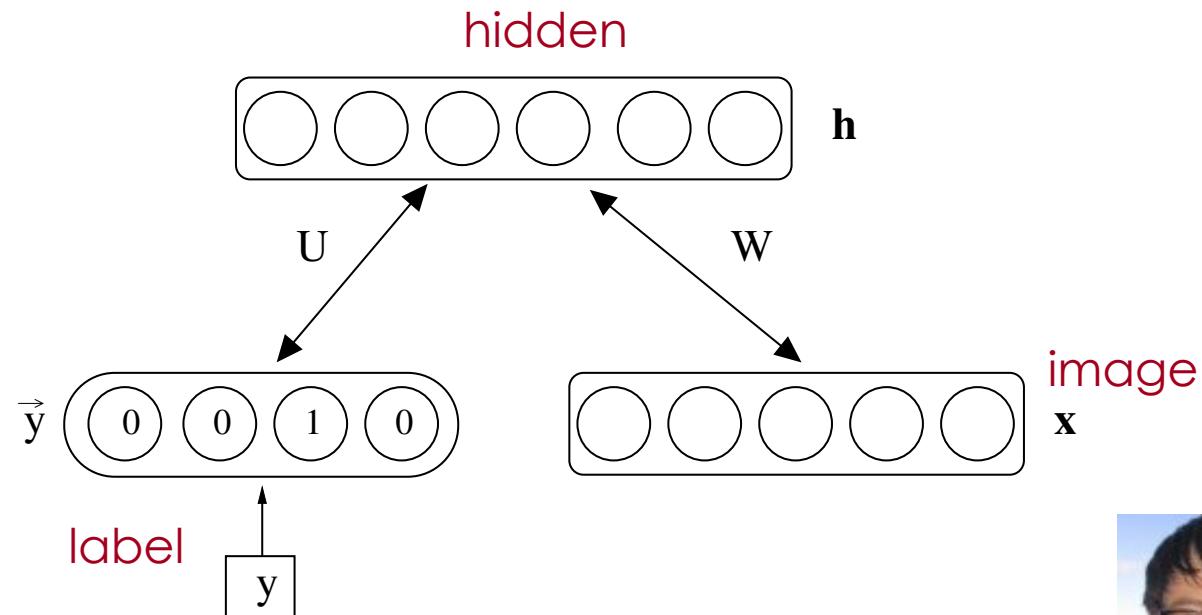
$P(h|x)$  and  $P(x|h)$  factorize

$$P(h|x) = \prod_i P(h_i|x)$$

- Easy inference
- Efficient **block Gibbs** sampling  $x \rightarrow h \rightarrow x \rightarrow h \dots$

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

# RBM with (image, label) visible units



(Larochelle & Bengio 2008)

# RBMs are Universal Approximators

(Le Roux & Bengio 2008)



- Adding one hidden unit (with proper choice of parameters) guarantees increasing likelihood
- With enough hidden units, can perfectly model any discrete distribution
- RBMs with variable # of hidden units = non-parametric

# Boltzmann Machine Gradient

$$P(x) = \frac{1}{Z} \sum_h e^{-\text{Energy}(x,h)} = \frac{1}{Z} e^{-\text{FreeEnergy}(x)}$$

- Gradient has two components:

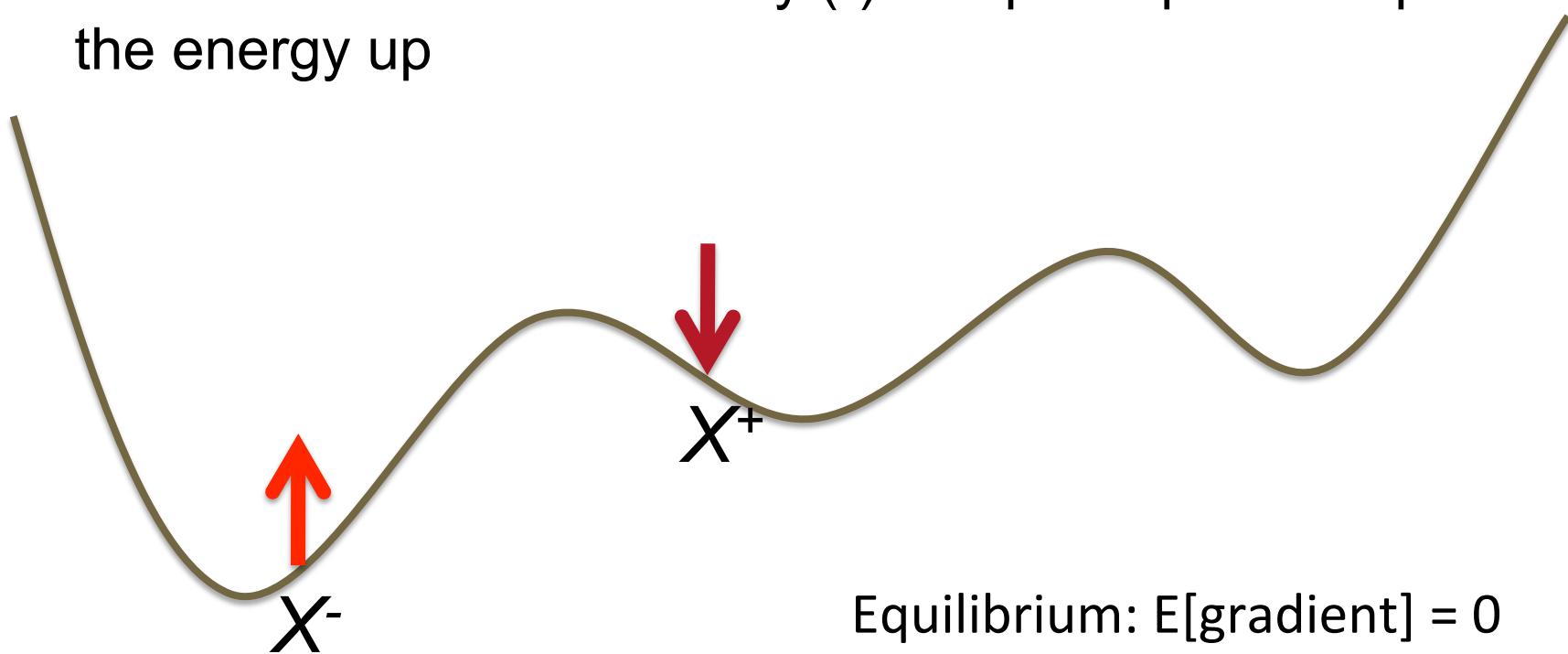
$$\begin{aligned}\frac{\partial \log P(x)}{\partial \theta} &= - \frac{\partial \text{FreeEnergy}(x)}{\partial \theta} + \sum_{\tilde{x}} P(\tilde{x}) \frac{\partial \text{FreeEnergy}(\tilde{x})}{\partial \theta} \\ &= - \sum_h P(h|x) \frac{\partial \text{Energy}(x,h)}{\partial \theta} + \sum_{\tilde{x}, \tilde{h}} P(\tilde{x}, \tilde{h}) \frac{\partial \text{Energy}(\tilde{x}, \tilde{h})}{\partial \theta}\end{aligned}$$

“positive phase”                            “negative phase”

- In RBMs, easy to sample or sum over  $h|x$
- Difficult part: sampling from  $P(x)$ , typically with a Markov chain

# Positive & Negative Samples

- Observed (+) examples push the energy down
- Generated / dream / fantasy (-) samples / particles push the energy up



# Training RBMs

Contrastive Divergence: start negative Gibbs chain at observed  $x$ , run  $k$  (CD- $k$ ) Gibbs steps

SML/Persistent CD: run negative Gibbs chain in background while (PCD) weights slowly change

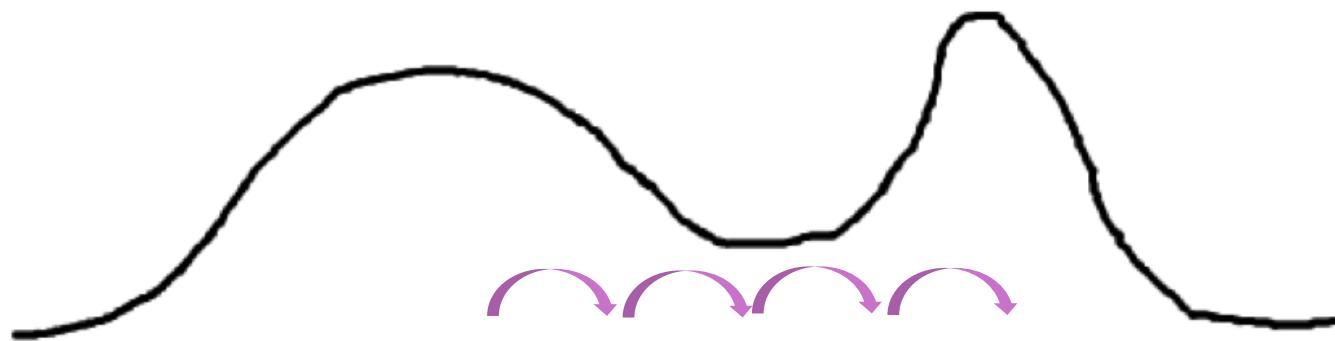
Fast PCD: two sets of weights, one with a large learning rate only used for negative phase, quickly exploring modes

Herding: Deterministic near-chaos dynamical system defines both learning and sampling

Tempered MCMC: use higher temperature to escape modes

# Obstacle: Vicious Circle Between Learning and MCMC Sampling

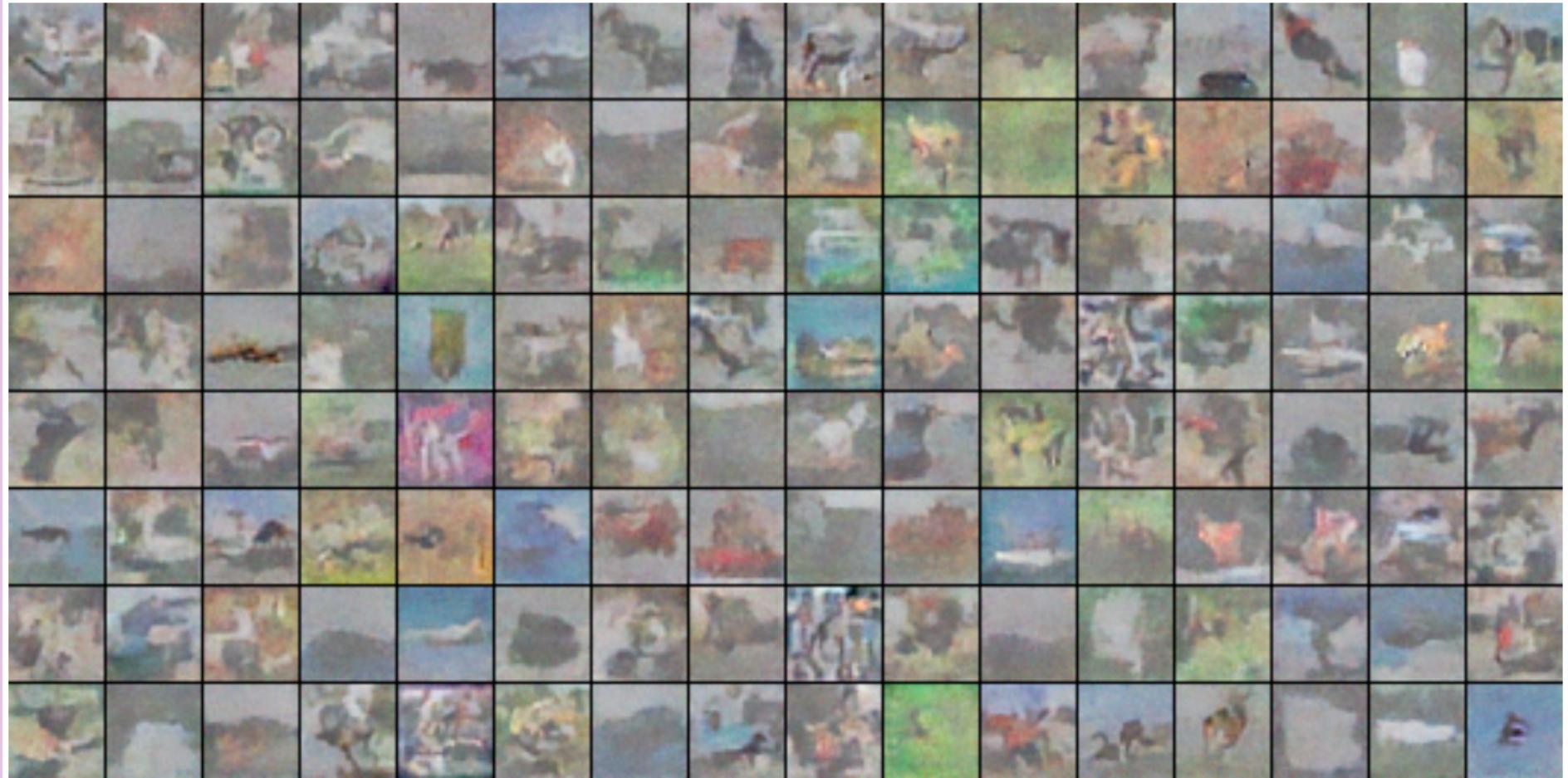
- Early during training, density smeared out, mode bumps overlap



- Later on, hard to cross empty voids between modes



# Convolutionally Trained Spike & Slab RBMs Samples



# ssRBM is not Cheating

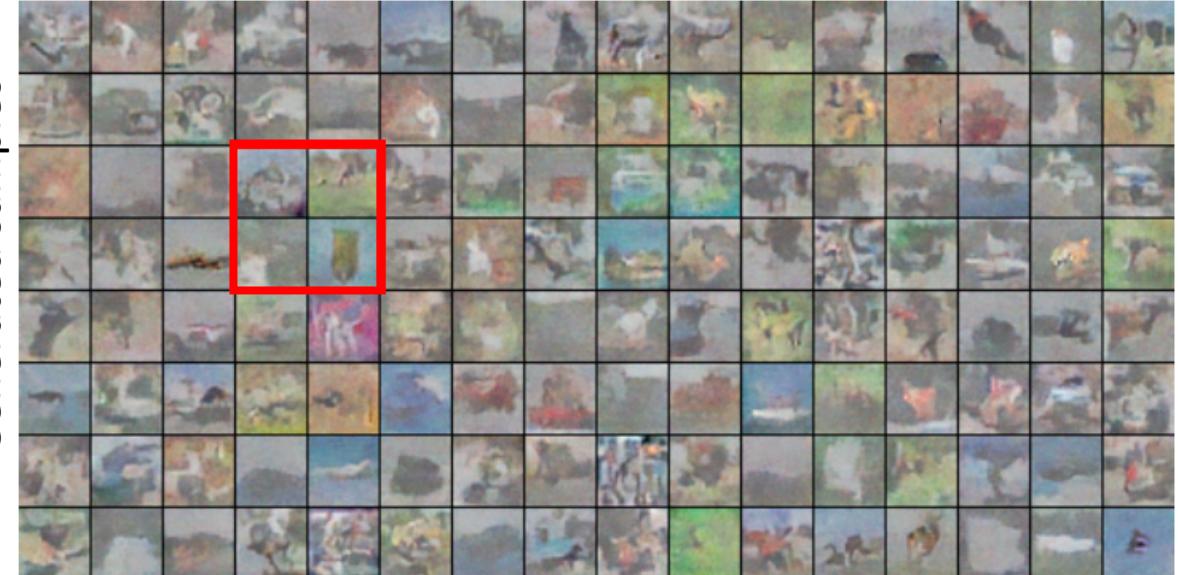
Samples from  $\mu$ -ssRBM:



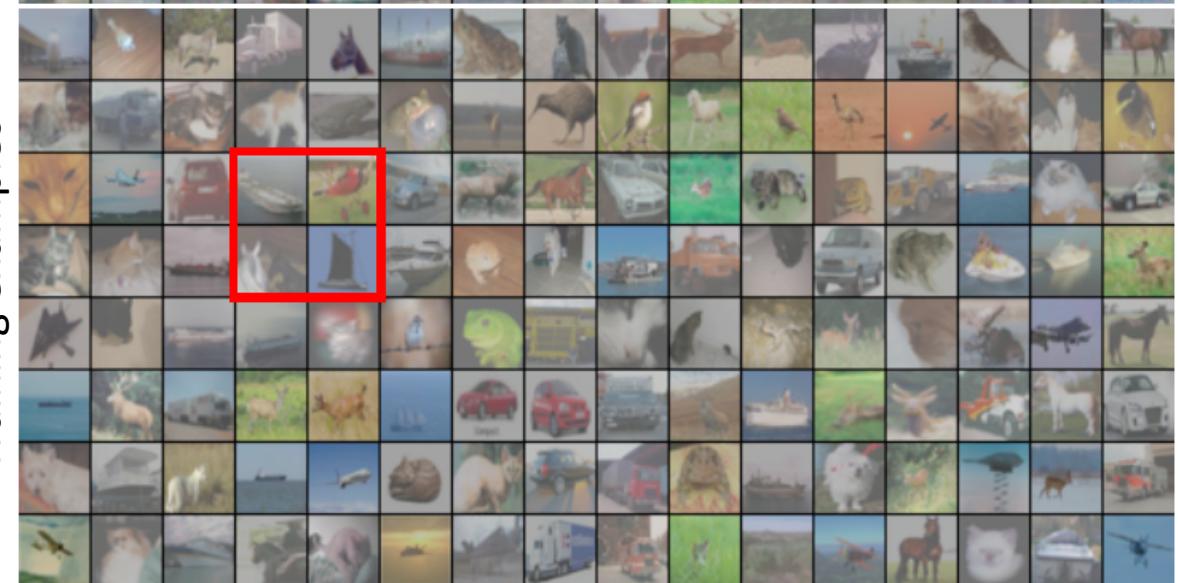
Nearest examples in CIFAR:  
(least square dist.)



Generated samples



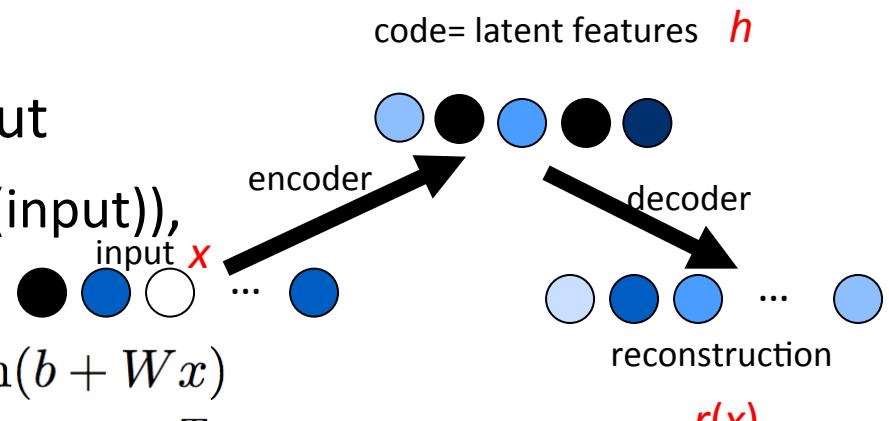
Training examples



# Auto-Encoders & Variants: Learning a computational graph

# Simple Auto-Encoders

- MLP whose target output = input
- Reconstruction=decoder(encoder(input)),  
e.g.



$$h = \tanh(b + Wx)$$

$$\text{reconstruction} = \tanh(c + W^T h)$$

$$\text{Loss } L(x, \text{reconstruction}) = \|\text{reconstruction} - x\|^2$$

- With bottleneck, code = new coordinate system
- Encoder and decoder can have 1 or more layers
- Training deep auto-encoders notoriously difficult

## (Auto-Encoder) Reconstruction Loss

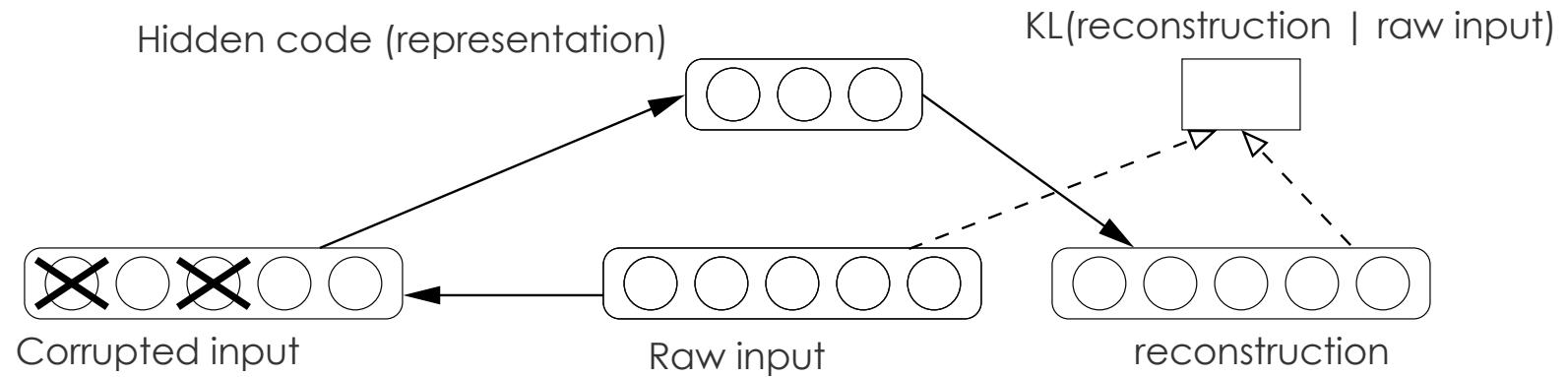
- Discrete inputs: cross-entropy for binary inputs
  - $-\sum_i x_i \log r_i(x) + (1-x_i) \log(1-r_i(x))$  (with  $0 < r_i(x) < 1$ )  
or log-likelihood reconstruction criterion, e.g., for a multinomial (one-hot) input
  - $-\sum_i x_i \log r_i(x)$  (where  $\sum_i r_i(x)=1$ , summing over subset of inputs associated with this multinomial variable)
- In general: consider what are appropriate loss functions to predict each of the input variables,  
typically, **reconstruction neg. log-likelihood  $-\log P(x|h(x))$**

# Denoising Auto-Encoder

(Vincent et al 2008)



- Corrupt the input during training only
- Train to reconstruct the uncorrupted input



- Encoder & decoder: any parametrization
- As good or better than RBMs for unsupervised pre-training

# Denoising Auto-Encoder

- Learns a vector field pointing towards higher probability direction (Alain & Bengio 2013)

$$r(x) - x \propto d\log(p(x)) / dx$$

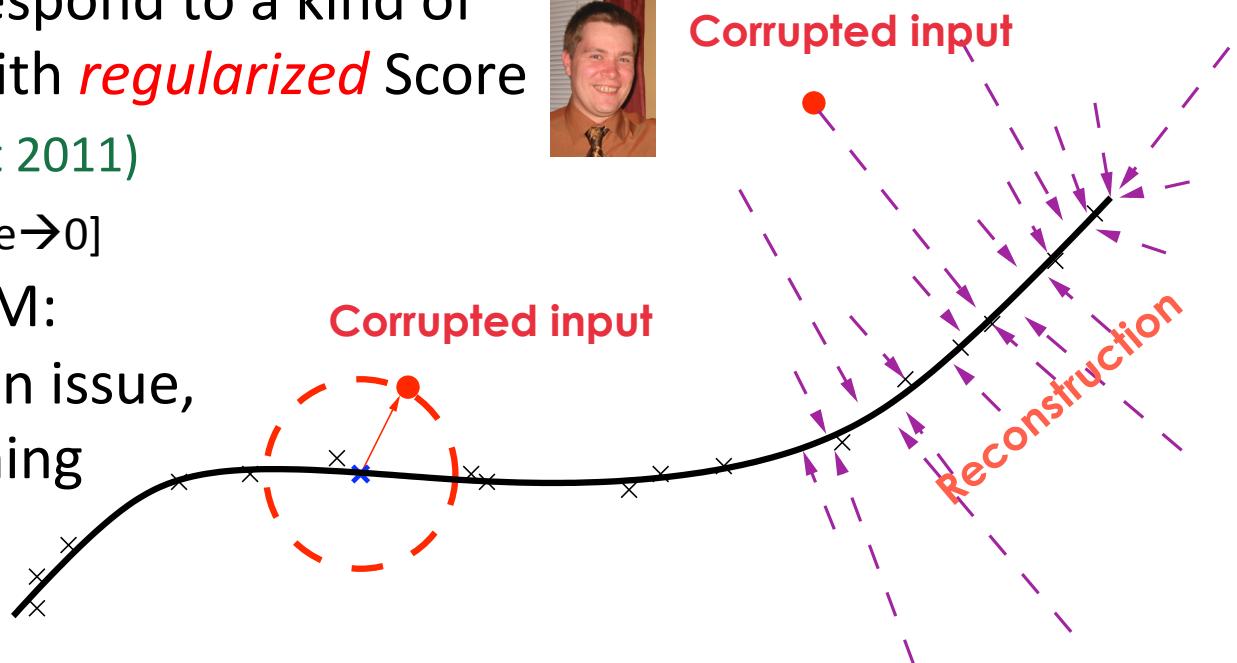


prior: examples concentrate near a lower dimensional “manifold”

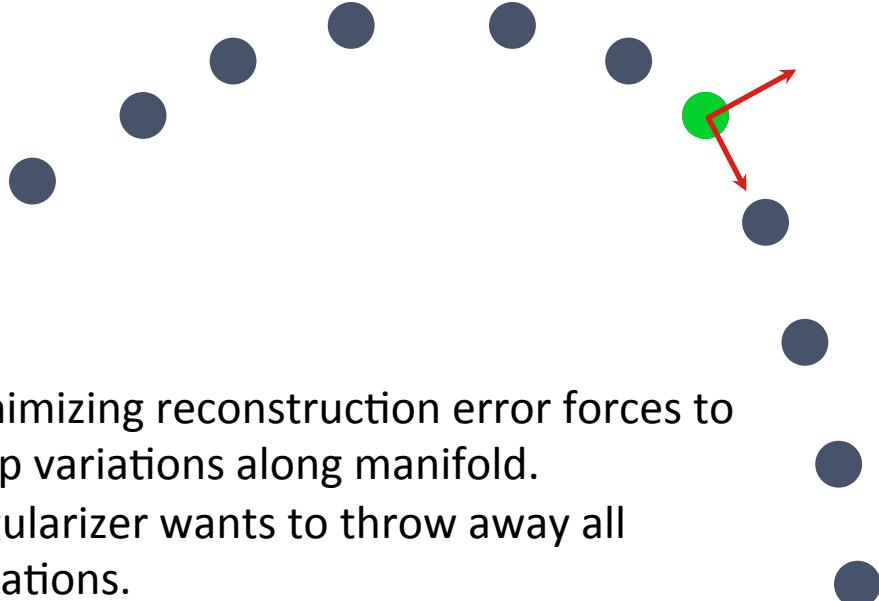
- Some DAEs correspond to a kind of Gaussian RBM with *regularized* Score Matching (Vincent 2011)

[equivalent when noise  $\rightarrow 0$ ]

- Compared to RBM:  
No partition function issue,  
+ can measure training criterion

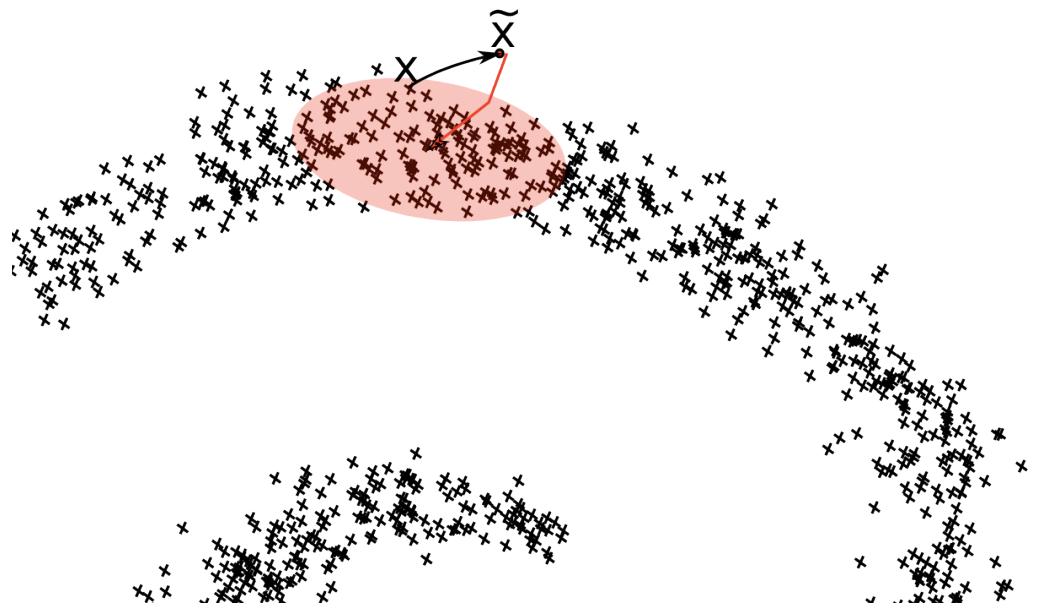
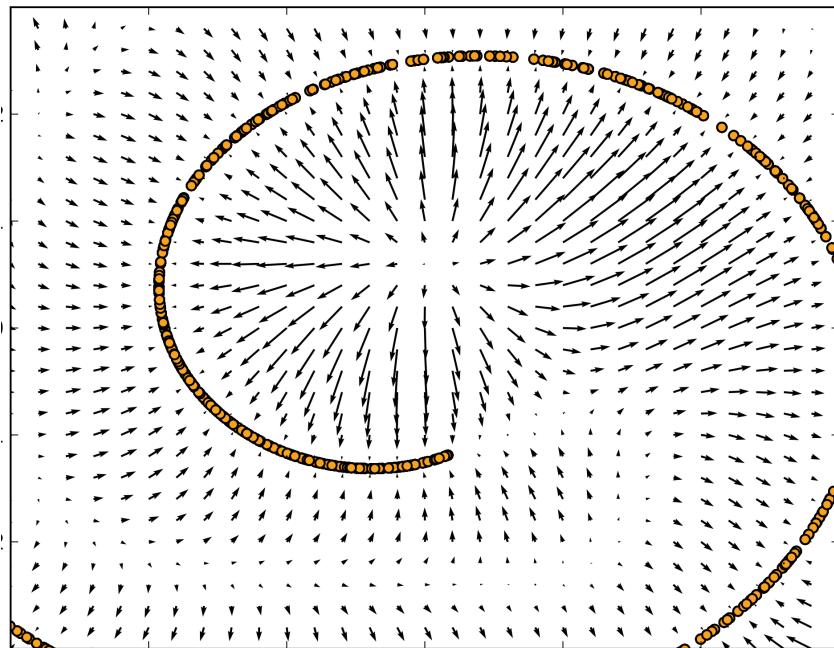


# Auto-Encoders Learn Salient Variations, Like a non-Linear PCA



# Regularized Auto-Encoders Learn a Vector Field or a Markov Chain Transition Distribution

- (Bengio, Vincent & Courville, TPAMI 2013) review paper
- (Alain & Bengio ICLR 2013; Bengio et al, arxiv 2013)



# Contractive Auto-Encoders



(Rifai, Vincent, Muller, Glorot, Bengio ICML 2011; Rifai, Mesnil, Vincent, Bengio, Dauphin, Glorot ECML 2011; Rifai, Dauphin, Vincent, Bengio, Muller NIPS 2011)

$$\text{reconstruction}(x) = g(h(x)) = \text{decoder}(\text{encoder}(x))$$

Training criterion:

$$\mathcal{J}_{CAE}(\theta) = \sum_{x \in D_n} \lambda \sum_{ij} \left( \frac{\partial h_j(x)}{\partial x_i} \right)^2 + L(x, \text{reconstruction}(x))$$

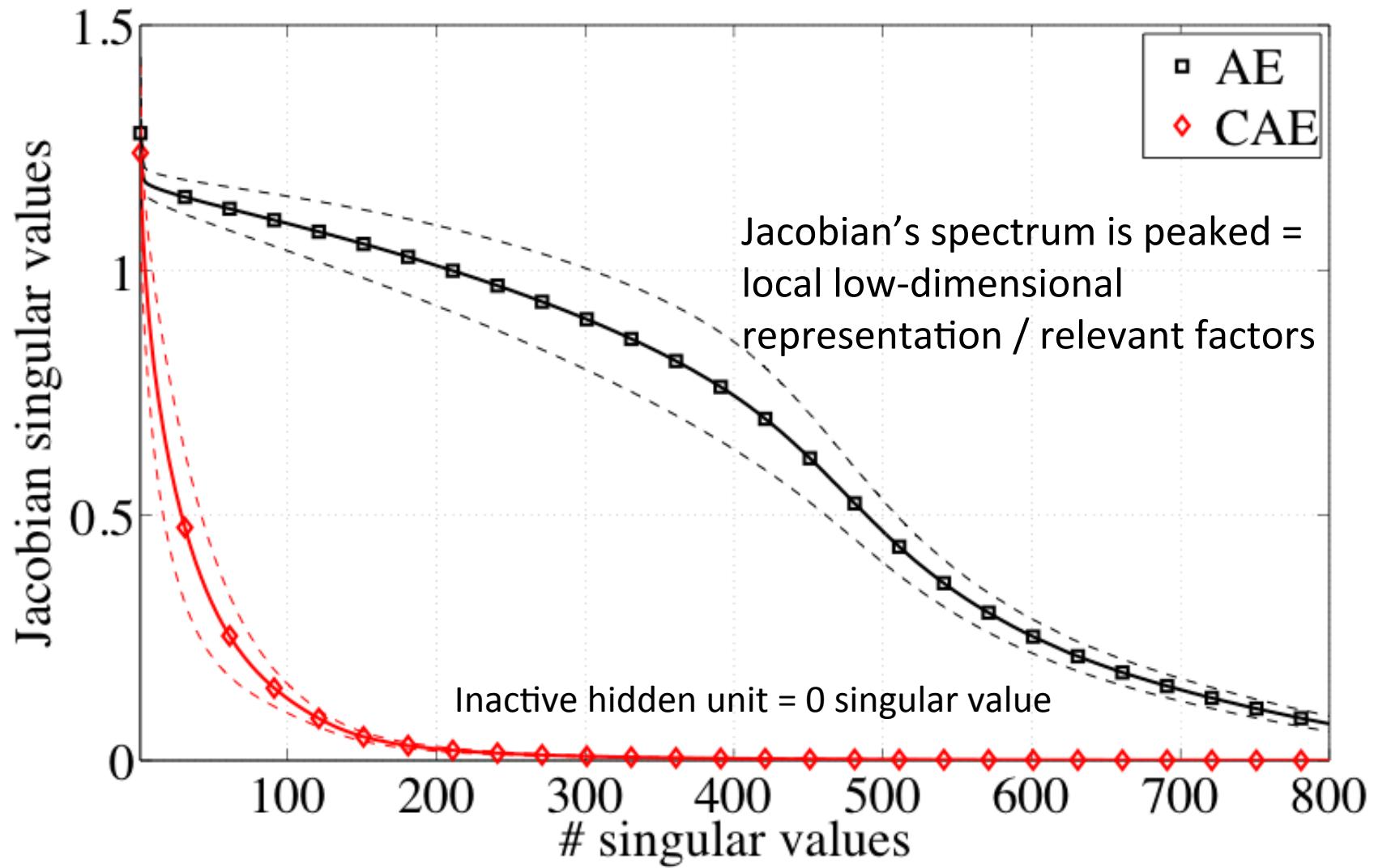
wants contraction in all directions

cannot afford contraction in manifold directions

If  $h_j = \text{sigmoid}(b_j + W_j x)$

$$(\frac{\partial h_j(x)}{\partial x_i})^2 = h_j^2 (1-h_j)^2 W_{ji}^2$$

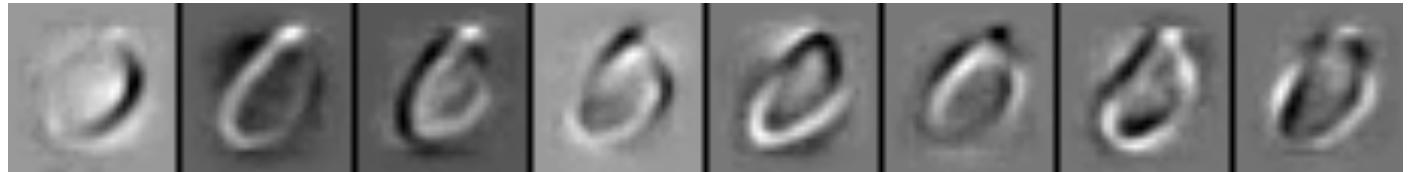
## CIFAR-10



Input Point



Tangents



$$\text{Input Point} + 0.5 \times \text{Tangent} = \text{Output Image}$$

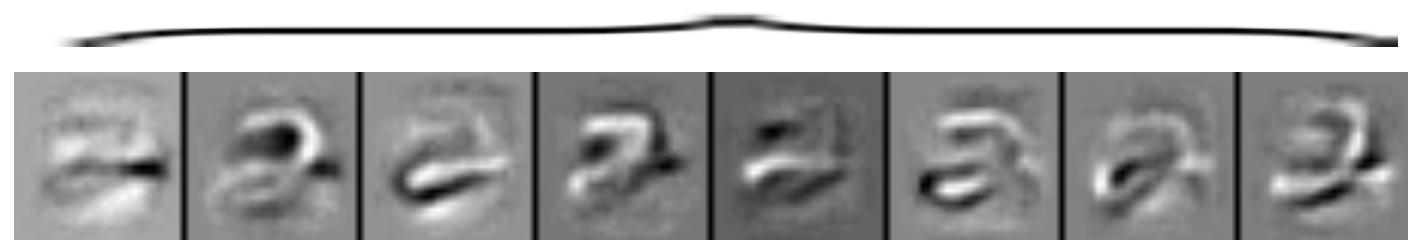
A diagram illustrating a linear combination of the input point and its tangents to produce a new output image. The input point is a handwritten digit '0'. The tangent is represented by a small image of the digit '0' with a prominent vertical stroke on the right. The result is a slightly distorted version of the original digit '0'.

MNIST

Input Point

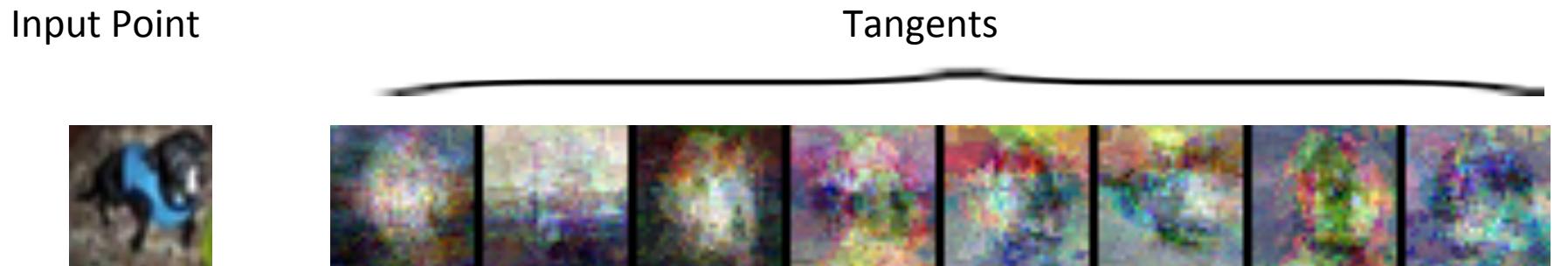


Tangents



MNIST Tangents

# Distributed vs Local (CIFAR-10 unsupervised)



Local PCA (no sharing across regions)



Contractive Auto-Encoder

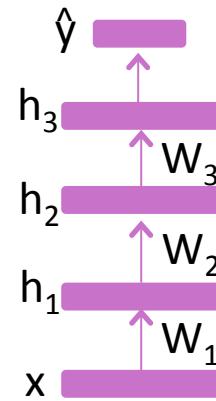
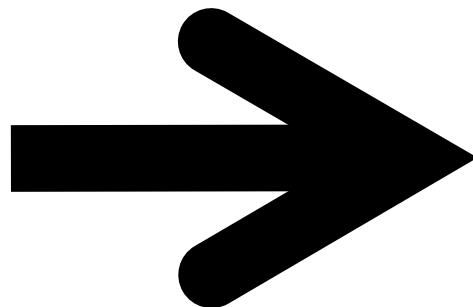
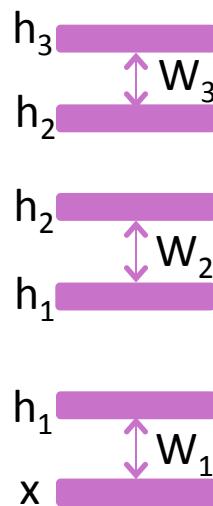
# Deep Variants

## 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, auto-encoder, denoising auto-encoder, etc can help a lot
- Helps most the layers further away from the target
- Not just an effect of the unsupervised prior
- Jointly training all the levels of a deep architecture is difficult because of the increased non-linearity / non-smoothness
- Initializing using a **level-local learning algorithm** is a useful trick
- Providing intermediate-level targets can help tremendously  
*(Gulcehre & Bengio ICLR 2013)*

# Stack of RBMs / AEs → Deep MLP

- Encoder or  $P(h|v)$  becomes MLP layer

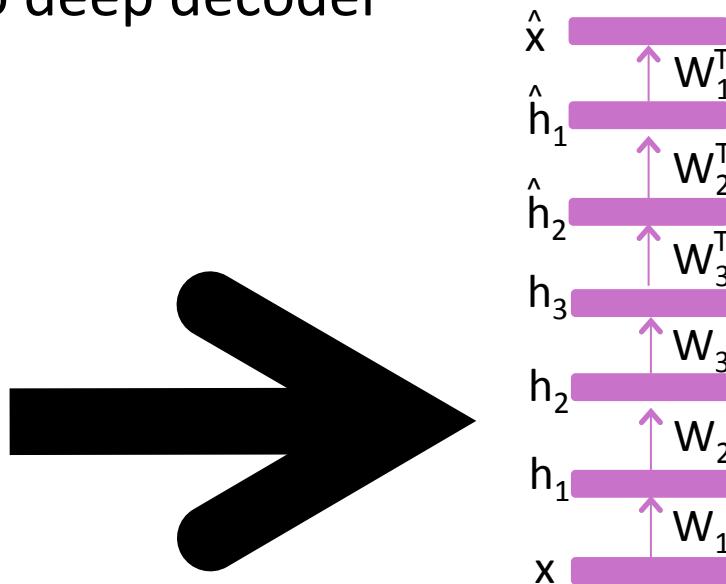
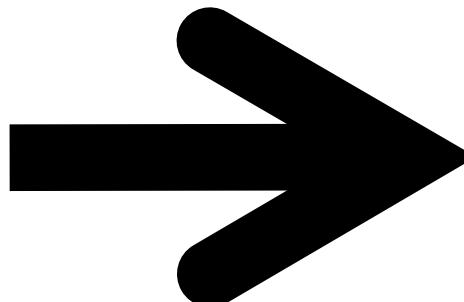
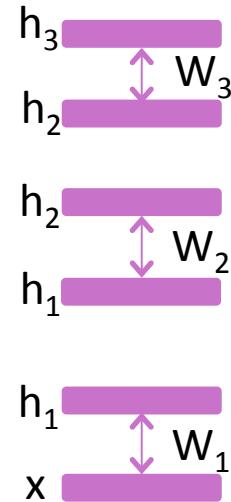


# Stack of RBMs / AEs → Deep Auto-Encoder

(Hinton & Salakhutdinov 2006)



- Stack encoders /  $P(h|x)$  into deep encoder
- Stack decoders /  $P(x|h)$  into deep decoder

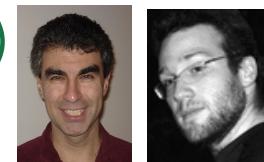


# Stack of RBMs / AEs → Deep Recurrent Auto-Encoder

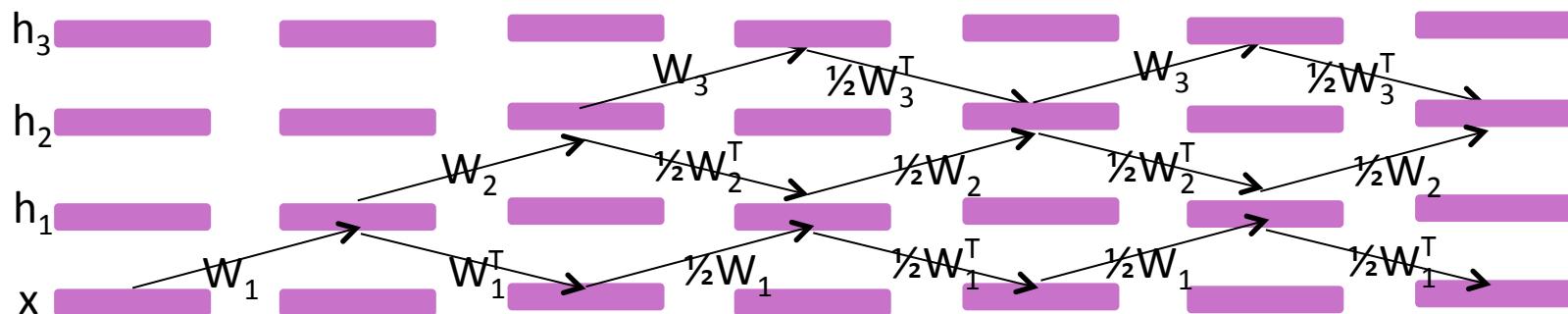
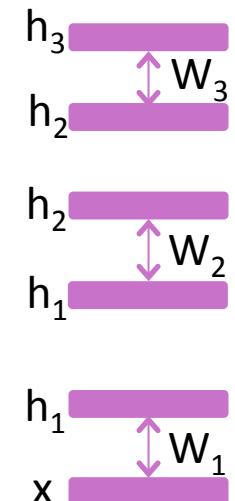
(Savard 2011)



(Bengio & Laufer, arxiv 2013)



- Each hidden layer receives input from below and above
- Deterministic (mean-field) recurrent computation (Savard 2011)
- Stochastic (injecting noise) recurrent computation: Deep Generative Stochastic Networks (GSNs)  
(Bengio & Laufer arxiv 2013)

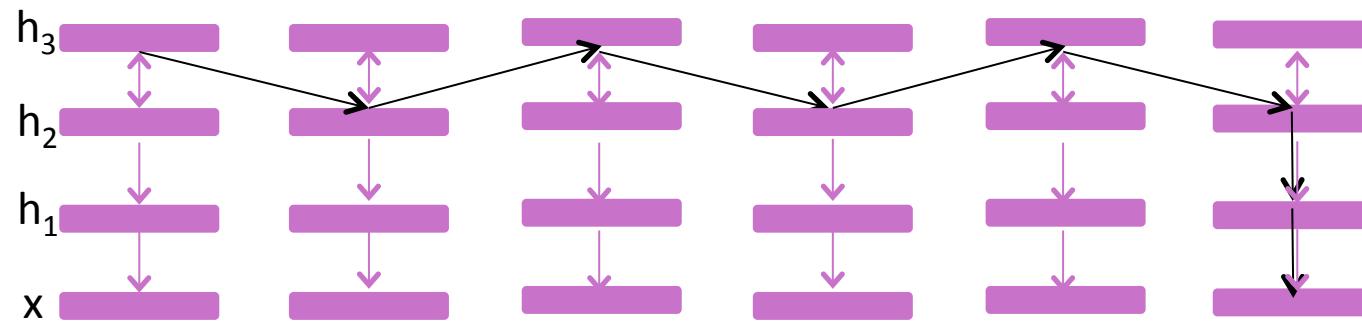


# Stack of RBMs → Deep Belief Net



(Hinton et al 2006)

- Stack lower levels RBMs'  $P(x|h)$  along with top-level RBM
- $P(x, h_1, h_2, h_3) = P(h_2, h_3) P(h_1|h_2) P(x | h_1)$
- Sample: Gibbs on top RBM, propagate down

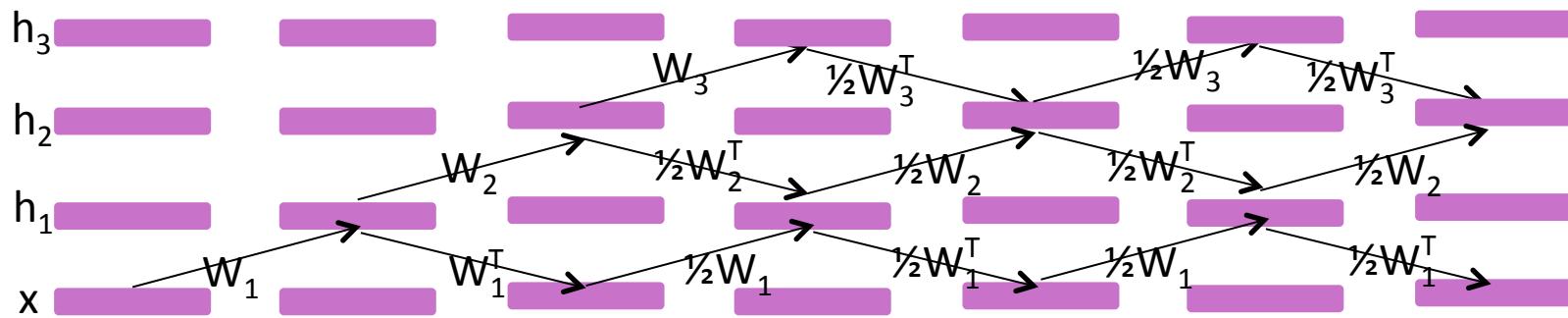


# Stack of RBMs → Deep Boltzmann Machine

(Salakhutdinov & Hinton AISTATS 2009)



- Halve the RBM weights because each layer now has inputs from below and from above
- Positive phase: (mean-field) variational inference = recurrent AE
- Negative phase: Gibbs sampling (stochastic units)
- train by SML/PCD

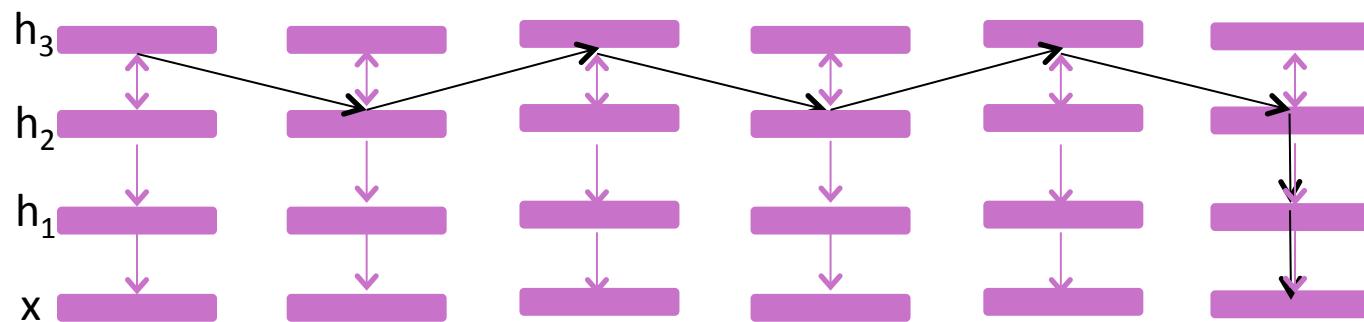


# Stack of Auto-Encoders → Deep Generative Auto-Encoder

(Rifai et al ICML 2012)



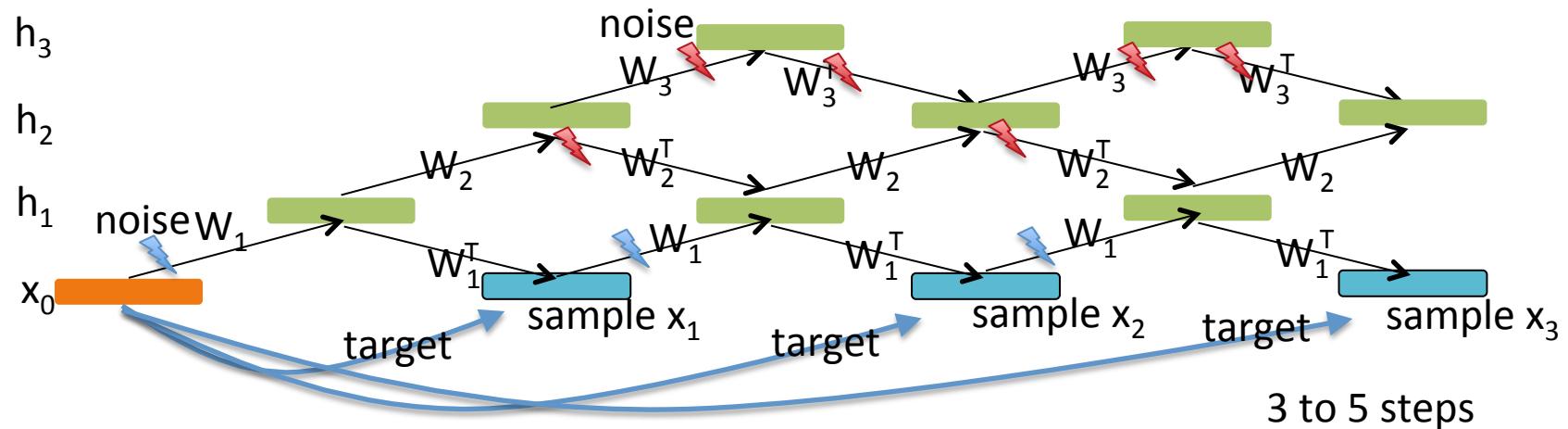
- MCMC on top-level auto-encoder
  - $h_{t+1} = \text{encode}(\text{decode}(h_t)) + \sigma \text{ noise}$   
where noise is  $\text{Normal}(0, d/dh \text{ encode}(\text{decode}(h_t)))$
- Then deterministically propagate down with decoders



# Generative Stochastic Networks (GSN)

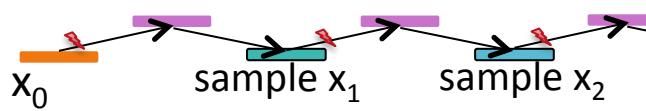
(Bengio, Yao, Alain & Vincent, arxiv 2013; Bengio & Laufer, arxiv 2013)

- Recurrent parametrized stochastic computational graph that defines a transition operator for a Markov chain whose asymptotic distribution is implicitly estimated by the model
- Noise injected in input and hidden layers
- Trained to max. reconstruction prob. of example at each step
- **Example** structure inspired from the DBM Gibbs chain:

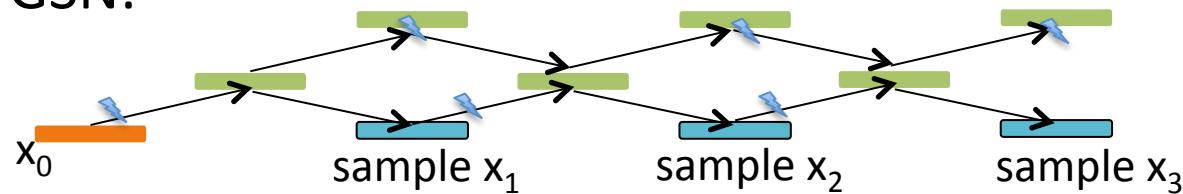


# Experiments: Shallow vs Deep

- Shallow (DAE), no recurrent path at higher levels, state= $X$  only

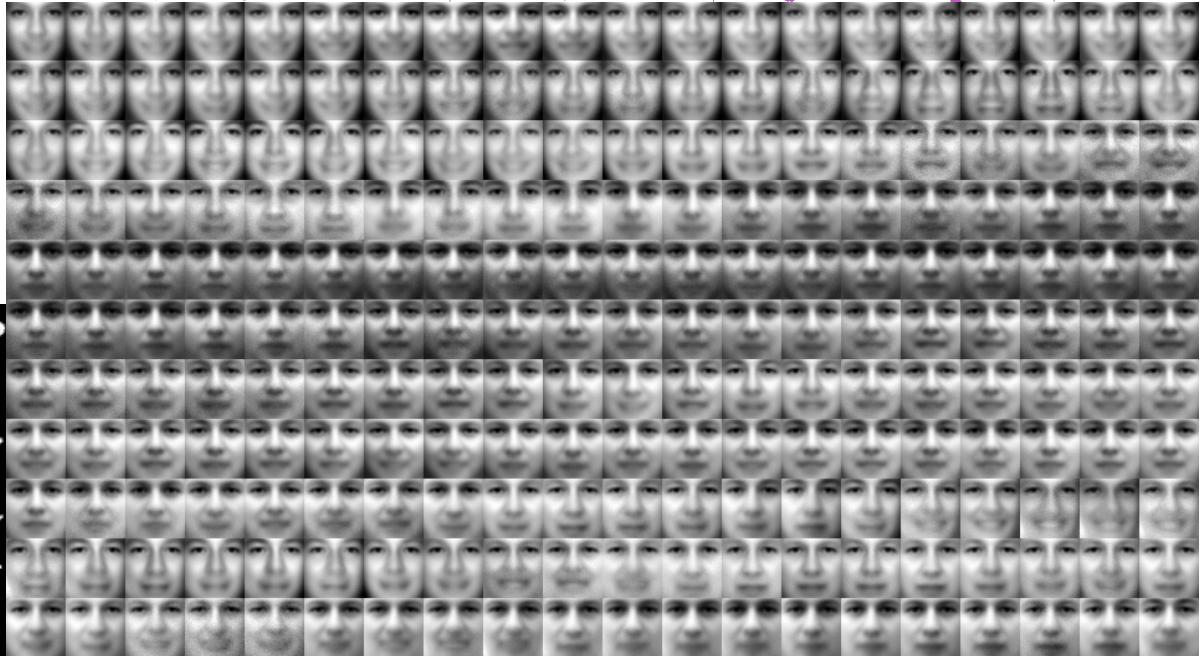


- Deep GSN:



# GSN Experiments: Consecutive Samples

Filling-in the LHS

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	
1 1 1 1 1 1 1 1 1 1 1 1 1 1	
2 2 2 2 2 2 2 2 2 2 2 2 2 2	
3 3 3 3 3 3 3 3 3 3 3 3 3 3	
4 4 4 4 4 4 4 4 4 4 4 4 4 4	
5 5 5 5 5 5 5 5 5 5 5 5 5 5	
6 6 6 6 6 6 6 6 6 6 6 6 6 6	
7 7 7 7 7 7 7 7 7 7 7 7 7 7	
8 8 8 8 8 8 8 8 8 8 8 8 8 8	
9 9 9 9 9 9 9 9 9 9 9 9 9 9	
4 4 4 4 4 4 4 4 4 4 4 4 4 4	
4 4 4 4 4 4 4 4 4 4 4 4 4 4	
9 9 9 9 9 9 9 9 9 9 9 9 9 9	
8 8 8 8 8 8 8 8 8 8 8 8 8 8	
8 8 8 8 8 8 8 8 8 8 8 8 8 8	
5 5 5 5 5 5 5 5 5 5 5 5 5 5	
5 5 5 5 5 5 5 5 5 5 5 5 5 5	
6 6 6 6 6 6 6 6 6 6 6 6 6 6	
6 6 6 6 6 6 6 6 6 6 6 6 6 6	
0 0 0 0 0 0 0 0 0 0 0 0 0 0	
0 0 0 0 0 0 0 0 0 0 0 0 0 0	

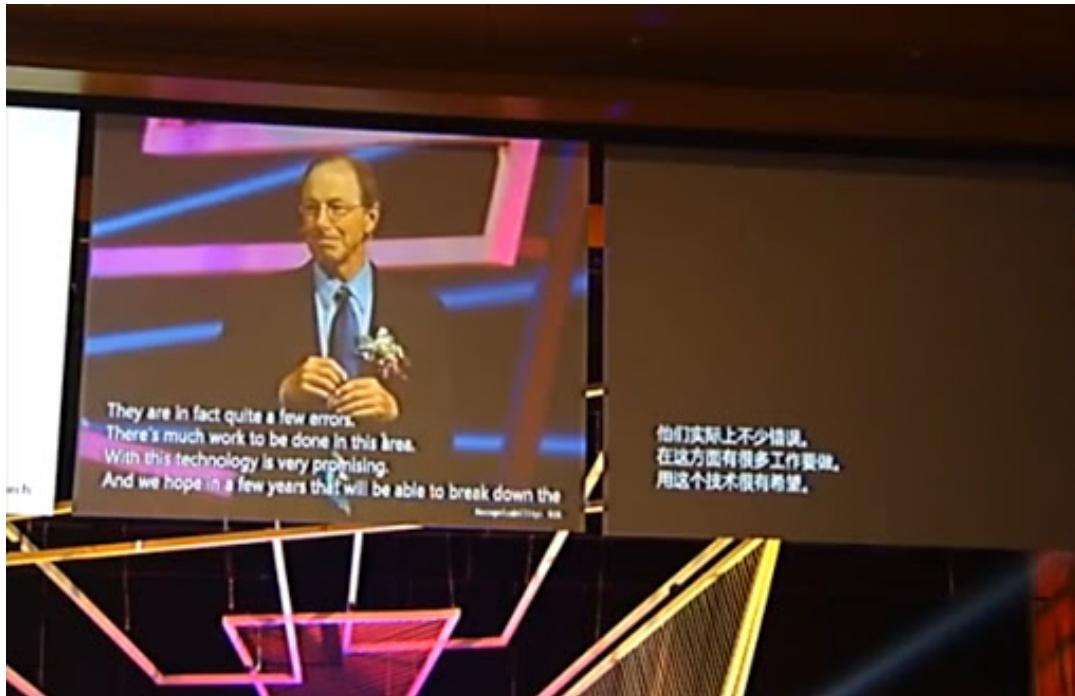
# Applications

# AI Tasks

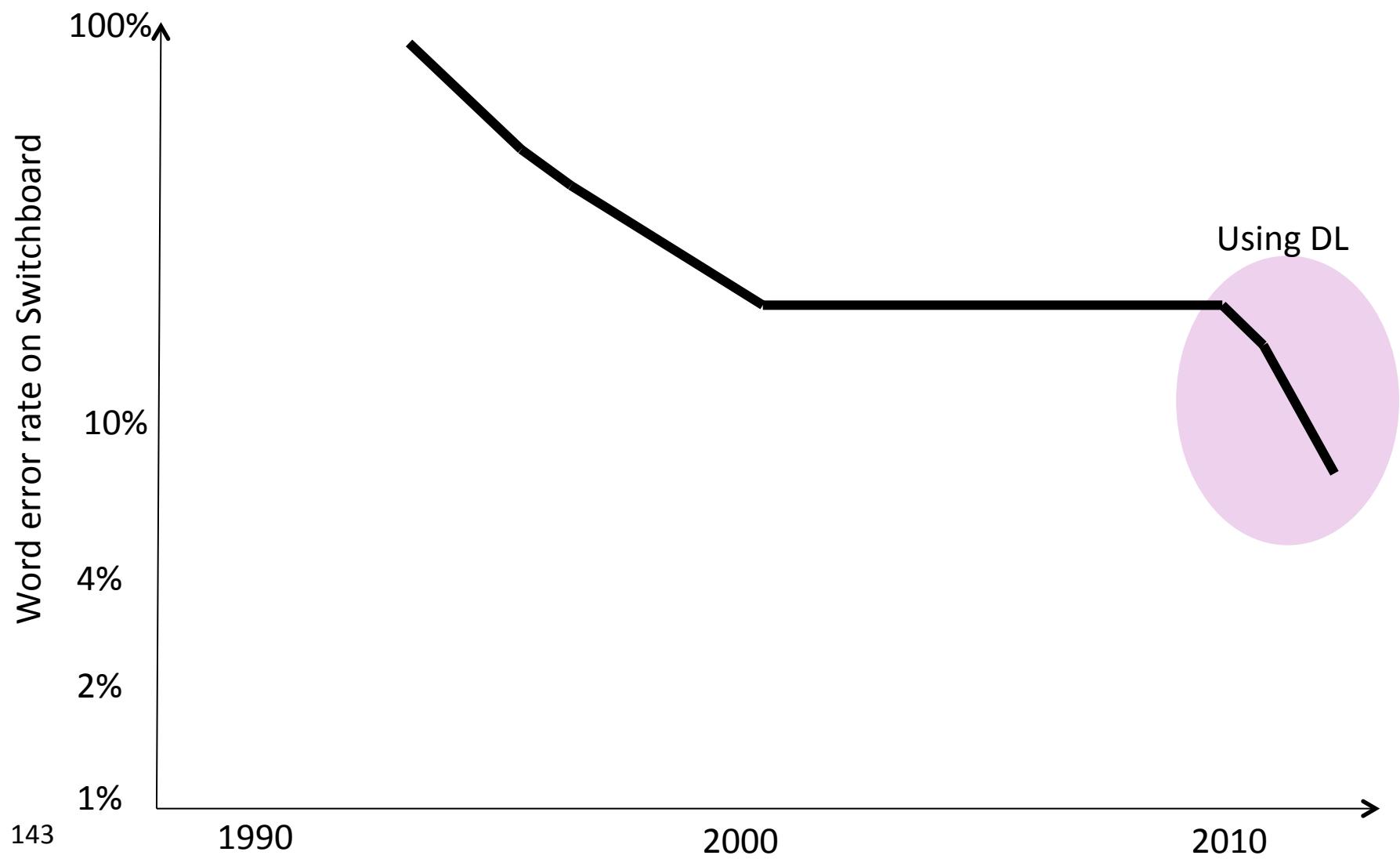
- Perception
  - Vision
  - Speech
  - Multiple modalities
- Natural language understanding
- Reinforcement learning & control
- COMPLEX HIGHLY-STRUCTURED DISTRIBUTION
- LOTS OF DATA (maybe mostly unlabeled)

## 2012: Industrial-scale success in speech recognition

- Google uses DL in their android speech recognizer (both server-side and on some phones with enough memory)
- Microsoft uses DL in their speech recognizer
- Error reductions on the order of 30%, a major progress



# The dramatic impact of Deep Learning on Speech Recognition (according to Microsoft)



# Deep Networks for Speech Recognition: results from Google, IBM, Microsoft

task	Hours of training data	Deep net+HMM	GMM+HMM same data	GMM+HMM more data
Switchboard	309	16.1	23.6	17.1 (2k hours)
English Broadcast news	50	17.5	18.8	
Bing voice search	24	30.4	36.2	
Google voice input	5870	12.3		16.0 (lots more)
Youtube	1400	47.6	52.3	

# Some Applications of DL

- **Language Modeling** (Speech Recognition, Machine Translation)
- **Acoustic Modeling** (**speech recognition**, music modeling)
- **NLP syntactic/semantic tagging** (Part-Of-Speech, chunking, Named Entity Recognition, Semantic Role Labeling, Parsing)
- **NLP applications**: sentiment analysis, paraphrasing, question-answering, Word-Sense Disambiguation
- **Object recognition in images**: photo search and image search: handwriting recognition, document analysis, handwriting synthesis, **superhuman traffic sign classification**, street view house numbers, **emotion detection from facial images**, roads from satellites.
- **Personalization/recommendation/fraud/ads**
- **Molecular properties**: QSAR, quantum calculations

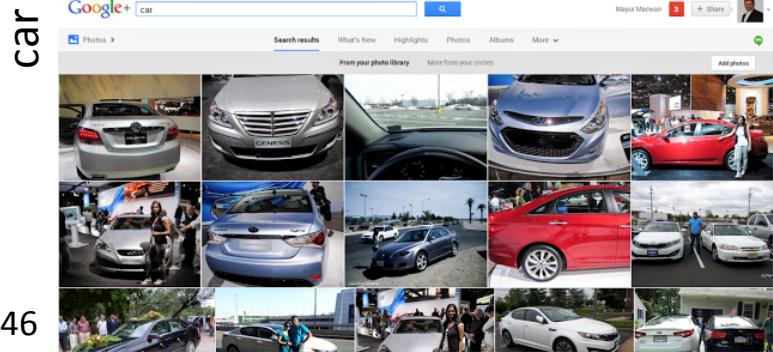


# Industrial-scale success in object recognition

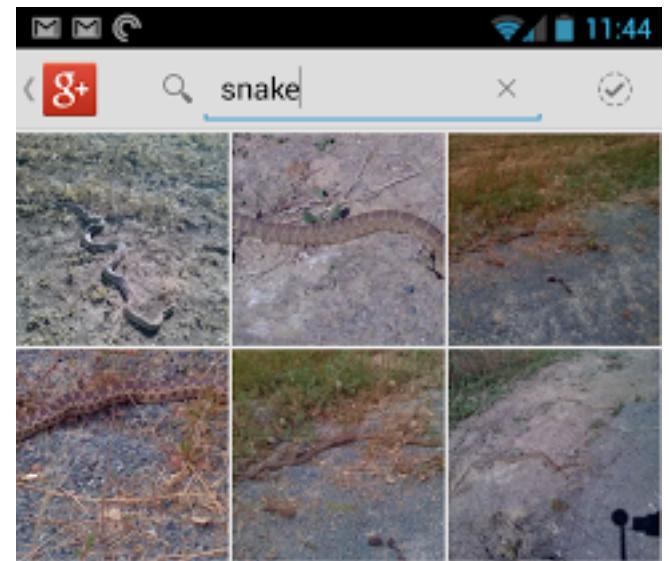
- Krizhevsky, Sutskever & Hinton NIPS 2012

	1 <sup>st</sup> choice	Top-5
2 <sup>nd</sup> best		27% err
Previous SOTA	45% err	26% err
Krizhevsky et al	37% err	15% err

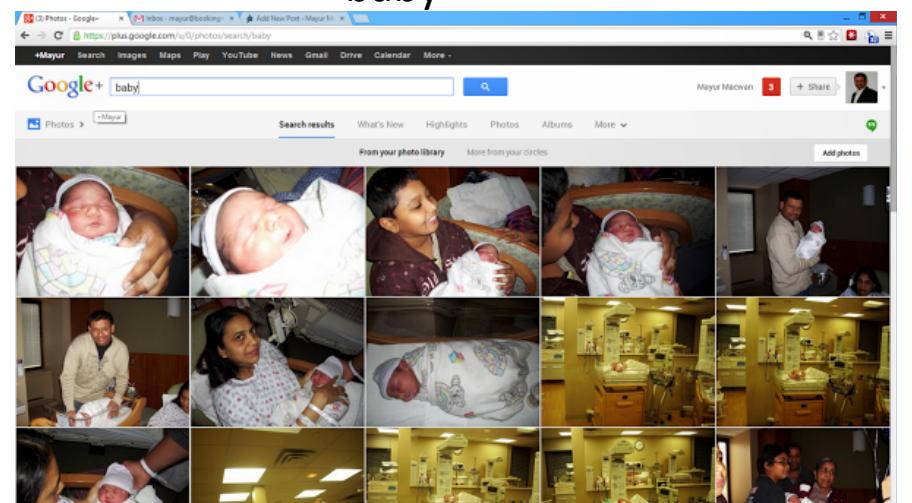
- Google incorporates DL in Google+ photo search, “A step across the semantic gap” (Google Research blog, June 12, 2013)**
- Baidu now offers similar services



146



baby



# Montreal Deep Nets Win Emotion Recognition in the Wild Challenge

Predict emotional expression from video (using images + audio)

National University

Dec. 9, 2013

## Results!

Team Name	Classification accuracy
Audio baseline	22.4 %
Video baseline	22.7 %
Fusion	27.5 %
Nottingham	24.7 %
Oulu	21.5 %
KIT	29.8 %
UCSD	37.1 %
ICT@CAS	35.9 %
York	27.6 %
LNMIIT	20.5 %
Montreal	41.0 %
Ulm	27.2 %

# More Successful Applications

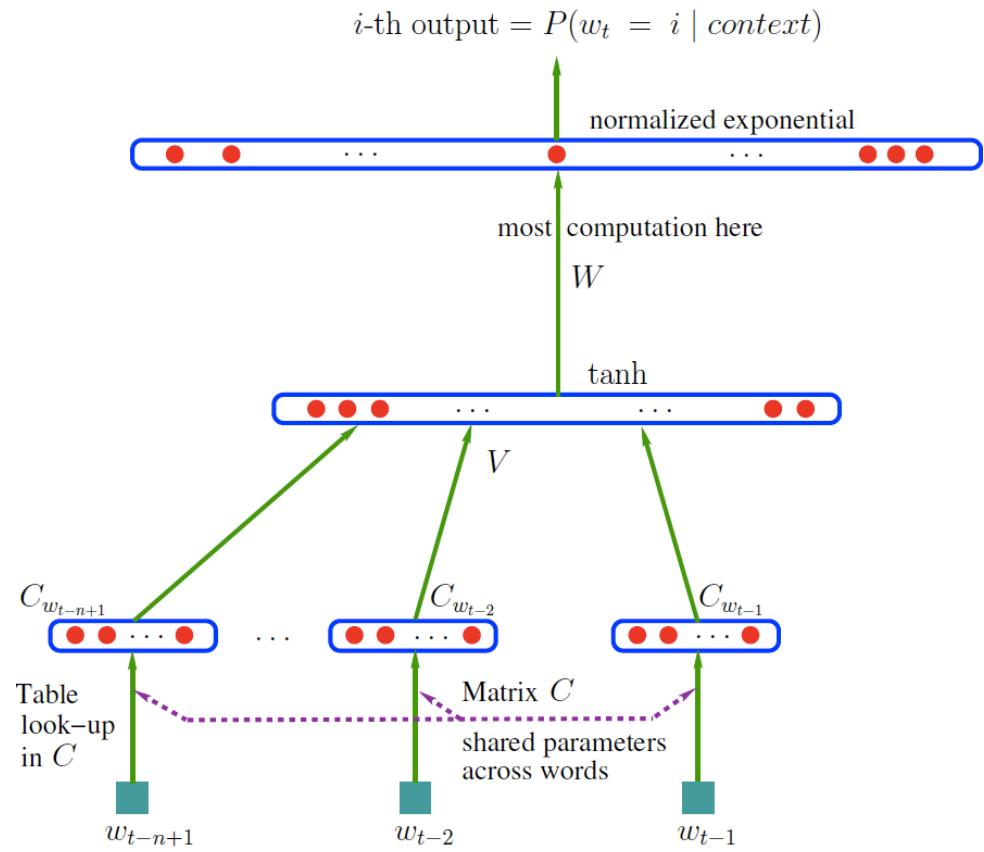
- Microsoft uses DL for speech rec. service (audio video indexing), based on Hinton/Toronto's DBNs ([Mohamed et al 2012](#))
- Google uses DL in its Google Goggles service, using Ng/Stanford DL systems, and in its Google+ photo search service, using deep convolutional nets
- NYT talks about these: [http://www.nytimes.com/2012/06/26/technology/in-a-big-network-of-computers-evidence-of-machine-learning.html?\\_r=1](http://www.nytimes.com/2012/06/26/technology/in-a-big-network-of-computers-evidence-of-machine-learning.html?_r=1)
- Substantially beating SOTA in language modeling (perplexity from 140 to 102 on Broadcast News) for speech recognition (WSJ WER from 16.9% to 14.4%) ([Mikolov et al 2011](#)) and translation (+1.8 BLEU) ([Schwenk 2012](#))
- SENNA: Unsup. pre-training + multi-task DL reaches SOTA on POS, NER, SRL, chunking, parsing, with >10x better speed & memory ([Collobert et al 2011](#))
- Recursive nets surpass SOTA in paraphrasing ([Socher et al 2011](#))
- Denoising AEs substantially beat SOTA in sentiment analysis ([Glorot et al 2011](#))
- Contractive AEs SOTA in knowledge-free MNIST (.8% err) ([Rifai et al NIPS 2011](#))
- Le Cun/NYU's stacked PSDs most accurate & fastest in pedestrian detection and DL in top 2 winning entries of German road sign recognition competition

# Already Many NLP Applications of DL

- Language Modeling (Speech Recognition, Machine Translation)
- Acoustic Modeling
- Part-Of-Speech Tagging
- Chunking
- Named Entity Recognition
- Semantic Role Labeling
- Parsing
- Sentiment Analysis
- Paraphrasing
- Question-Answering
- Word-Sense Disambiguation

# Neural Language Model

- *Bengio et al NIPS'2000 and JMLR 2003 "A Neural Probabilistic Language Model"*
  - Each word represented by a distributed continuous-valued code vector = embedding
  - Generalizes to sequences of words that are **semantically similar** to training sequences

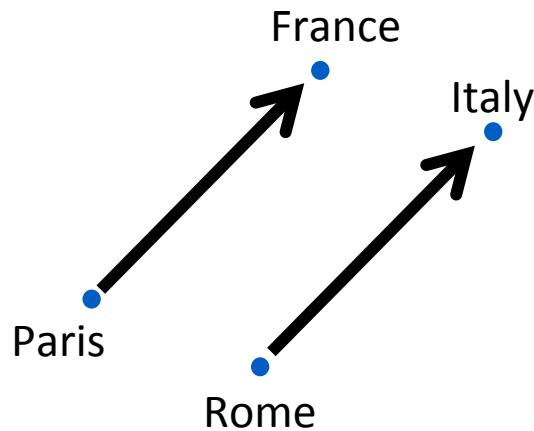


# Neural word embeddings – visualization



## Analogical Representations for Free (Mikolov et al, ICLR 2013)

- Semantic relations appear as linear relationships in the space of learned representations
- King – Queen  $\approx$  Man – Woman
- Paris – France + Italy  $\approx$  Rome



# Practical Considerations

# Deep Learning Tricks of the Trade

- Y. Bengio (2013), “Practical Recommendations for Gradient-Based Training of Deep Architectures”
  - Unsupervised pre-training
  - Stochastic gradient descent and setting learning rates
  - Main hyper-parameters
    - Learning rate schedule
    - Early stopping
    - Minibatches
    - Parameter initialization
    - Number of hidden units
    - L1 and L2 weight decay
    - Sparsity regularization
  - Debugging
  - How to efficiently search for hyper-parameter configurations



# Stochastic Gradient Descent (SGD)

- Gradient descent uses total gradient over all examples per update, SGD updates after only 1 or few examples:

$$\theta^{(t)} \leftarrow \theta^{(t-1)} - \epsilon_t \frac{\partial L(z_t, \theta)}{\partial \theta}$$

- $L$  = loss function,  $z_t$  = current example,  $\theta$  = parameter vector, and  $\epsilon_t$  = learning rate.
- Ordinary gradient descent is a batch method, very slow, **should never be used**. 2<sup>nd</sup> order batch method are being explored as an alternative but SGD with selected learning schedule remains the method to beat.

# Learning Rates

- Simplest recipe: keep it fixed and use the same for all parameters.
- Collobert scales them by the inverse of square root of the fan-in of each neuron
- Better results can generally be obtained by allowing learning rates to decrease, typically in  $O(1/t)$  because of theoretical convergence guarantees, e.g.,

$$\epsilon_t = \frac{\epsilon_0 \tau}{\max(t, \tau)}$$

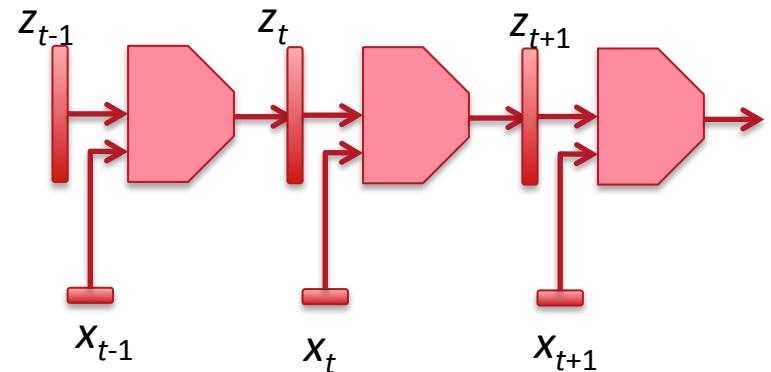
with hyper-parameters  $\epsilon_0$  and  $\tau$ .

- New papers on adaptive learning rates procedures (Schaul 2012, 2013), Adagrad (Duchi et al 2011 ), ADADELTA (Zeiler 2012)

# Early Stopping

- Beautiful **FREE LUNCH** (no need to launch many different training runs for each value of hyper-parameter for #iterations)
- Monitor validation error during training (after visiting # of training examples = a multiple of validation set size)
- Keep track of parameters with best validation error and report them at the end
- If error does not improve enough (with some patience), stop.

# Long-Term Dependencies and Clipping Trick



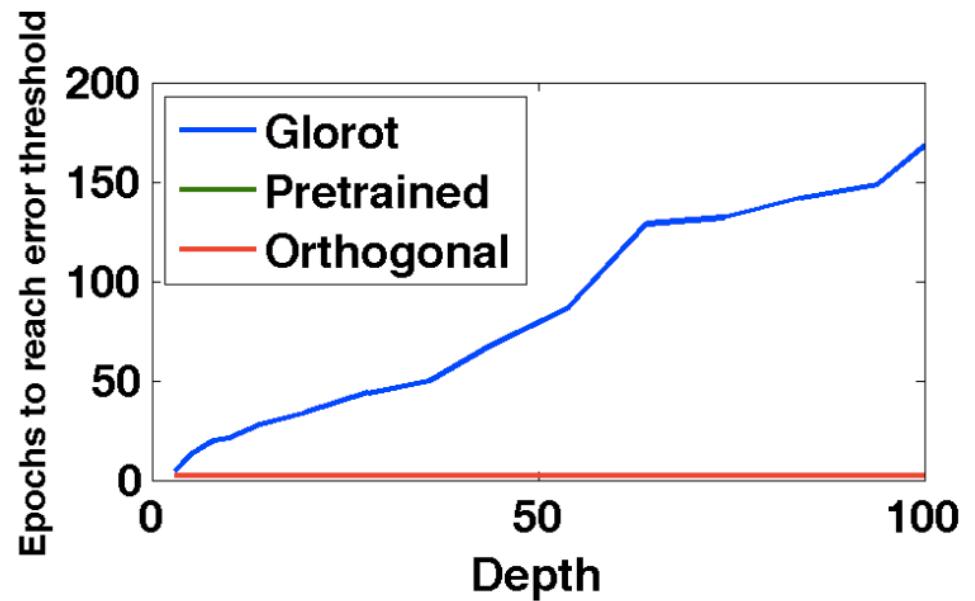
Trick first introduced by Mikolov is to clip gradients to a maximum NORM value.

Makes a big difference in Recurrent Nets ([Pascanu et al ICML 2013](#))

Allows SGD to compete with HF optimization on difficult long-term dependencies tasks. Helped to beat SOTA in text compression, language modeling, speech recognition.

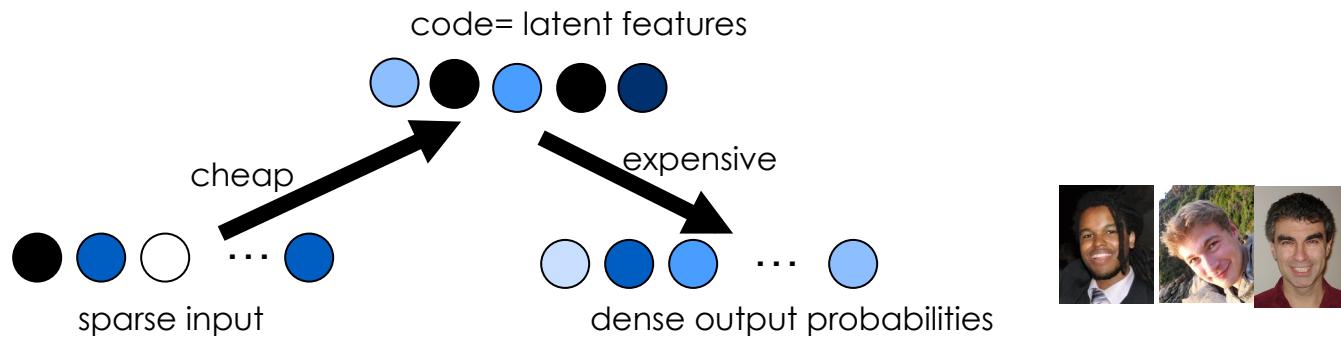
# Orthogonal Initialization Works Even Better

- Auto-encoder pre-training tends to yield orthogonal W
- (Saxe, McClelland & Ganguli ICLR 2014) showed that **very deep** nets initialized with random orthogonal weights are much easier to train
- All singular values = 1



# Handling Large Output Spaces

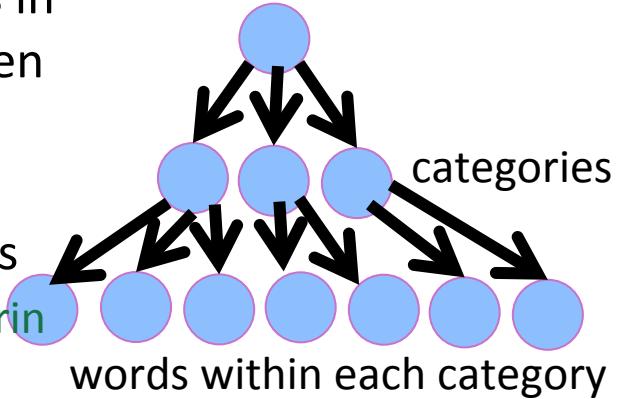
- Auto-encoders and RBMs reconstruct the input, which is sparse and high-dimensional; Language models have a huge output space (1 unit per word).



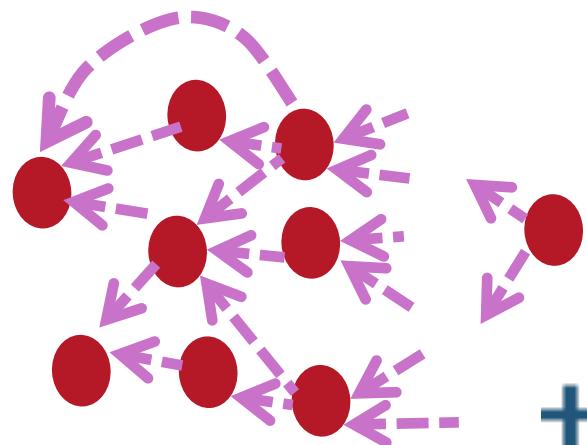
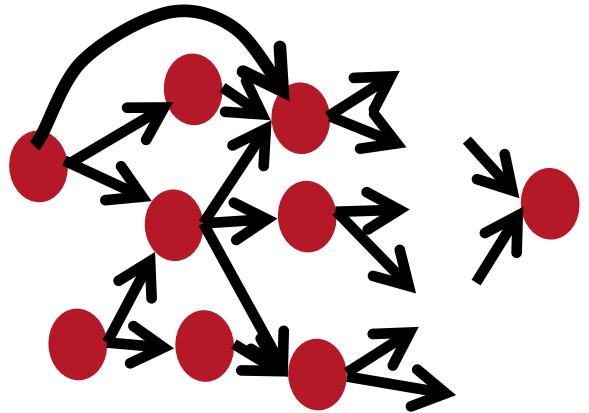
- (Dauphin et al, ICML 2011) Reconstruct the non-zeros in the input, and reconstruct as many randomly chosen zeros, + importance weights



- (Collobert & Weston, ICML 2008) sample a ranking loss
- Decompose output probabilities hierarchically (Morin & Bengio 2005; Blitzer et al 2005; Mnih & Hinton 2007,2009; Mikolov et al 2011)



# Automatic Differentiation



theano

- Makes it easier to quickly and safely try new models.
- Theano Library (python) does it symbolically. Other neural network packages (Torch, Lush) can compute gradients for any given run-time value.

(Bergstra et al SciPy'2010)



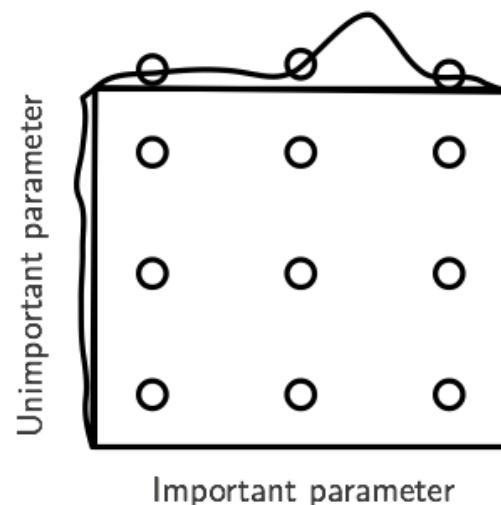
# Random Sampling of Hyperparameters

(Bergstra & Bengio 2012)

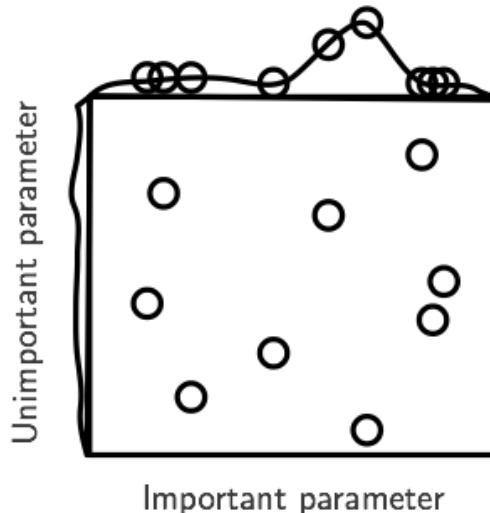


- Common approach: manual + grid search
- Grid search over hyperparameters: simple & wasteful
- Random search: simple & efficient
  - Independently sample each HP, e.g.  $l.rate \sim \exp(U[\log(.1), \log(.0001)])$
  - Each training trial is iid
  - If a HP is irrelevant grid search is wasteful
  - More convenient: ok to early-stop, continue further, etc.

Grid Layout



Random Layout



Part 4

## Challenges & Questions

# Deep Learning Challenges

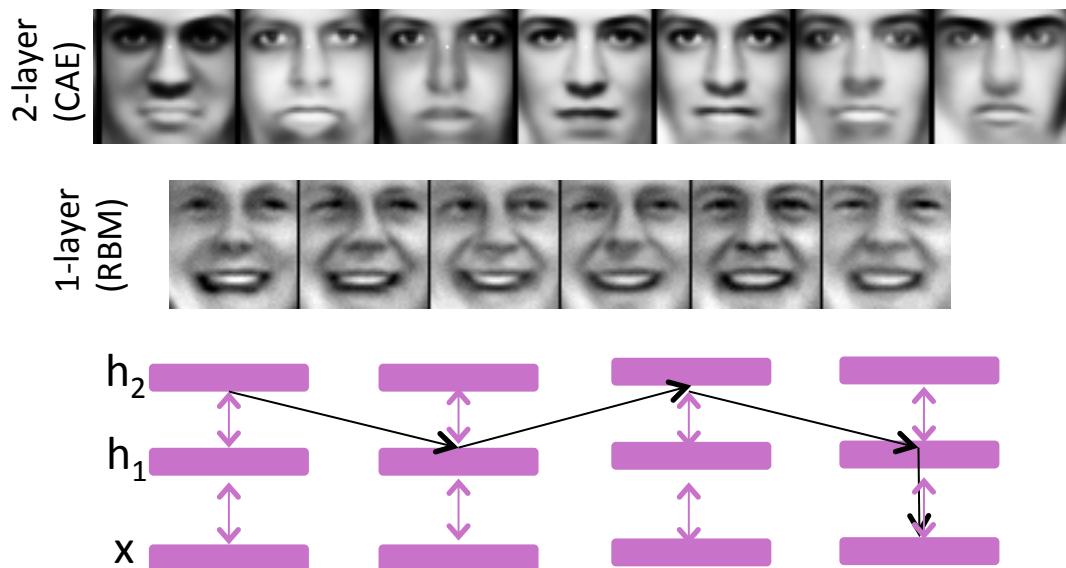
(Bengio, arxiv 1305.0445 Deep learning of representations: Looking forward)

- Computational Scaling
- Optimization & Underfitting
- Intractable Marginalization, Approximate Inference & Sampling
- Disentangling Factors of Variation
- Reasoning & One-Shot Learning of Facts

# Poor Mixing: Depth to the Rescue

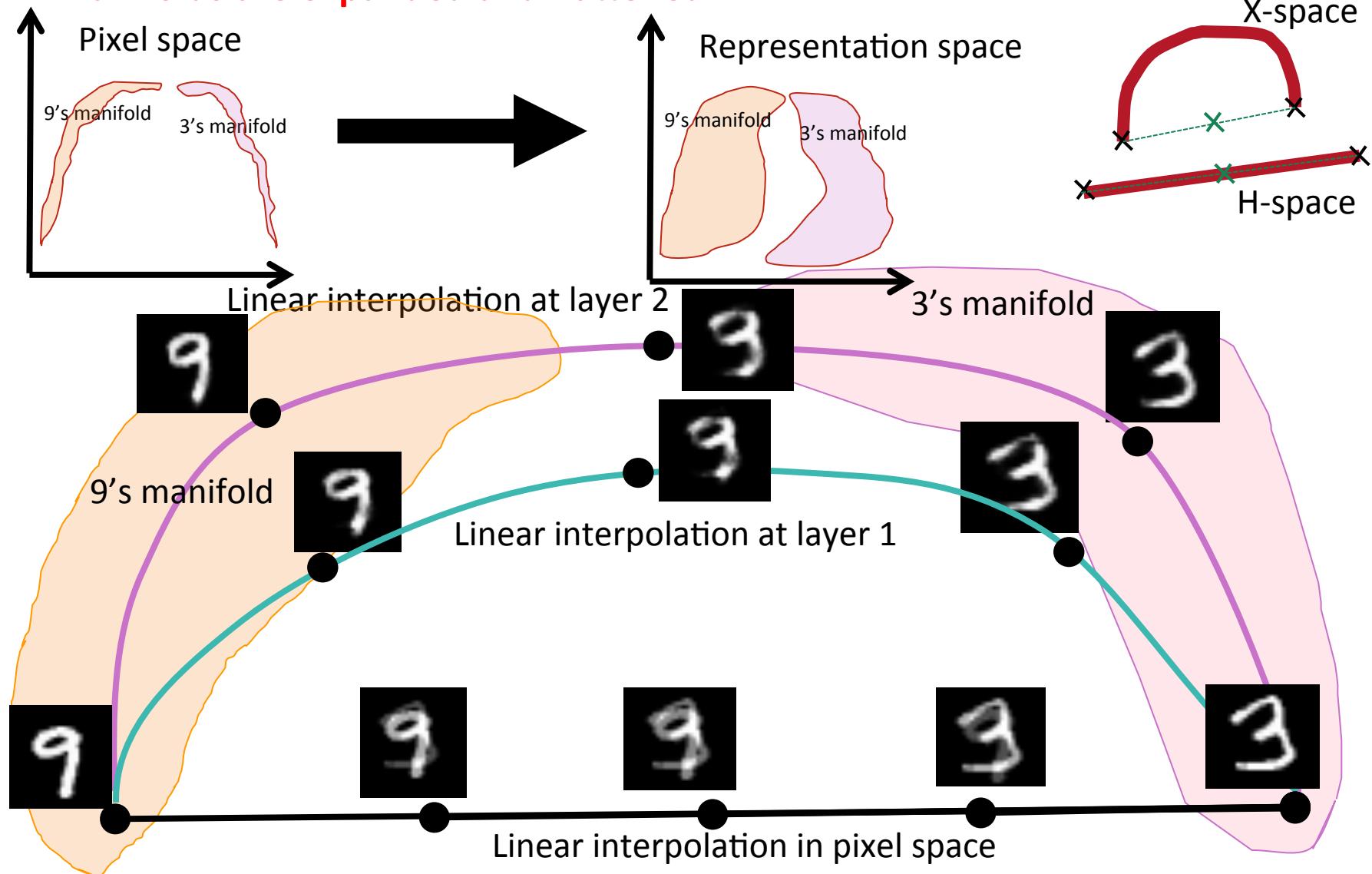
(Bengio et al ICML 2013)

- Sampling from DBNs and stacked Contractive Auto-Encoders:
  1. MCMC sampling from top layer model
  2. Propagate top-level representations to input-level repr.
- Deeper nets visit more modes (classes) faster



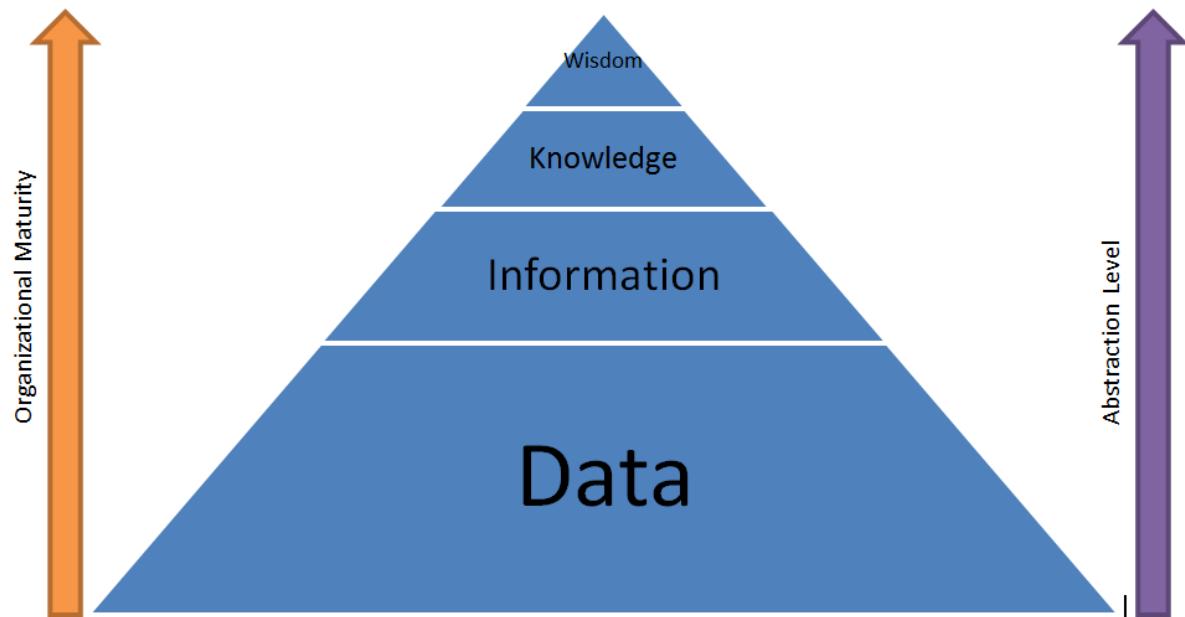
# Space-Filling in Representation-Space

- Deeper representations → abstractions → disentangling
- Manifolds are expanded and flattened



# Learning Multiple Levels of Abstraction

- The big payoff of deep learning is to allow learning higher levels of abstraction
- Higher-level abstractions disentangle the factors of variation, which allows much easier generalization and transfer



# Conclusions

- Deep Learning has matured
  - Int. Conf. on Learning Representation 2013 a huge success!
- Industrial applications (Google, Microsoft, Baidu, Facebook, ...)
- Room for improvement:
  - Scaling computation
  - Optimization
  - Bypass intractable marginalizations
  - More disentangled abstractions
  - Reason from incrementally added facts

If Time Permits...

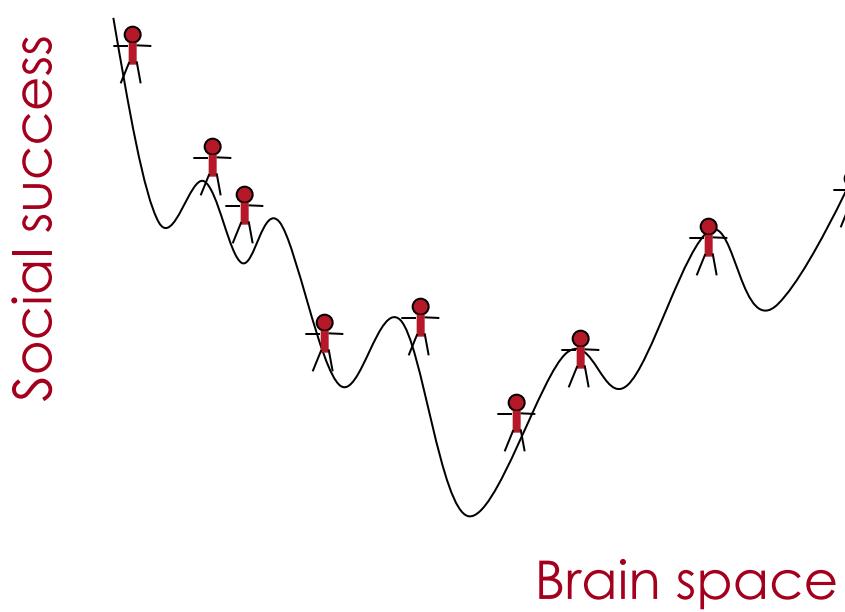
**Issue: underfitting due to combinatorially many poor effective local minima**

where the optimizer gets stuck

# Culture vs Effective Local Minima

Bengio 2013 (also arXiv 2012)

# Parallelized exploration in brain space



- Each brain explores a potential solution
- Instead of exchanging synaptic configurations, exchange ideas through language

# Memes

Genetic Algorithms

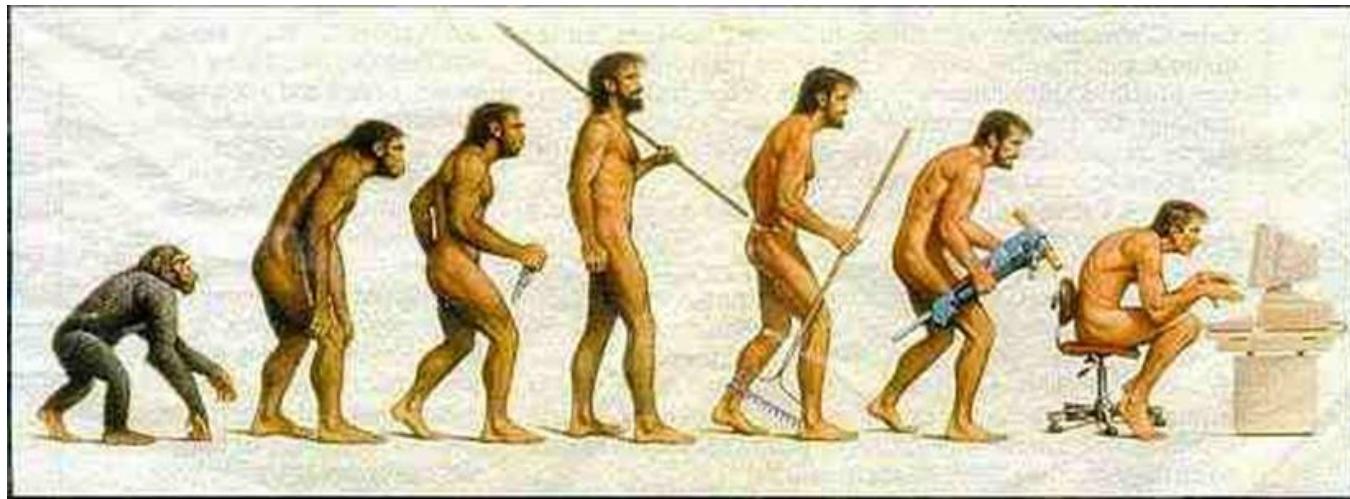
Population of candidate solutions

Recombination mechanism

Evolution of ideas

Brains

Culture and language



## Hypothesis 1

- When the brain of a single biological agent learns, it performs an approximate optimization with respect to some endogenous objective.

## Hypothesis 2

- When the brain of a single biological agent learns, it relies on approximate local descent in order to gradually improve itself.

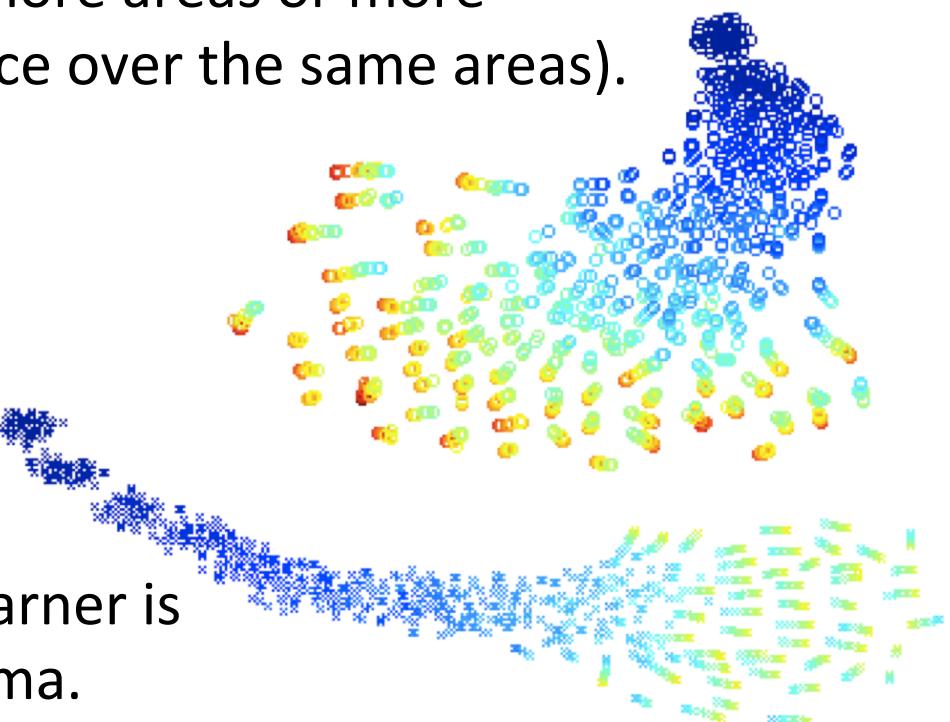
Theoretical and experimental results on deep learning suggest:

## Hypothesis 3

- Higher-level abstractions in brains are represented by deeper computations (going through more areas or more computational steps in sequence over the same areas).

## Hypothesis 4

- Learning of a single human learner is limited by *effective* local minima.



Possibly due to ill-conditioning, but behaves like local min

## Hypothesis 5

- A single human learner is unlikely to discover high-level abstractions by chance because these are represented by a deep sub-network in the brain.

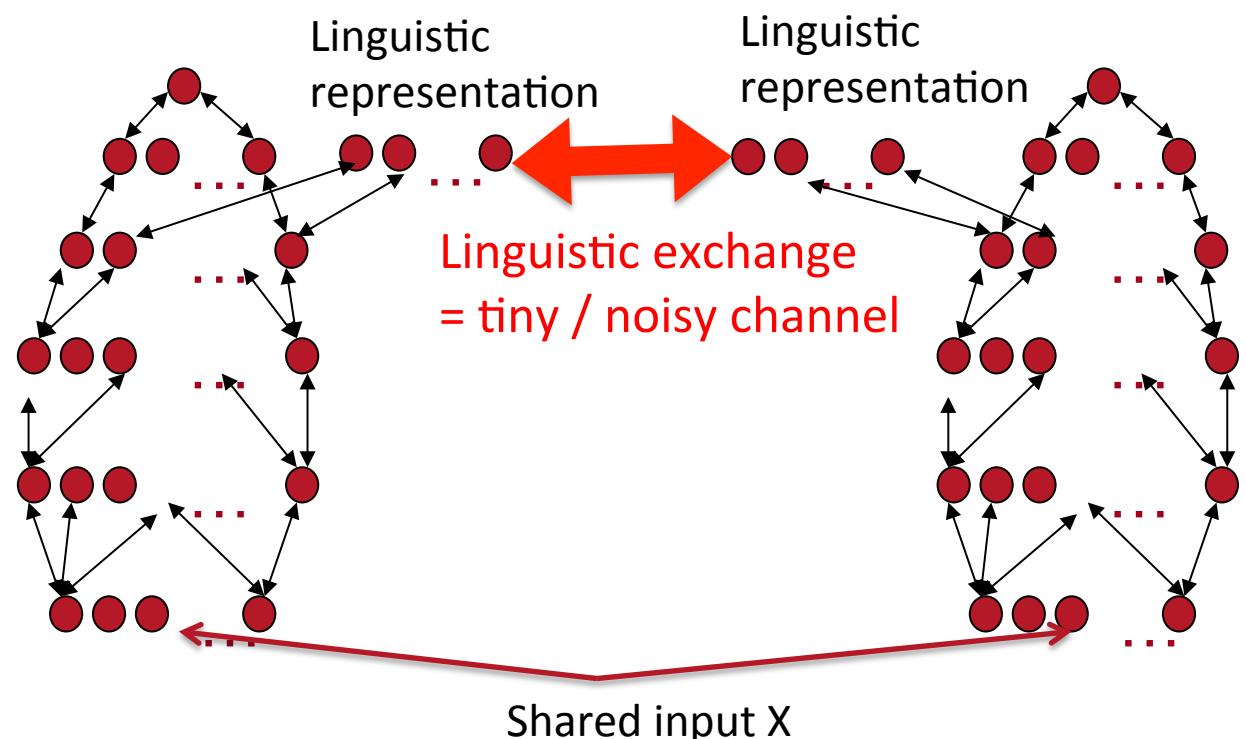
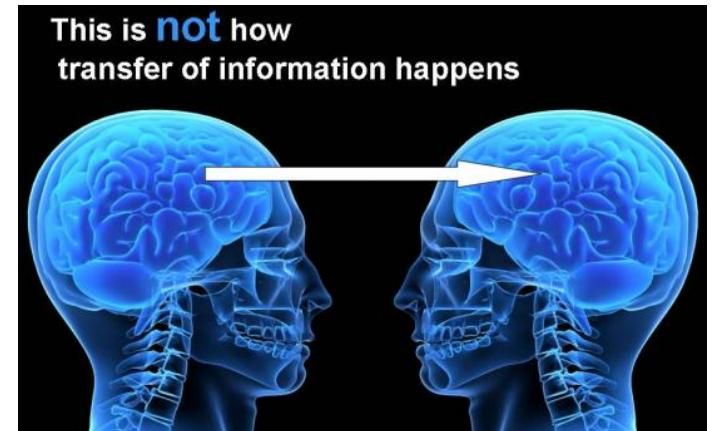
## Hypothesis 6

- A human brain can learn high-level abstractions if guided by the signals produced by other humans, which act as hints or indirect supervision for these high-level abstractions.

Curriculum learning  
(Bengio et al ICLR 2009)

Supporting evidence: (Gulcehre & Bengio ICLR 2013)

# How is one brain transferring abstractions to another brain?



# How do we escape Local minima?

- linguistic inputs = extra examples, summarize knowledge
- criterion landscape easier to optimize (e.g. curriculum learning)
- turn difficult unsupervised learning into easy supervised learning of intermediate abstractions

How could language/education/culture possibly help find the better local minima associated with more useful abstractions?

## Hypothesis 7

More than random search:  
potential exponential speed-up by divide-and-conquer  
combinatorial advantage:  
can combine solutions to independently solved sub-problems

- Language and meme recombination provide an efficient **evolutionary operator**, allowing rapid search in the space of **memes**, that helps humans build up better high-level internal representations of their world.

# From where do new ideas emerge?

- Seconds: **inference** (novel explanations for current x)
- Minutes, hours: **learning** (local descent, like current DL)
- Years, centuries: **cultural evolution** (global optimization, recombination of ideas from other humans)

# Related Tutorials

- Deep Learning tutorials (python): <http://deeplearning.net/tutorials>
- Stanford deep learning tutorials with simple programming assignments and reading list  
<http://deeplearning.stanford.edu/wiki/>
- ACL 2012 Deep Learning for NLP tutorial  
<http://www.socher.org/index.php/DeepLearningTutorial/>
- ICML 2012 Representation Learning tutorial  
<http://www.iro.umontreal.ca/~bengioy/talks/deep-learning-tutorial-2012.html>
- IPAM 2012 Summer school on Deep Learning  
<http://www.iro.umontreal.ca/~bengioy/talks/deep-learning-tutorial-aaai2013.html>
- **More reading:** Paper references in separate pdf, on my web page

# Software

- Theano (Python CPU/GPU) mathematical and deep learning library <http://deeplearning.net/software/theano>
  - Can do automatic, symbolic differentiation
- Senna: POS, Chunking, NER, SRL
  - by Collobert et al. <http://ronan.collobert.com/senna/>
  - State-of-the-art performance on many tasks
  - 3500 lines of C, extremely fast and using very little memory
- Torch ML Library (C++ + Lua) <http://www.torch.ch/>
- Recurrent Neural Network Language Model  
<http://www.fit.vutbr.cz/~imikolov/rnnlm/>
- Recursive Neural Net and RAE models for paraphrase detection, sentiment analysis, relation classification [www.socher.org](http://www.socher.org)

## Software: what's next

- Off-the-shelf SVM packages are useful to researchers from a wide variety of fields (no need to understand RKHS).
- To make deep learning more accessible: release off-the-shelf learning packages that handle hyper-parameter optimization, exploiting multi-core or cluster at disposal of user.
  - Spearmint (Snoek)
  - HyperOpt (Bergstra)