Deep Learning

MLSS 2014
Beijing, China

Yoshua Bengio

June 20, 2014

Université de Montréal

LISA

Outline of the Tutorial

1. Representation Learning, motivations
2. Algorithms
   - Feedforward deep networks
   - Convolutional nets
   - Recurrent and recursive nets
   - Generative nets
3. Practical Considerations and Applications
4. Challenges

Upcoming book: "Deep Learning" http://www.iro.umontreal.ca/~bengioy/dlbook/ for a pdf of the slides and draft chapters of the book.

Ultimate Goals

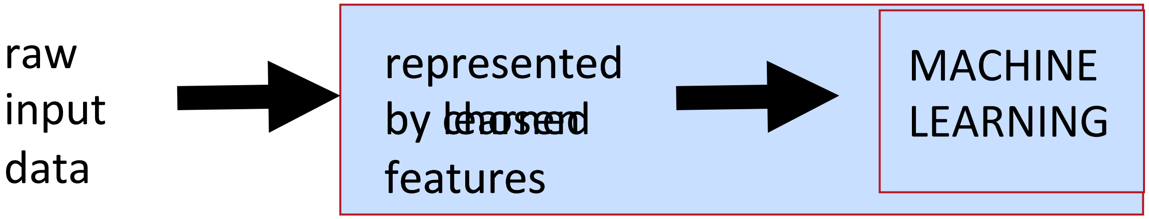
- **AI**
- Needs **knowledge**
- Needs **learning**
  (involves priors + ***optimization***/*search*)
- Needs **generalization**
  (guessing where probability mass concentrates)
- Needs ways to fight the curse of dimensionality
  (exponentially many configurations of the variables to consider)
- Needs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

Part 1

Motivations for Representation Learning and Deep Learning

Representation Learning

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



raw input data

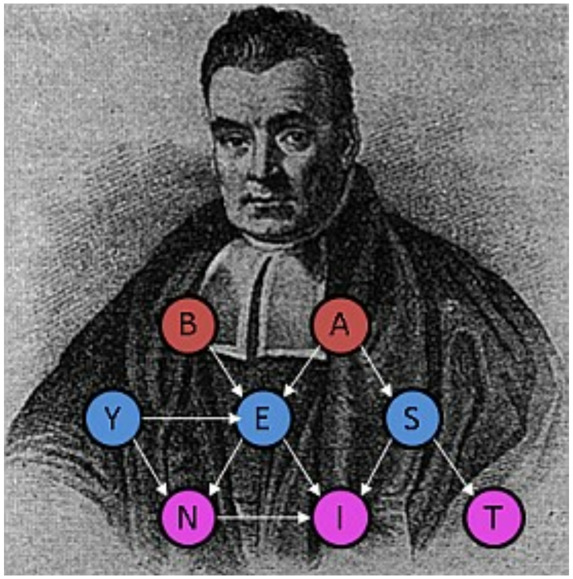
represented by chosen learned features

MACHINE LEARNING

- Handcrafting features vs learning them

- Good representation?
- guesses

  the features / factors / causes



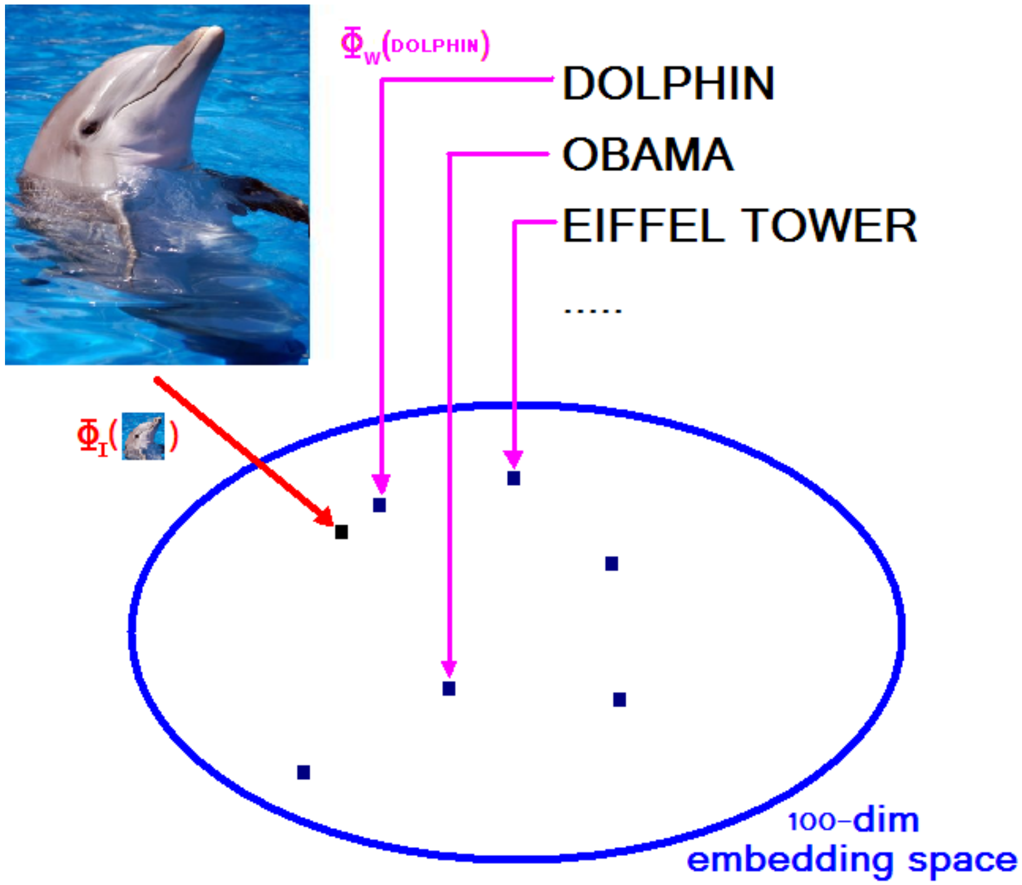
Google Image Search:

Different object types represented in the same space

Google:

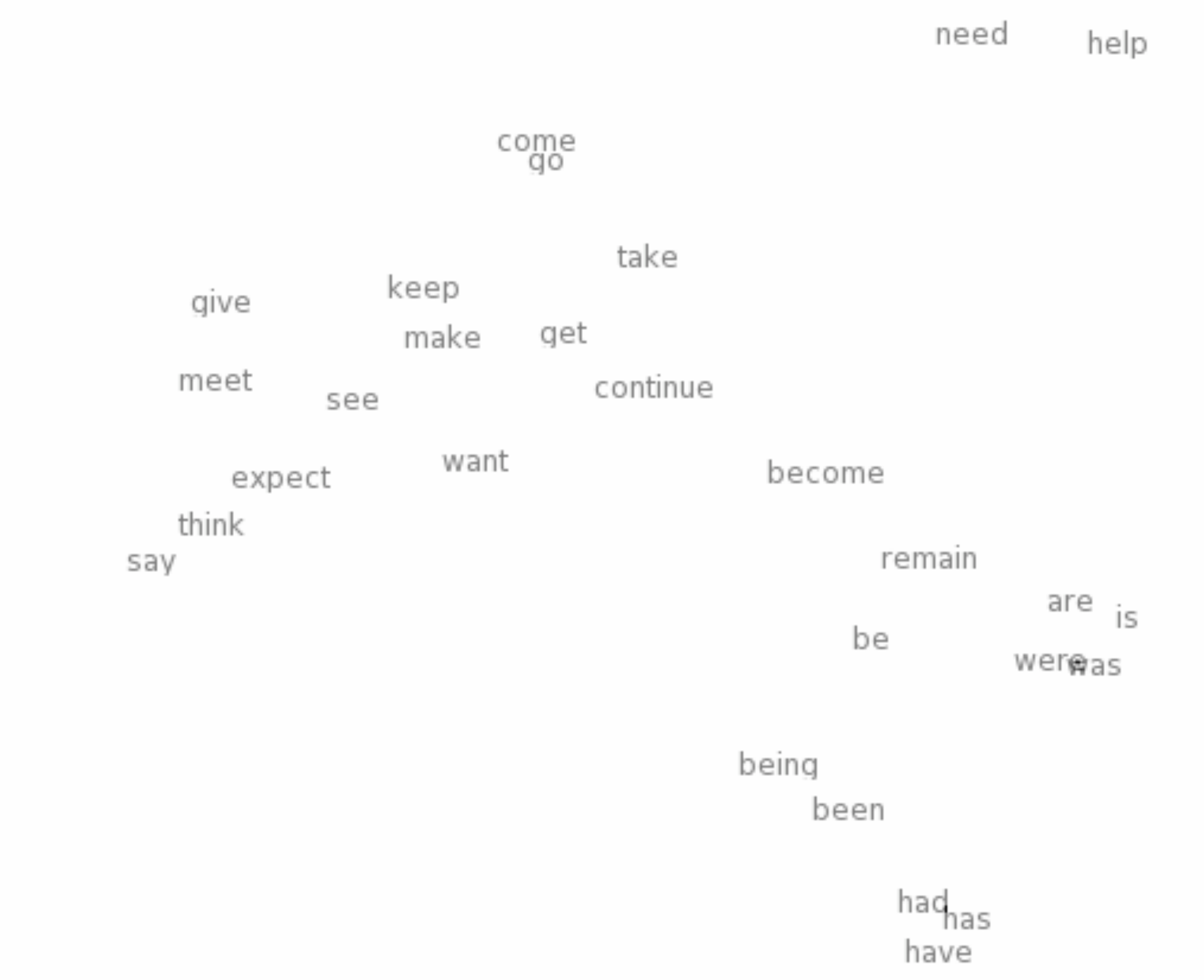
S. Bengio, J. Weston & N. Usunier

(IJCAI 2011, NIPS'2010, JMLR 2010, MLJ 2010)



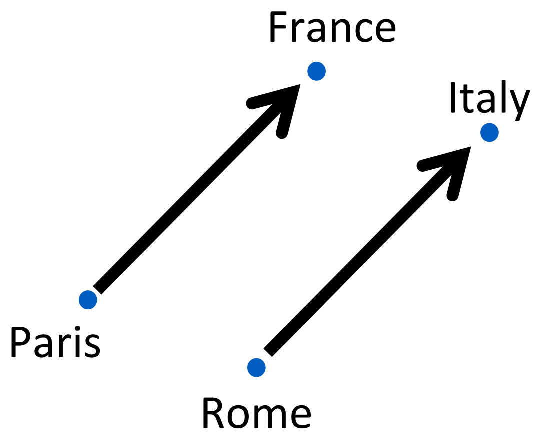
Learn $\Phi_I(\cdot)$ and $\Phi_W(\cdot)$ to optimize precision@k.

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 ≈ Man – Woman
- Paris – France + Italy ≈ Rome



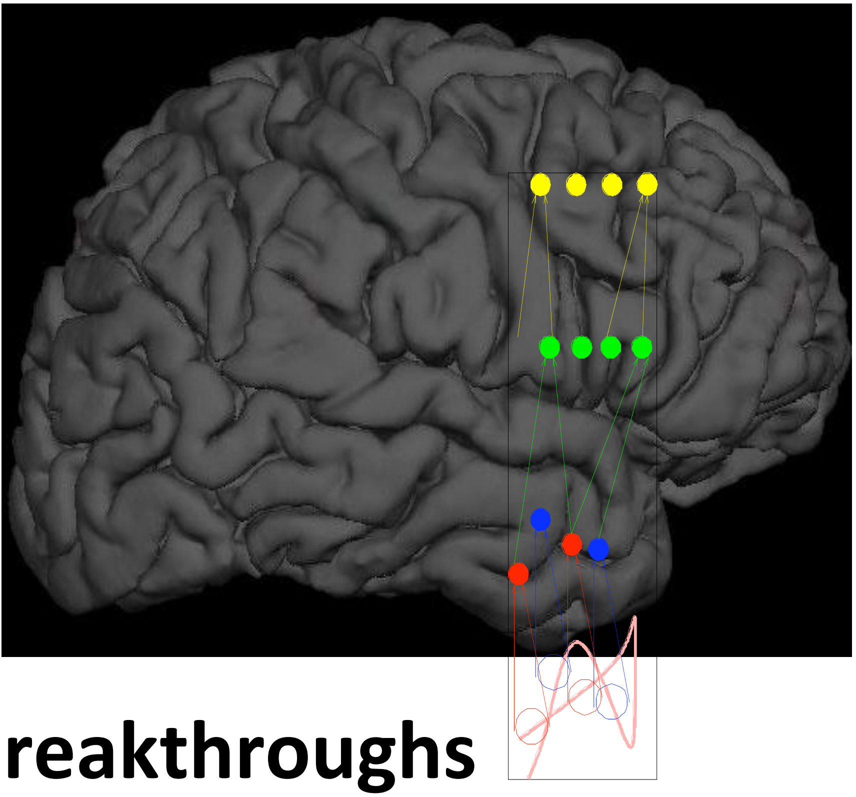
Learning multiple levels of representation

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 compose them**



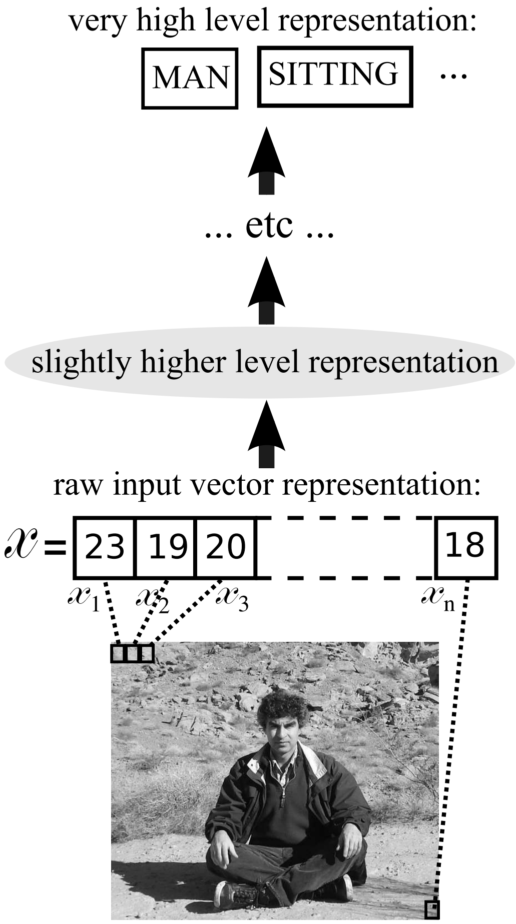
It works! Speech + vision breakthroughs

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?)



very high level representation:

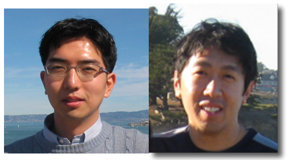
MAN SITTING ...

... etc ...

slightly higher level representation

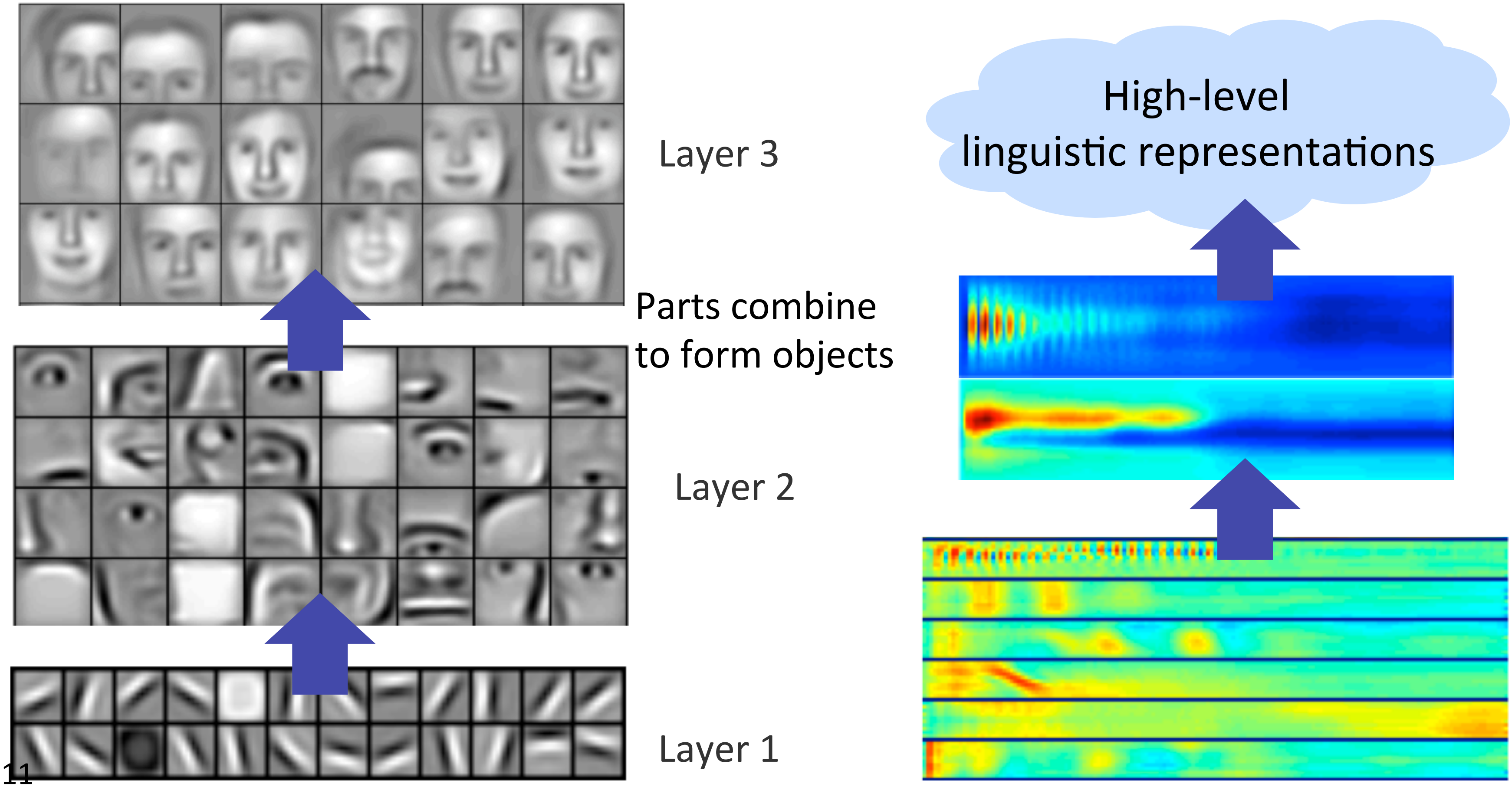
raw input vector representation:

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

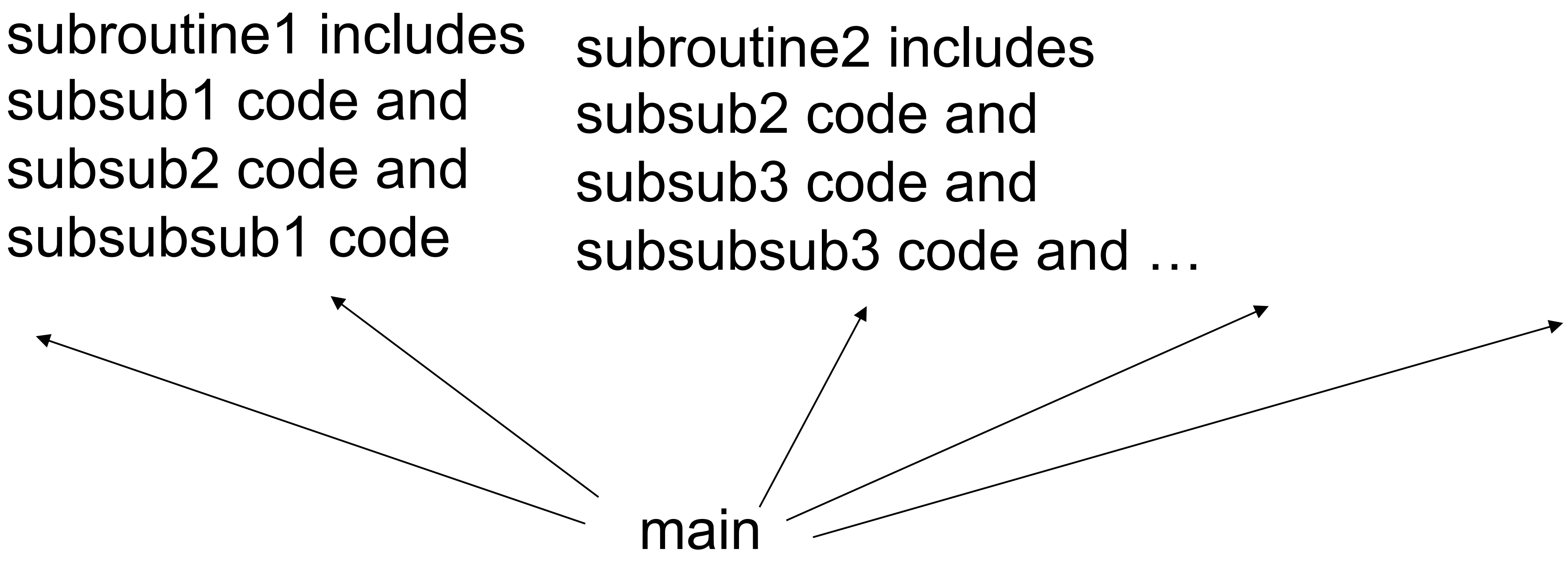
Parts combine to form objects

Layer 2

Layer 1

High-level linguistic representations

Prior: underlying factors & concepts compactly expressed w/ multiple levels of abstraction

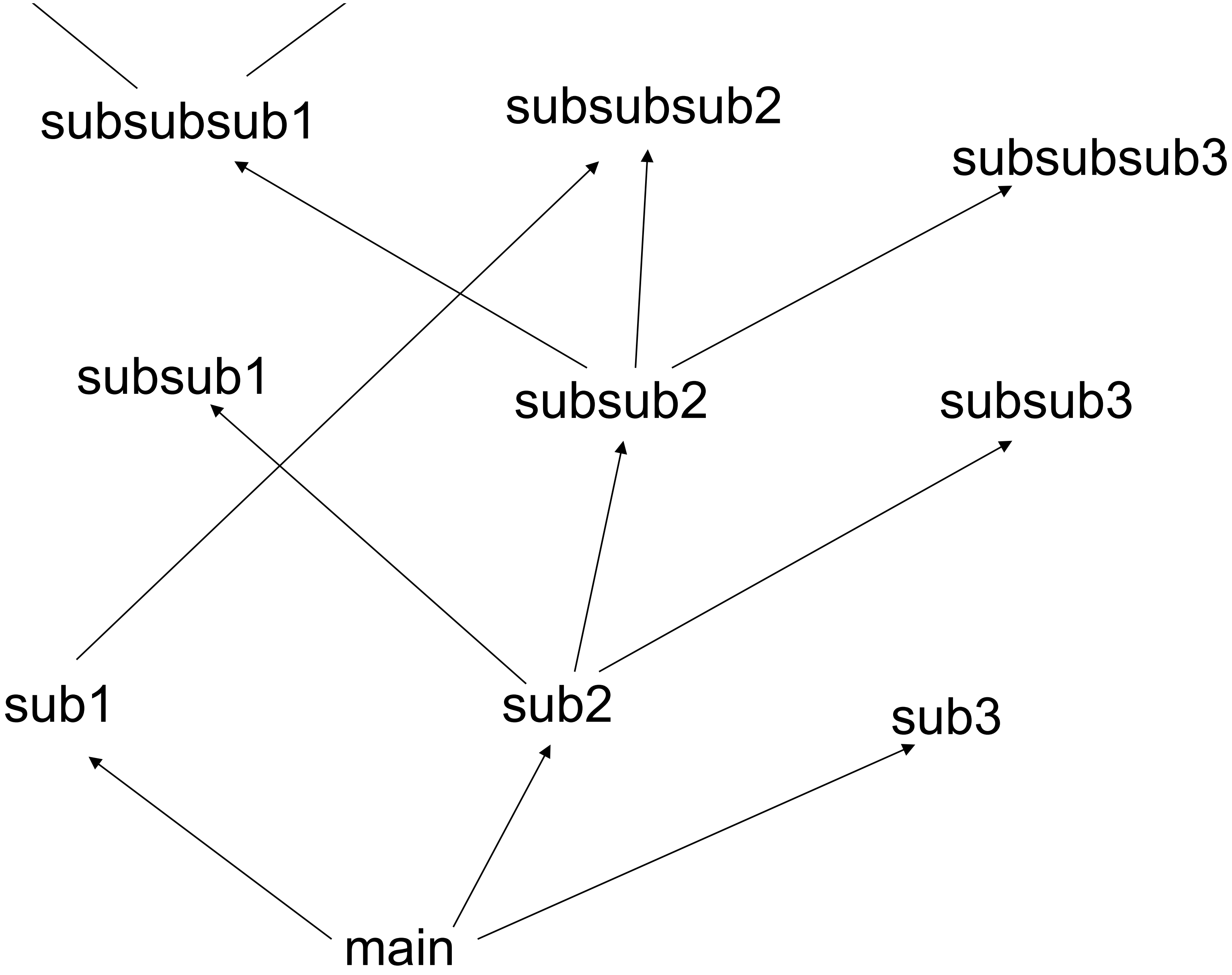


subroutine1 includes
subsub1 code and
subsub2 code and
subsubsub1 code

subroutine2 includes
subsub2 code and
subsub3 code and
subsubsub3 code and …

main

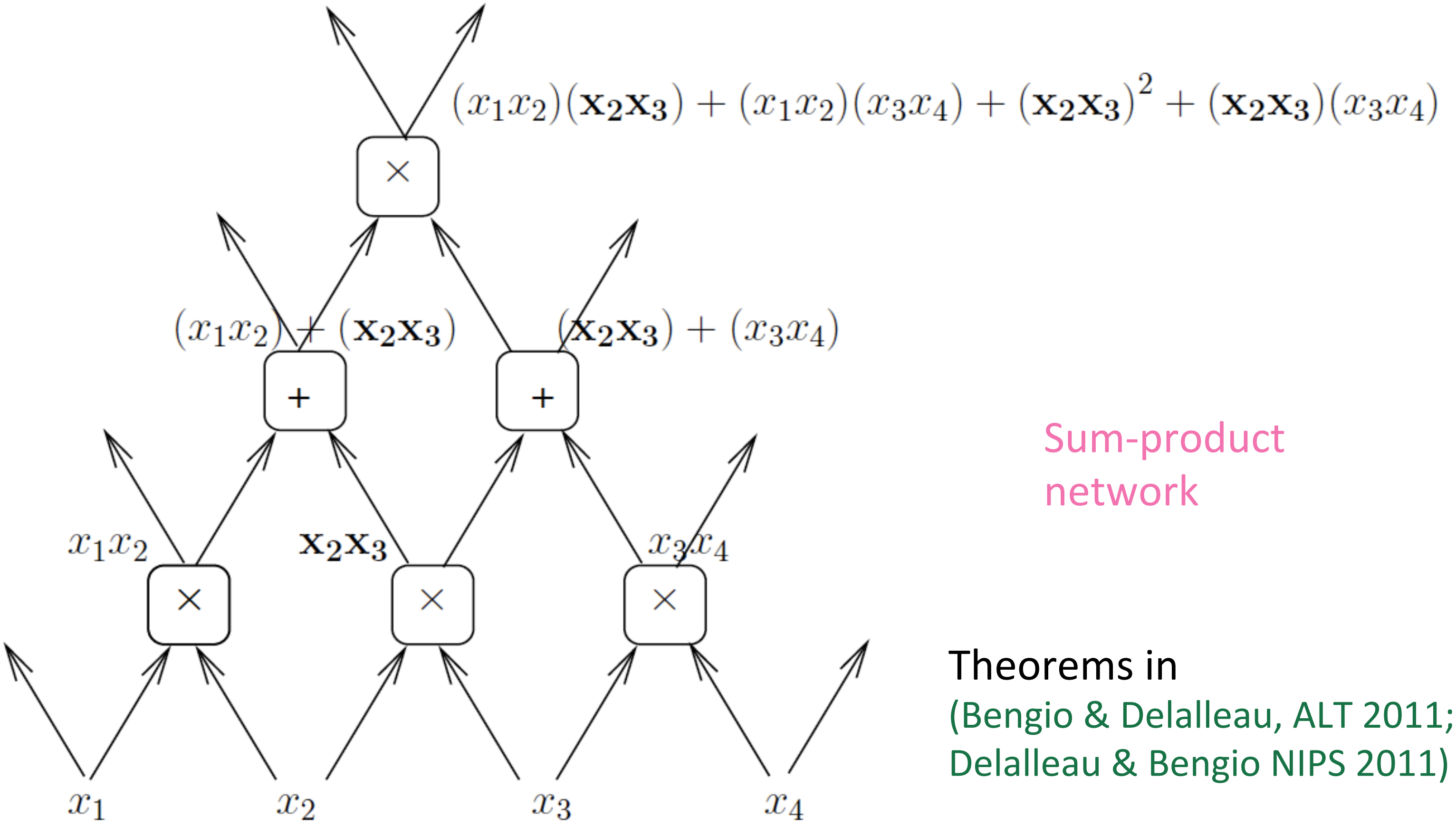
“Shallow” computer program



“Deep” computer program

Sharing Components in a Deep Architecture

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


$(x_1x_2)(\mathbf{x_2x_3}) + (x_1x_2)(x_3x_4) + (\mathbf{x_2x_3})^2 + (\mathbf{x_2x_3})(x_3x_4)$
×
$(x_1x_2) + (\mathbf{x_2x_3})$
+
$(\mathbf{x_2x_3}) + (x_3x_4)$
+
x_1x_2
$\mathbf{x_2x_3}$
x_3x_4
×
×
×
x_1
x_2
x_3
x_4

Sum-product network

Theorems in
(Bengio & Delalleau, ALT 2011;
Delalleau & Bengio NIPS 2011)

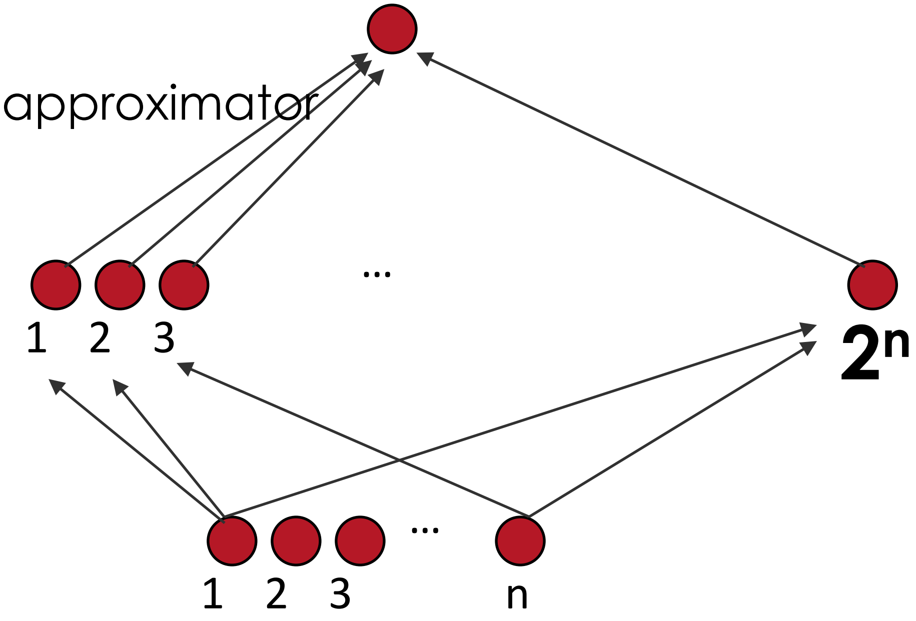
Deep Architectures are More Expressive

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

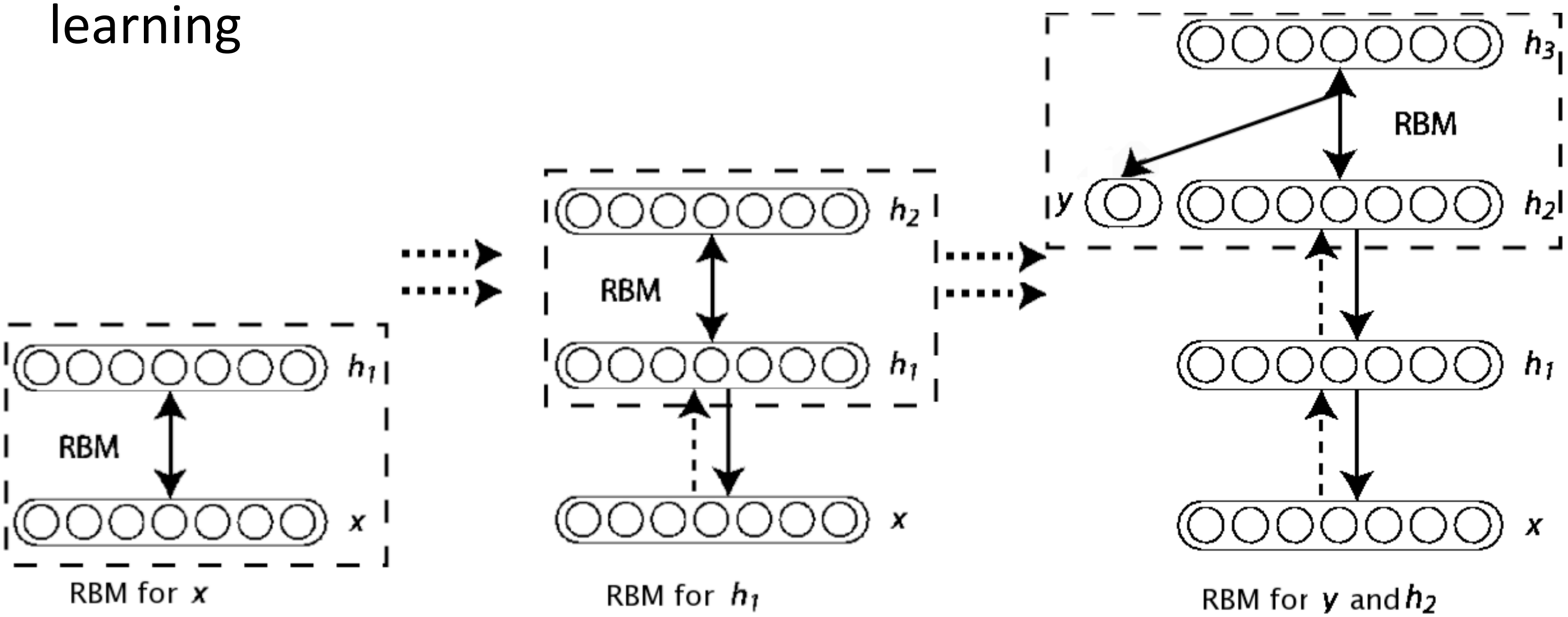
- Ability to train deep architectures by using layer-wise unsupervised learning, whereas previous purely supervised attempts had failed

- Unsupervised feature learners:
  - RBMs
  - Auto-encoder variants
  - Sparse coding variants

Bengio
Montréal
Toronto
Hinton
Le Cun
New York

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

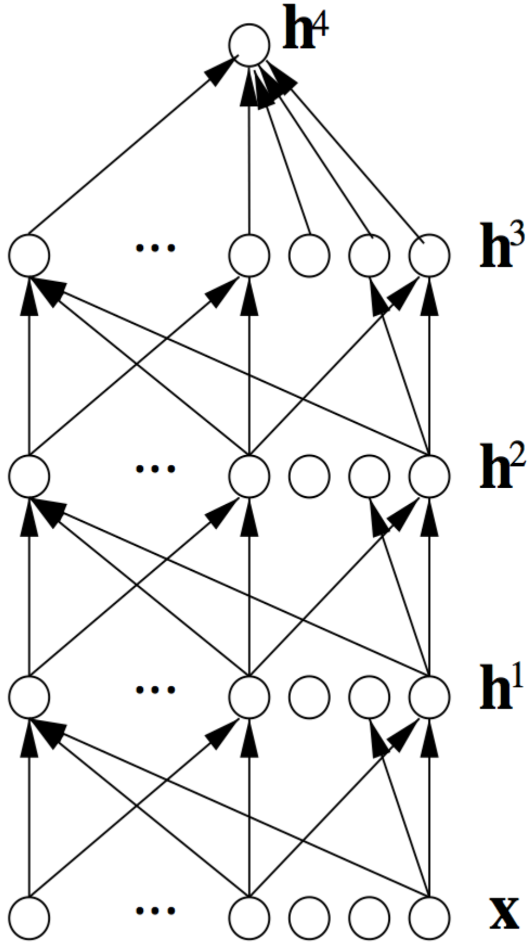
Deep Supervised Neural Nets

- 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

Yoshua Bengio. *Image: C*

WIRED

Researcher Dreams Up Machines That Learn Without Humans
06.27.13

The New York Times

Scientists See Promise in Deep-Learning Programs
John Markoff
November 23, 2012

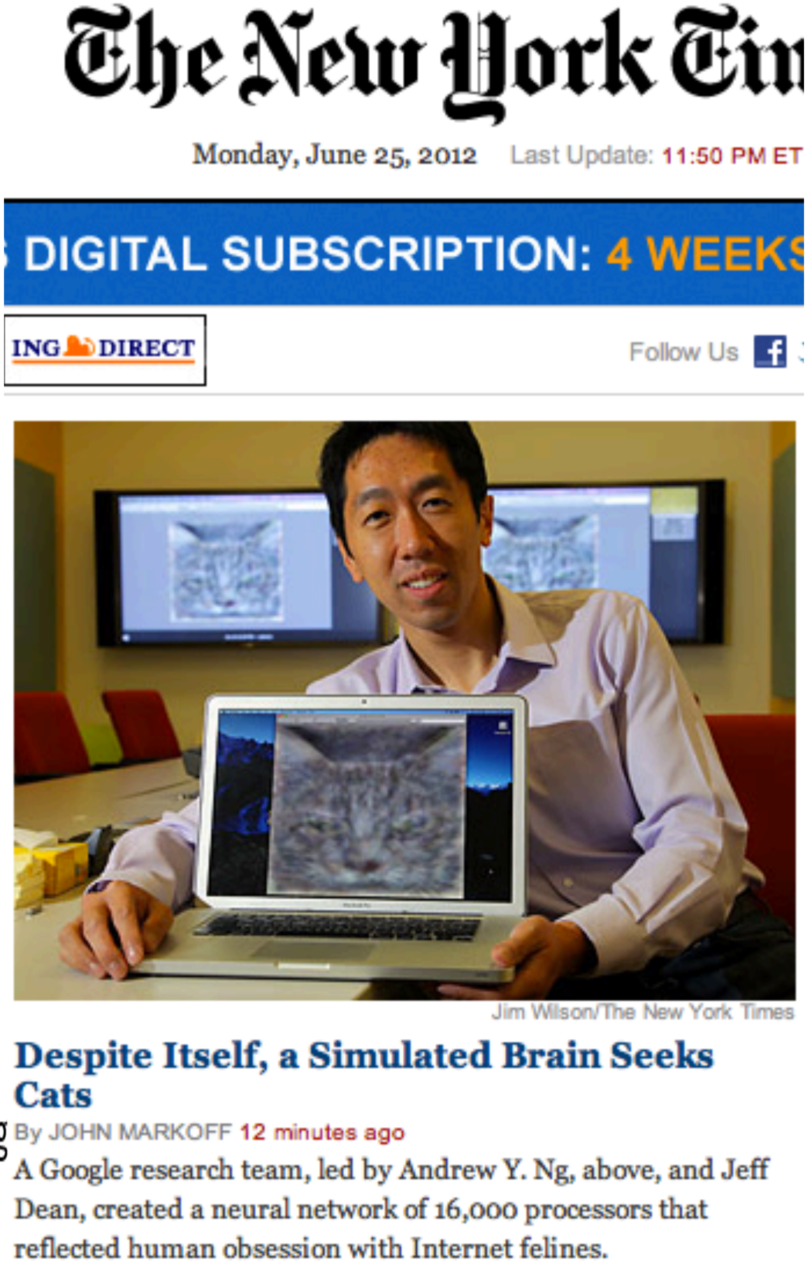
THE GLOBE AND MAIL
CANADA'S NATIONAL NEWSPAPER • FOUNDED 1844

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



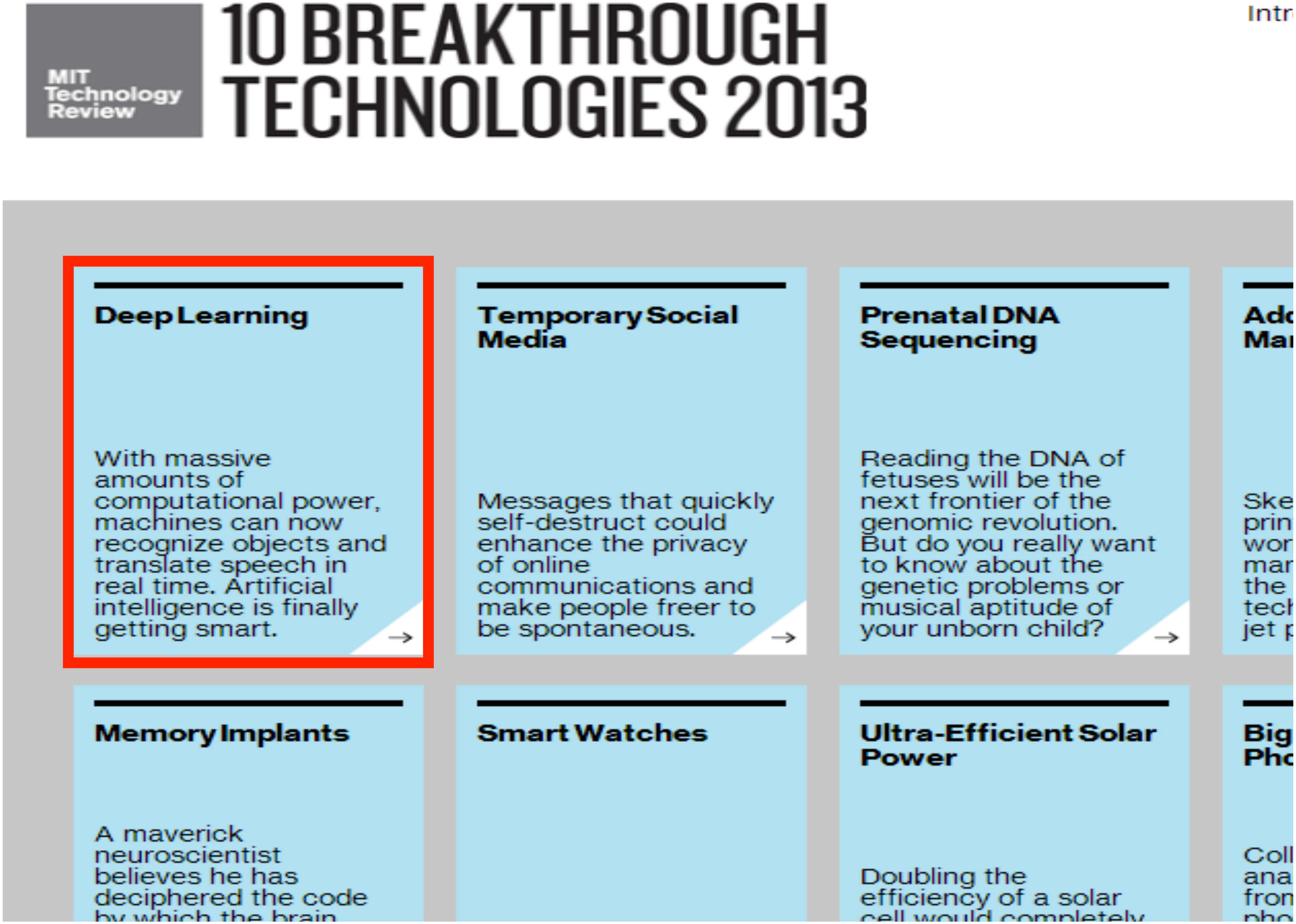
The New York Tim

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

DIGITAL SUBSCRIPTION: 4 WEEKS

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.



MIT Technology Review

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? →

Add Mar

Ske prin wor man the tech jet p

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

Big Pho

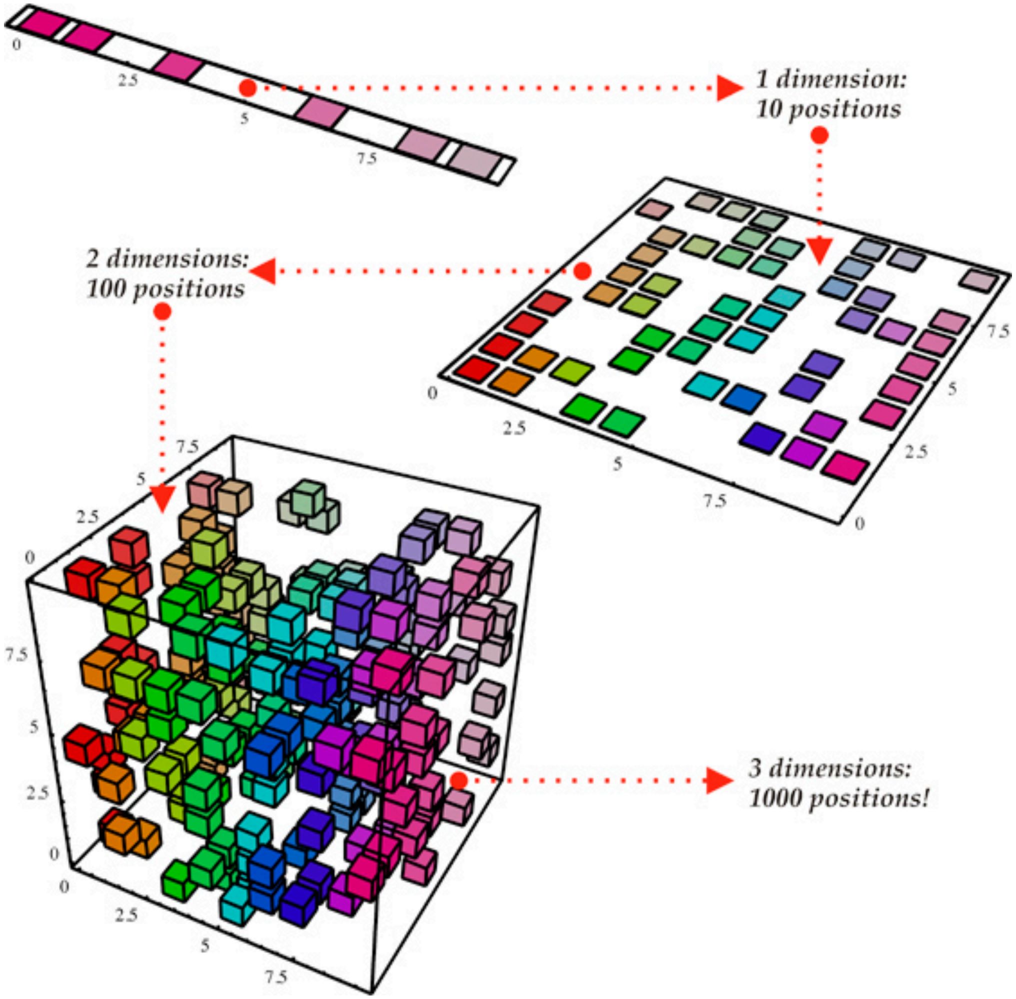
Coll ana from pho

Back to ML Basics

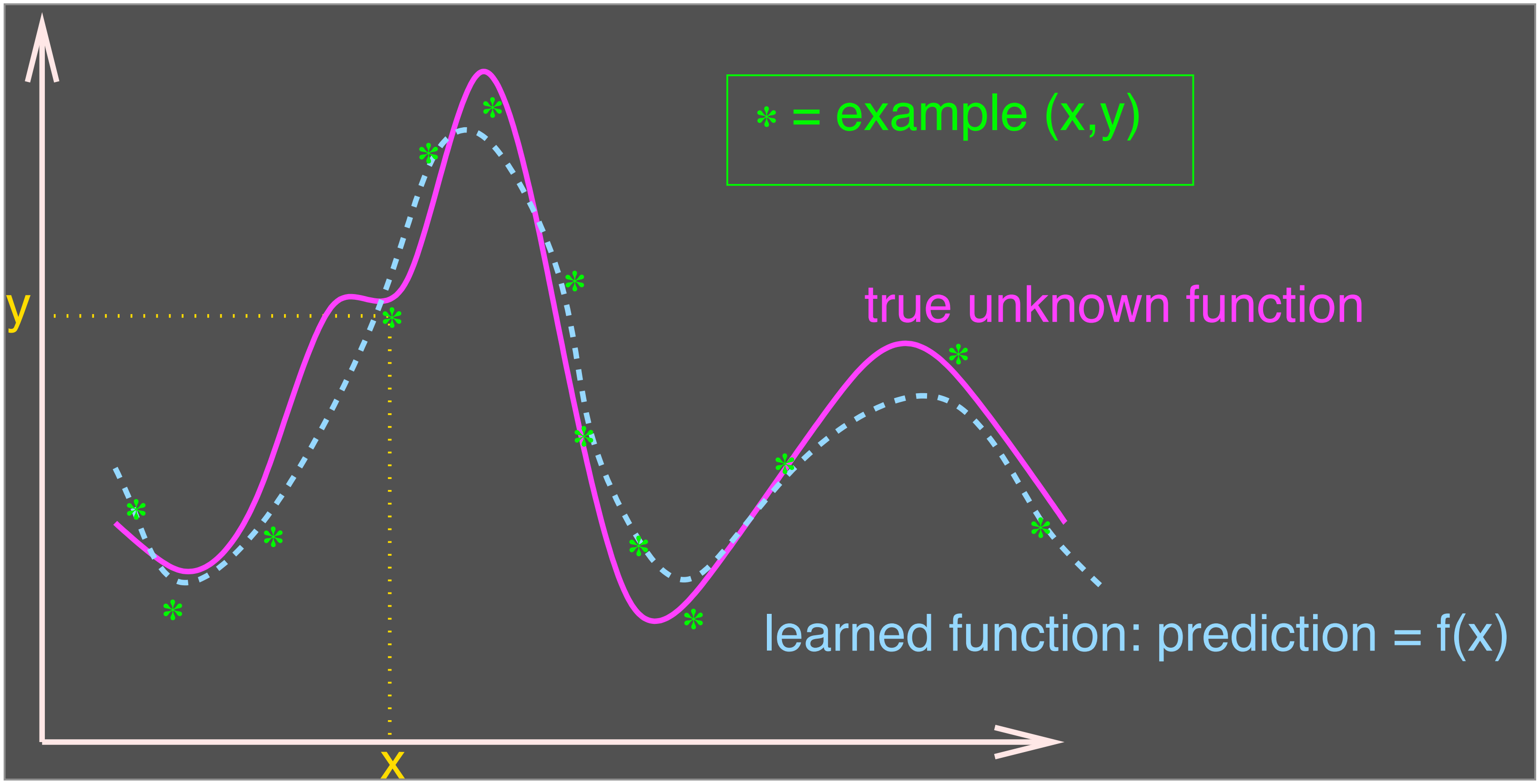
ML 101. What We Are Fighting Against: The Curse of Dimensionality

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

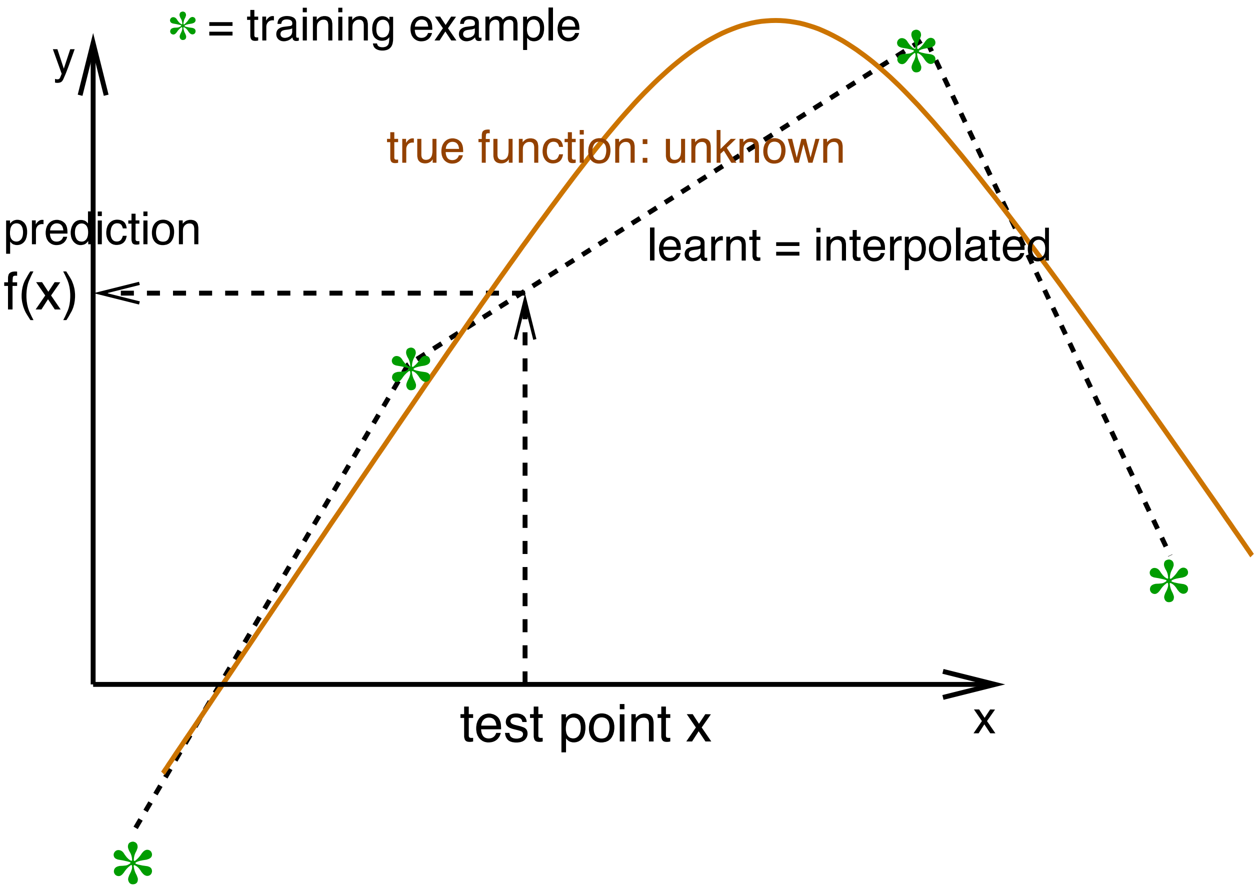
Classical solution: hope for a smooth enough target function, or make it smooth by handcrafting good features / kernel



Easy Learning

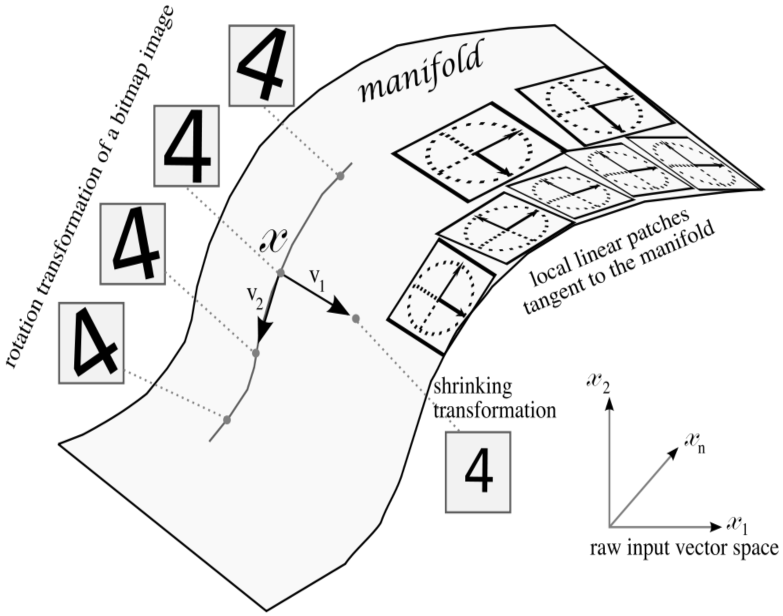


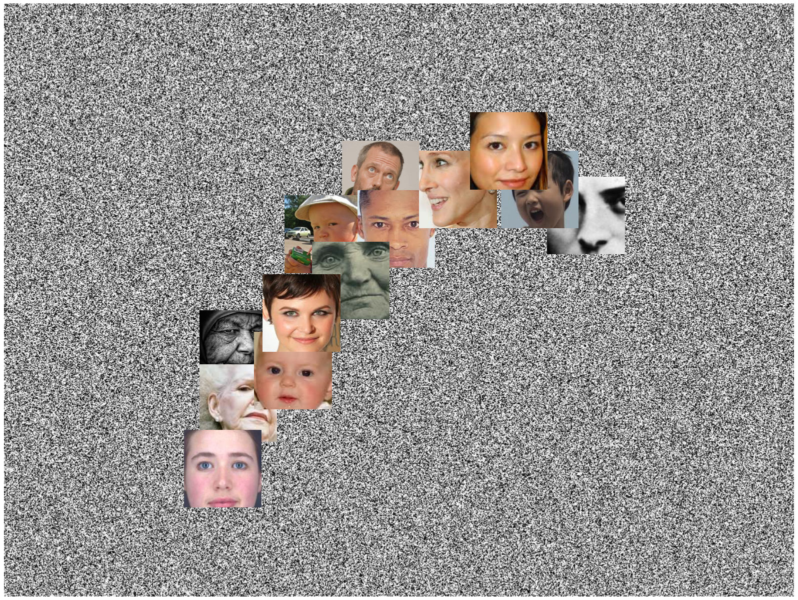
Local Smoothness Prior: Locally Capture the Variations



For AI Tasks: Manifold structure

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

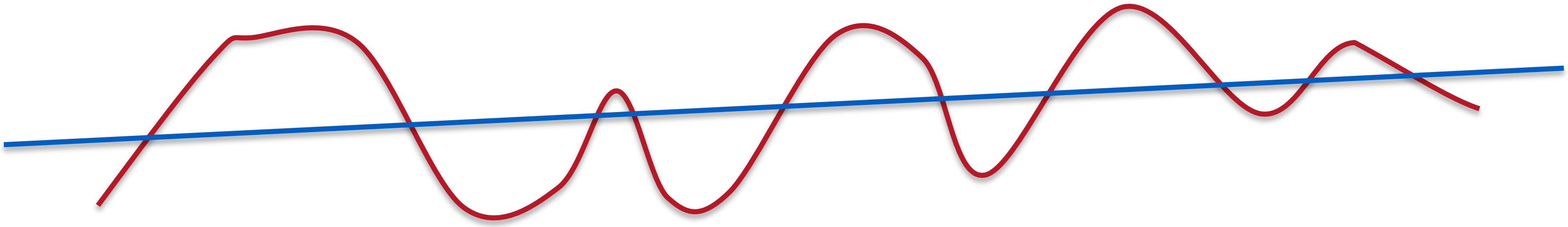




Not Dimensionality so much as Number of Variations

(Bengio, Dellalleau & Le Roux 2007)

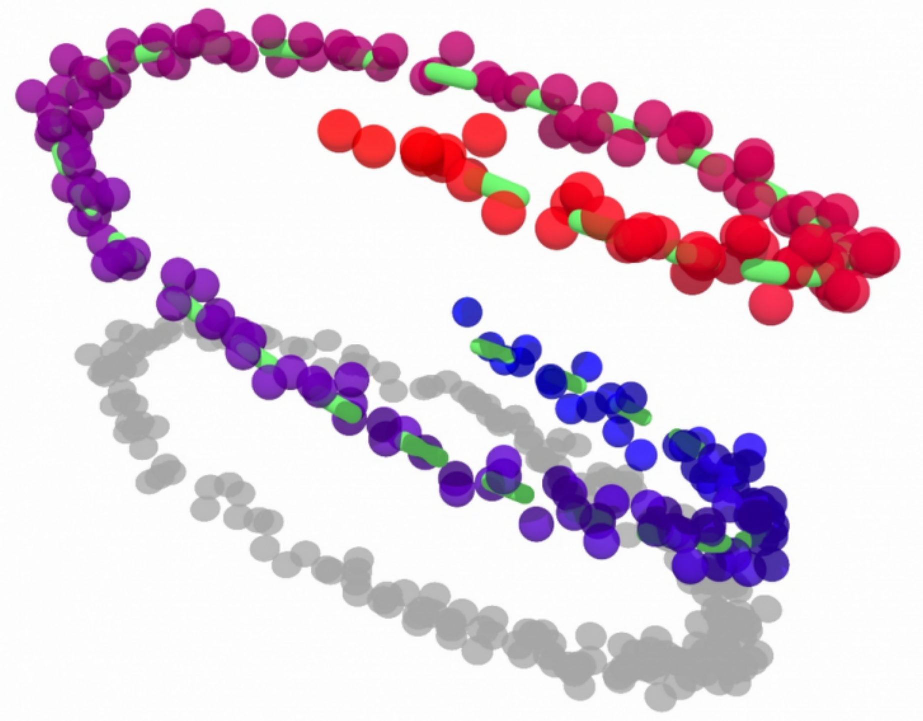
- Theorem: Gaussian kernel machines need at least *k* examples to learn a function that has *2k* zero-crossings along some line



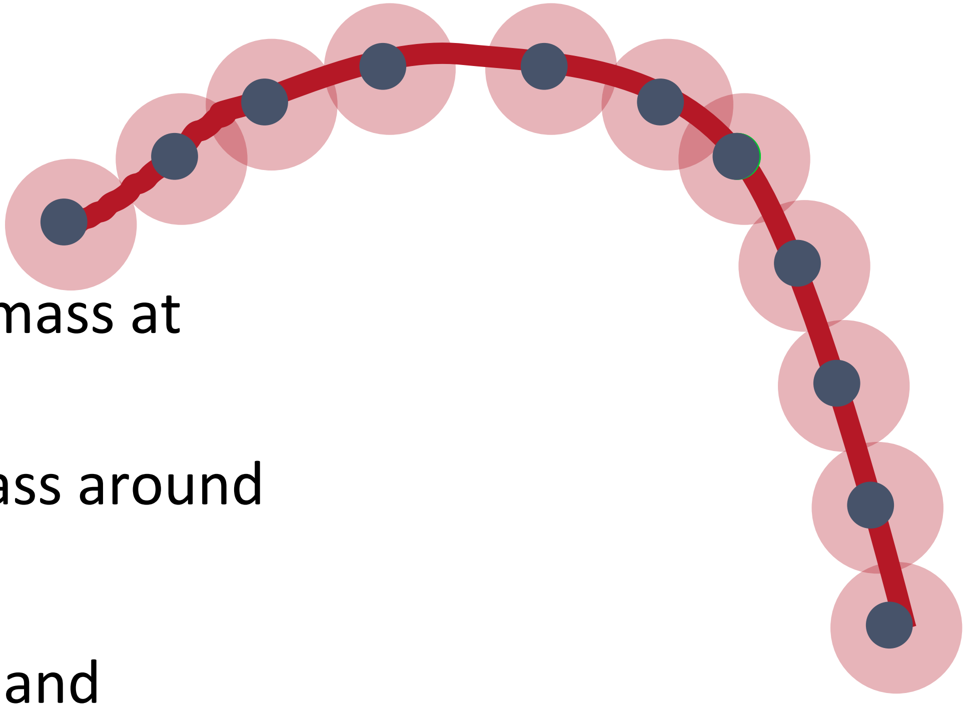
- Theorem: For a Gaussian kernel machine to learn some maximally varying functions over *d* inputs requires O(2^d) examples

Geometrical view on machine learning

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



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

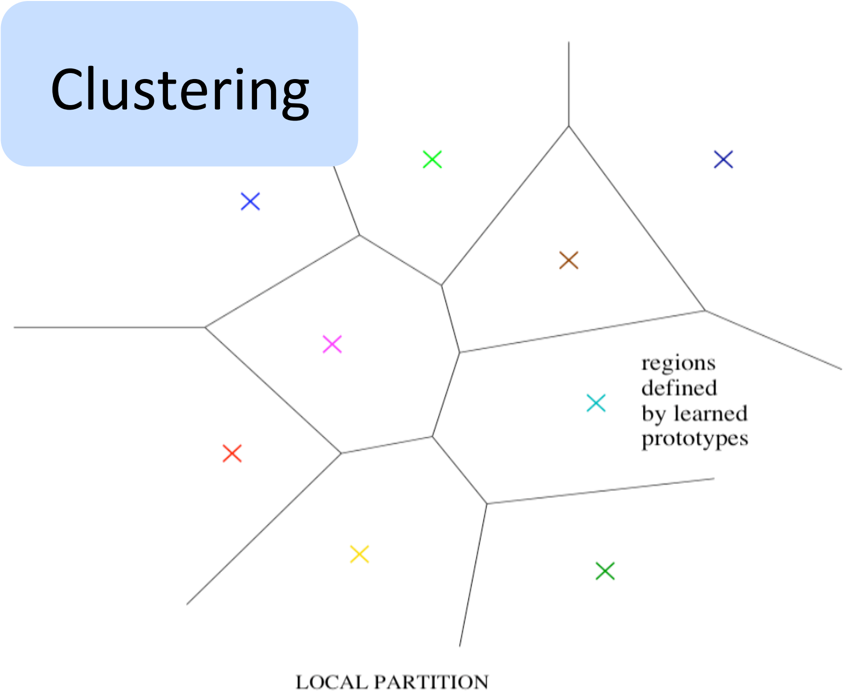
Exploiting compositionality gives an exponential gain in representational power

Distributed representations / embeddings: feature learning

Deep architecture: multiple levels of feature learning

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

Non-distributed representations



- 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**

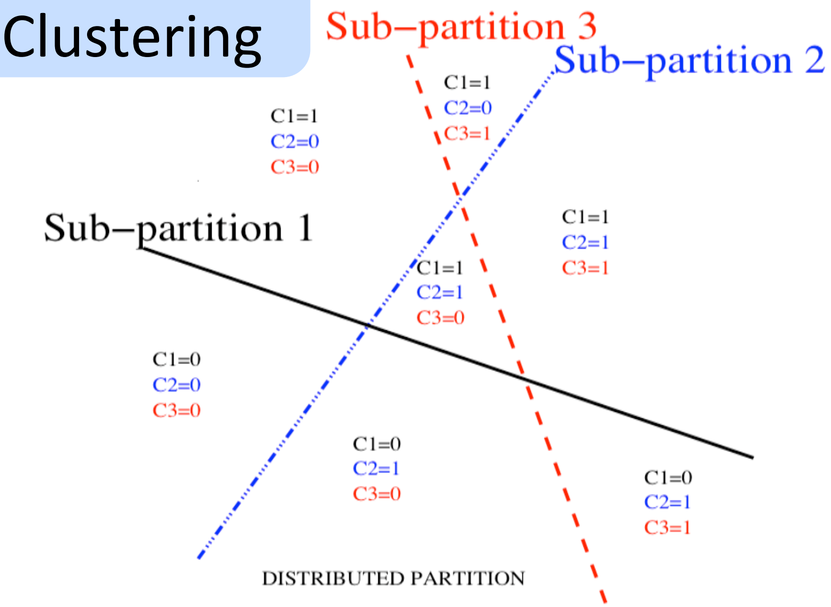
→ No non-trivial generalization to regions without examples

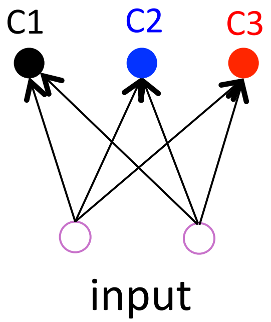
The need for distributed representations

Multi-Clustering

- 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**

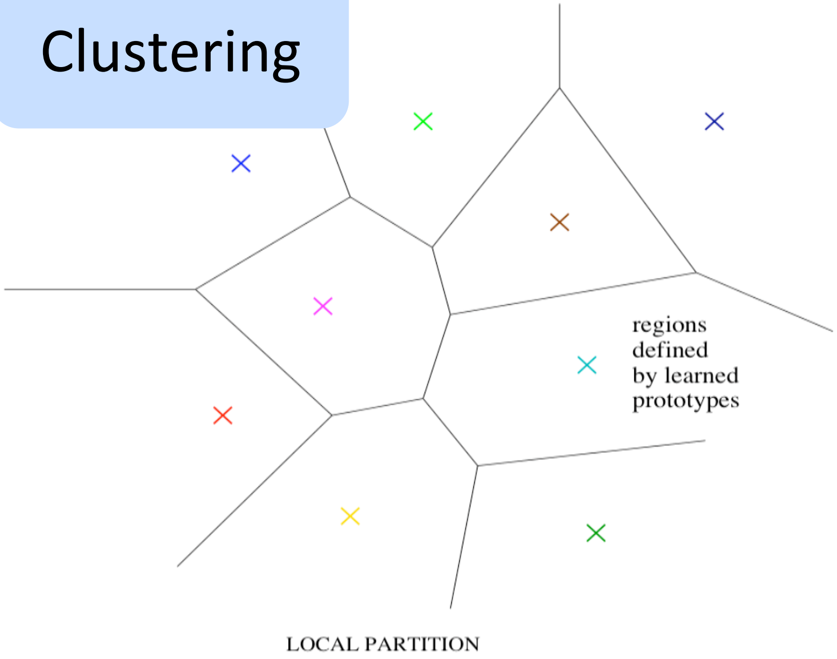




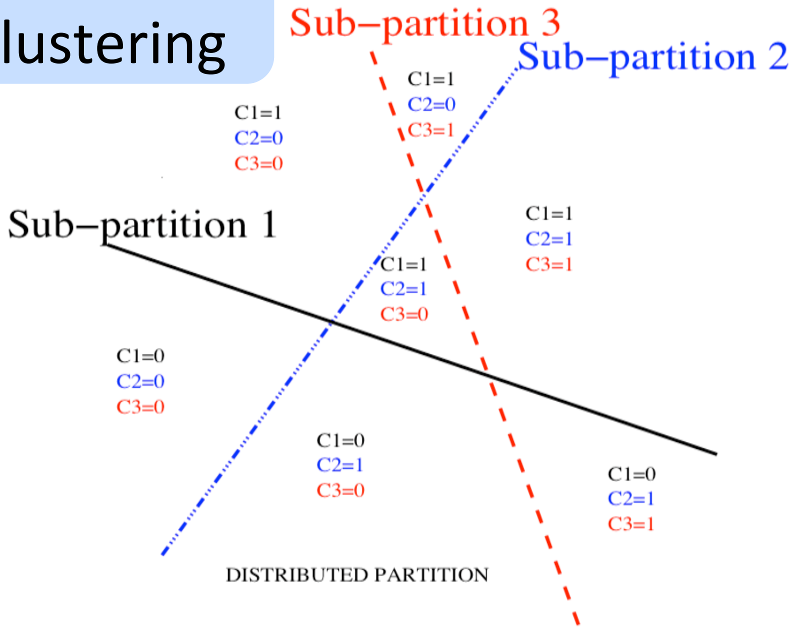
Non-mutually exclusive features/ attributes create a combinatorially large set of distinguiable configurations

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

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

Labels cannot possibly provide enough information

Most information acquired in an **unsupervised** fashion

How do humans generalize from very few examples?

- They **transfer** knowledge from previous learning:
  - Representations
  - Explanatory factors

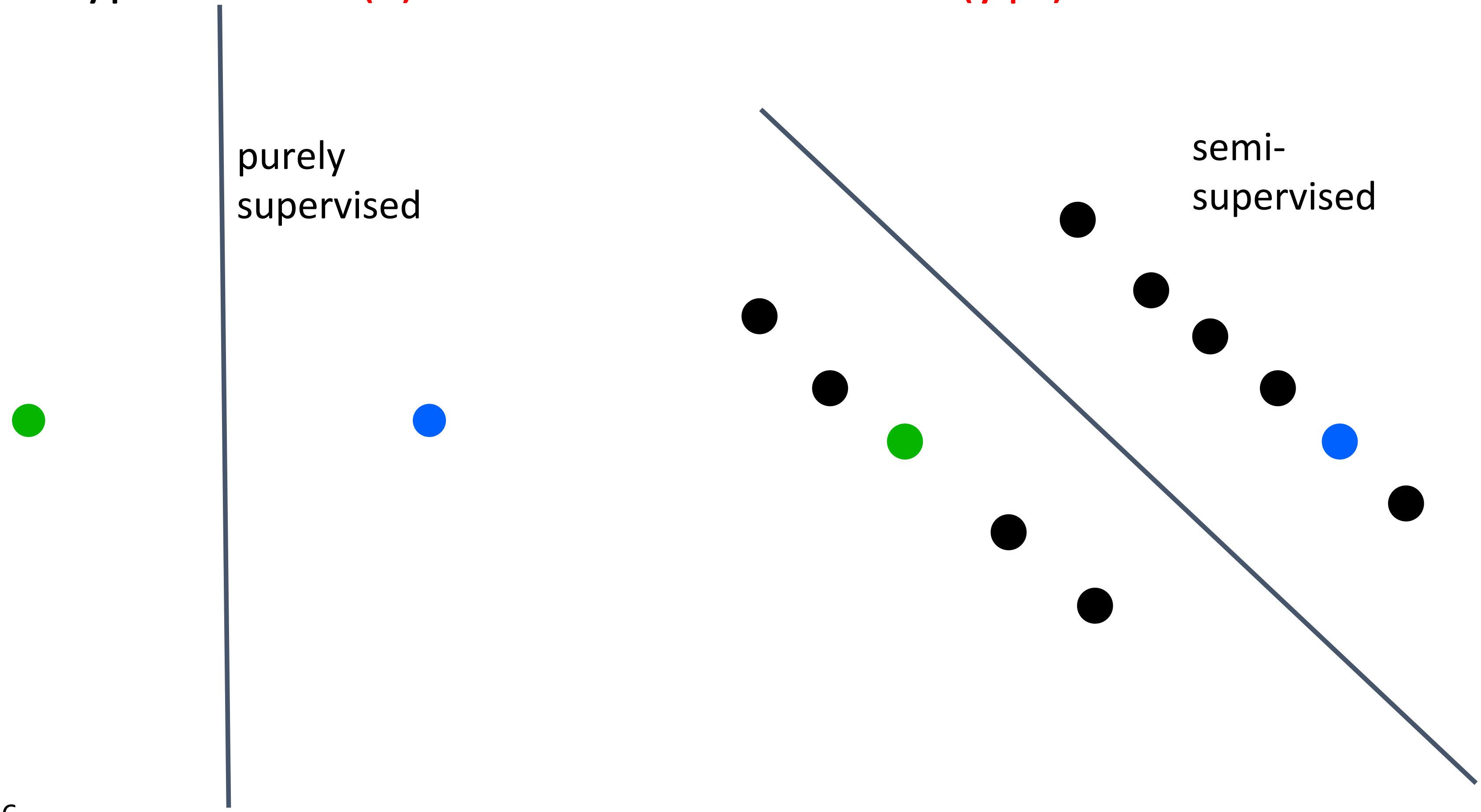
- Previous learning from: unlabeled data

  + labels for other tasks

- **Prior: shared underlying explanatory factors, in particular between P(x) and P(Y|x)**

Sharing Statistical Strength by Semi-Supervised Learning

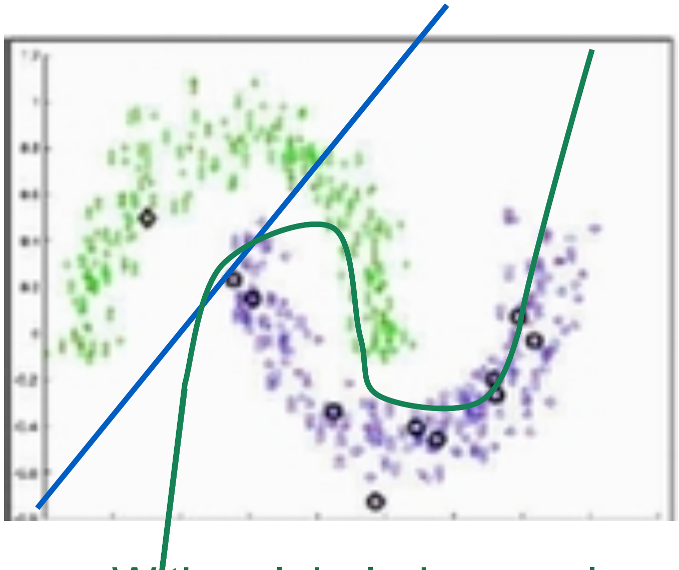
- Hypothesis: P(x) shares structure with P(y|x)



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

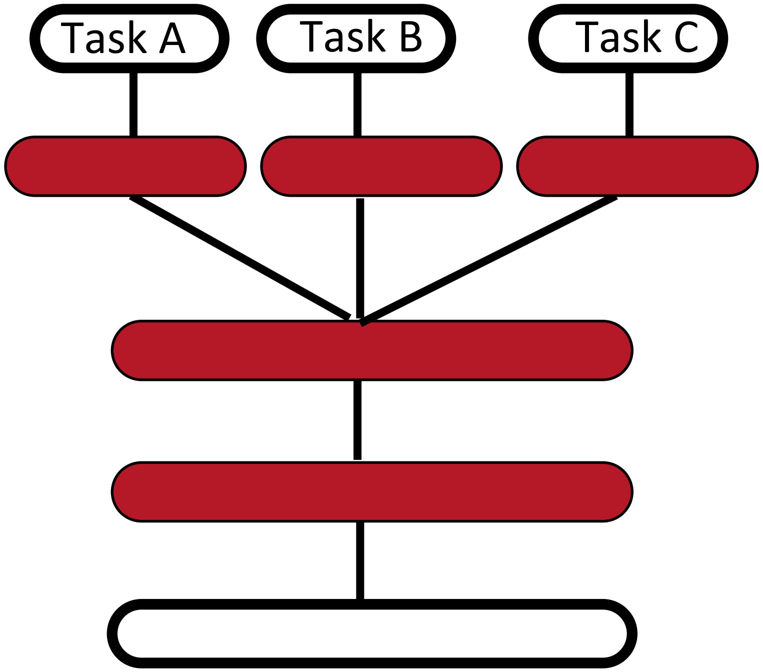
Multi-Task Learning

- 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**

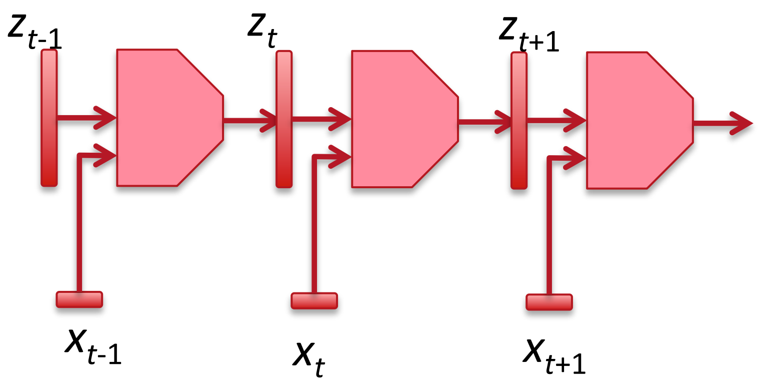


E.g. dictionary, with intermediate concepts re-used across many definitions

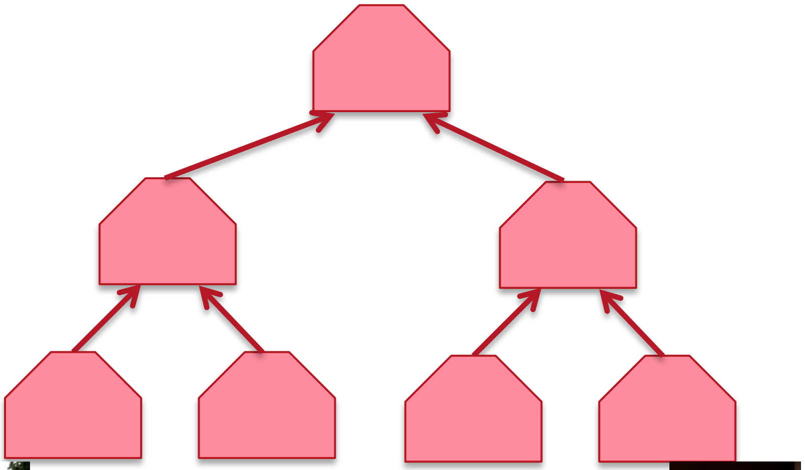
Prior: shared underlying explanatory factors between tasks

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

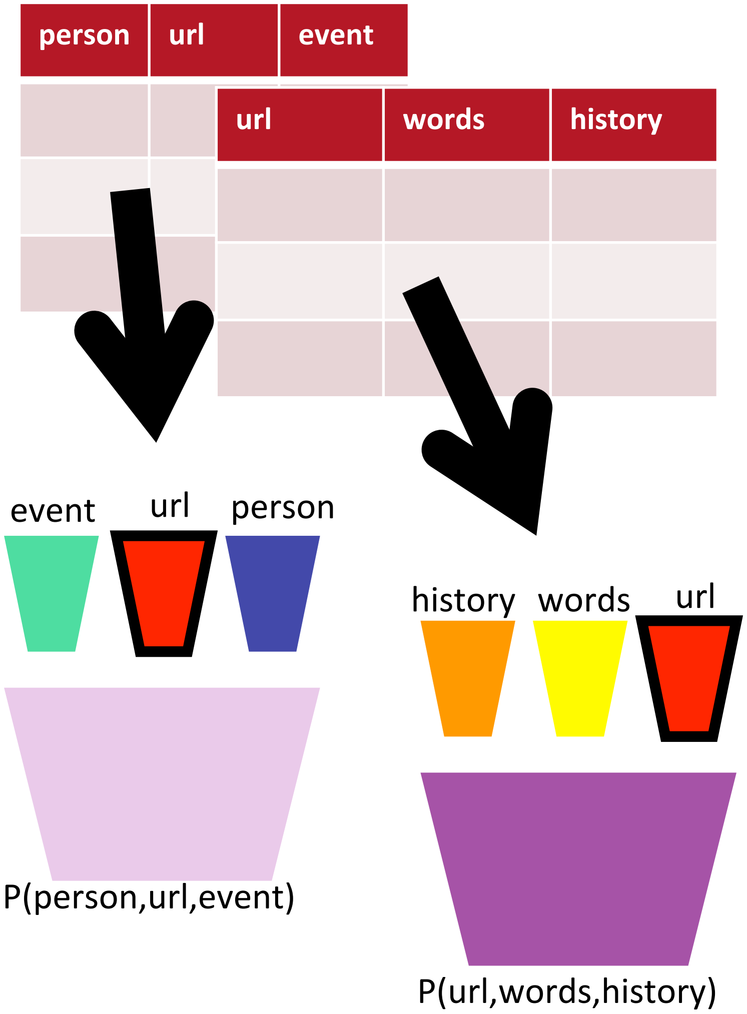


- Result after unfolding = deep computation / representation

(Bottou 2011, Socher et al 2011)

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**



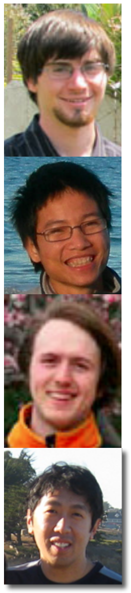
Invariance and Disentangling

- Invariant features

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

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)

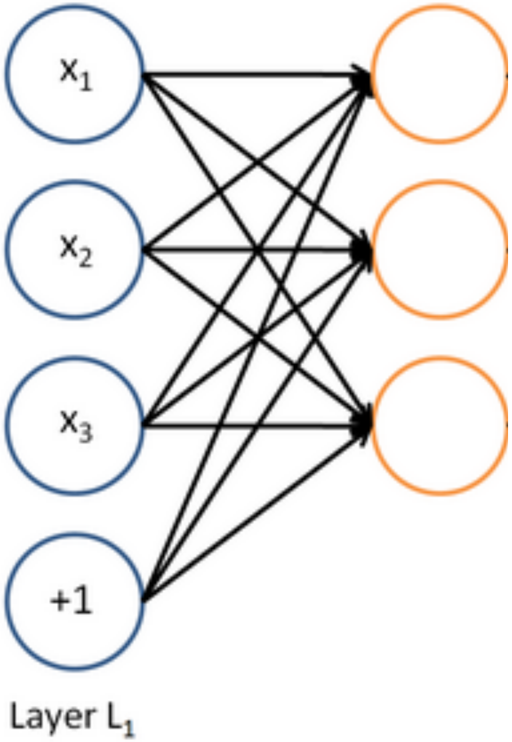
WHY?

Part 2

Representation Learning Algorithms

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

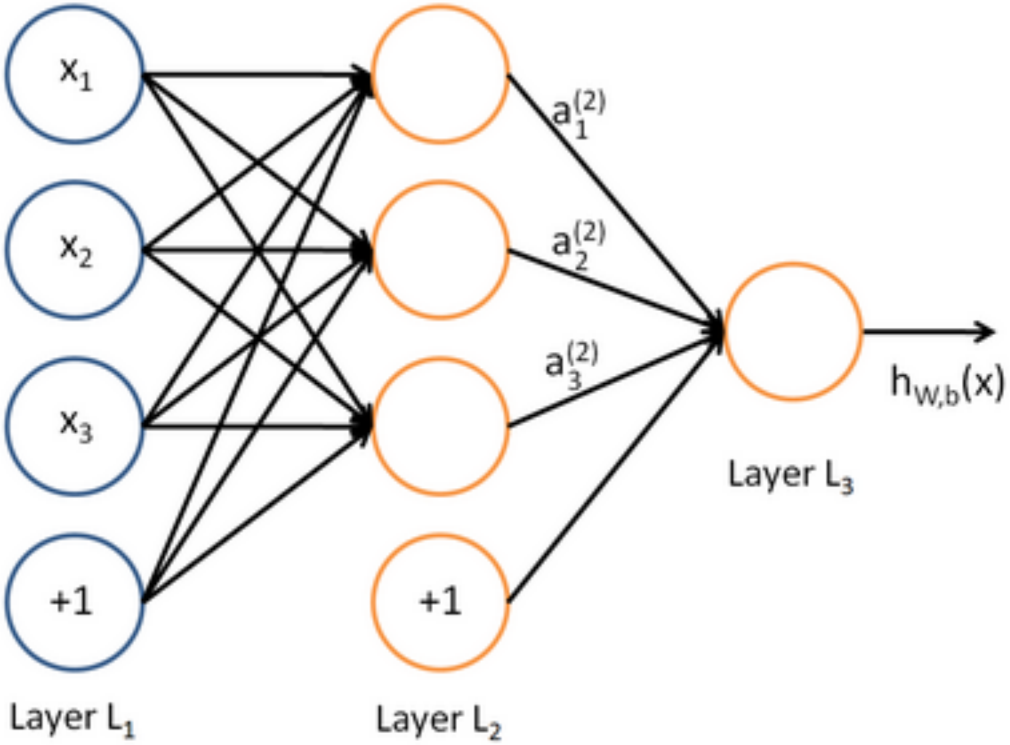
If we feed a vector of inputs through a bunch of logistic regression functions, then we get a vector of outputs



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

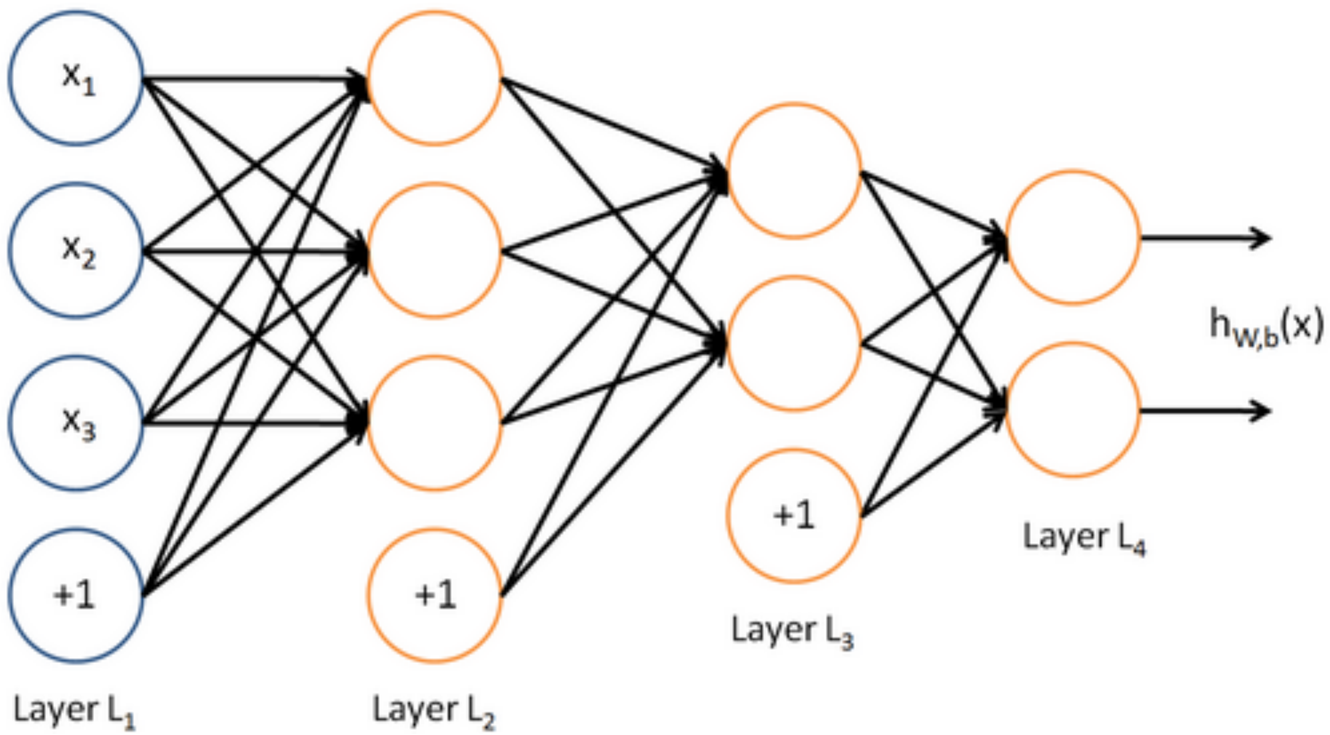
… which we can feed into another logistic regression function



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

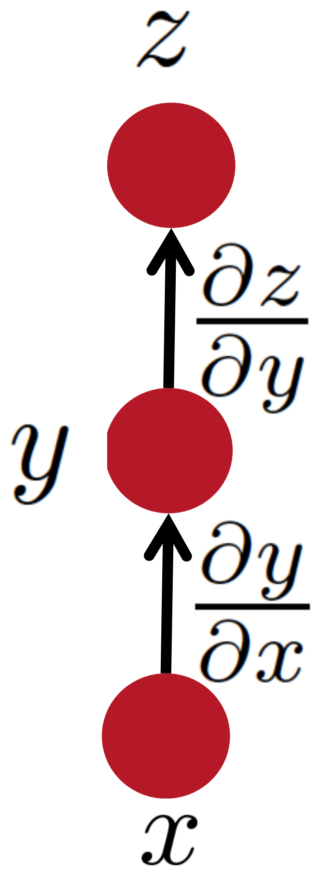
- Compute gradient of example-wise loss wrt parameters

- 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


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

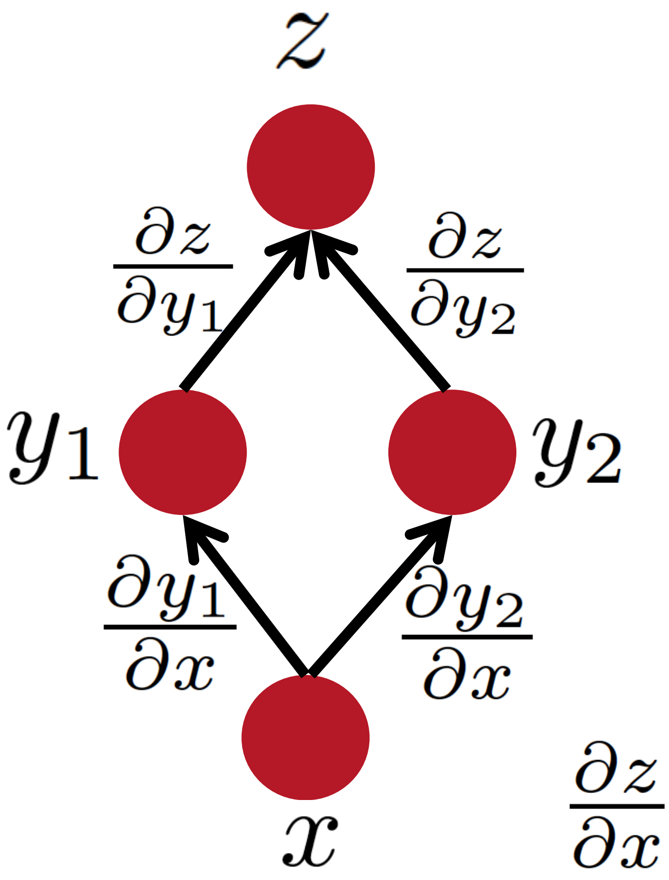
$$\Delta z = \frac{\partial z}{\partial y} \Delta y$$

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

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

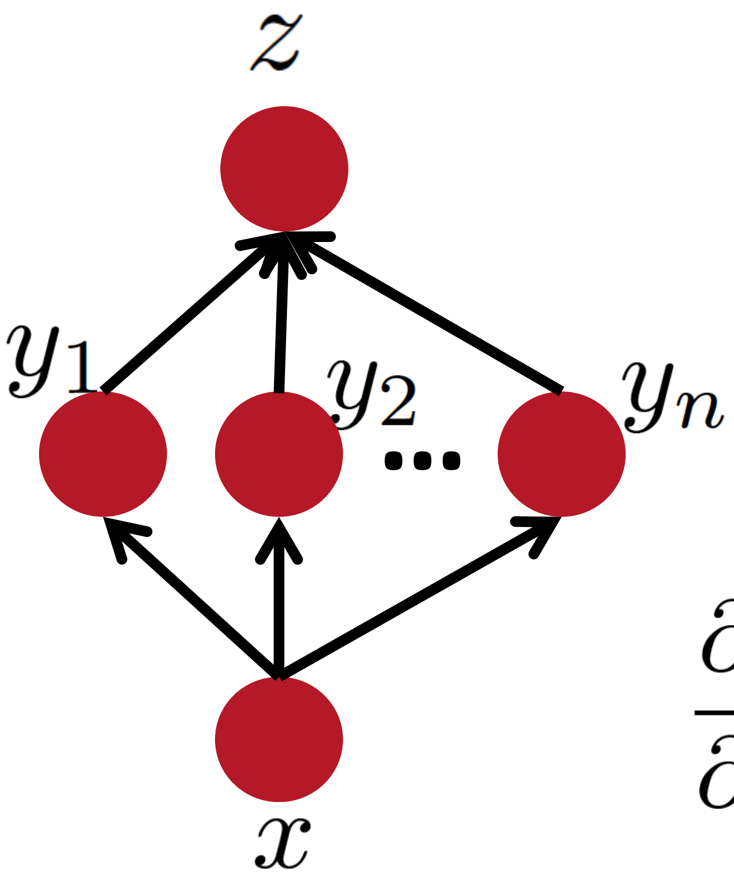
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}$$

Multiple Paths Chain Rule


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

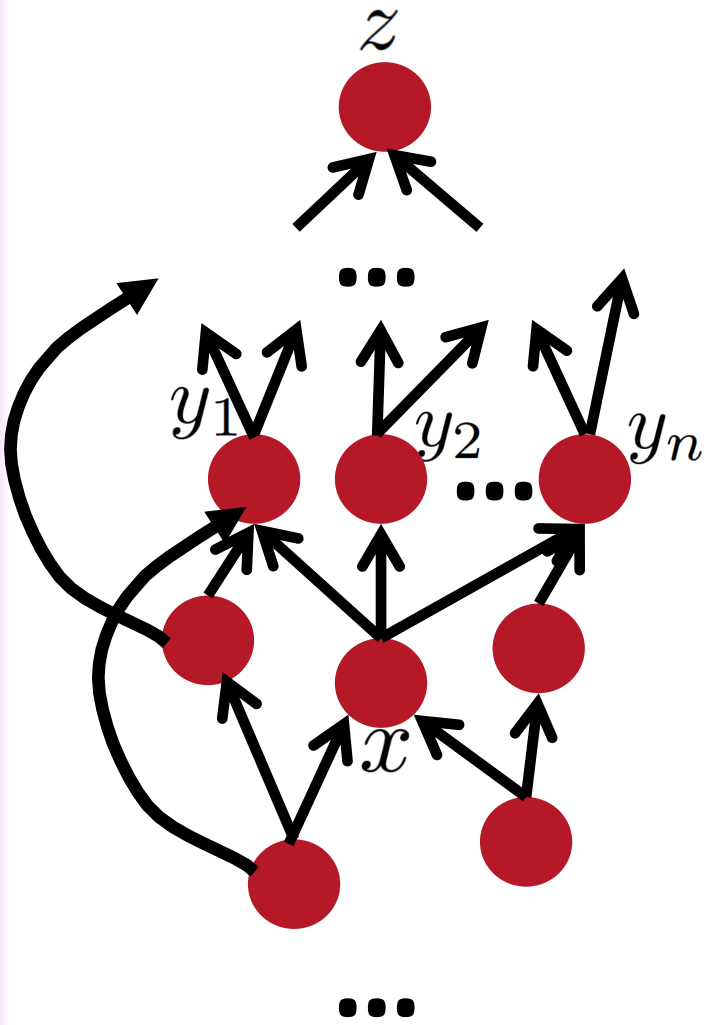
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial x}$$

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

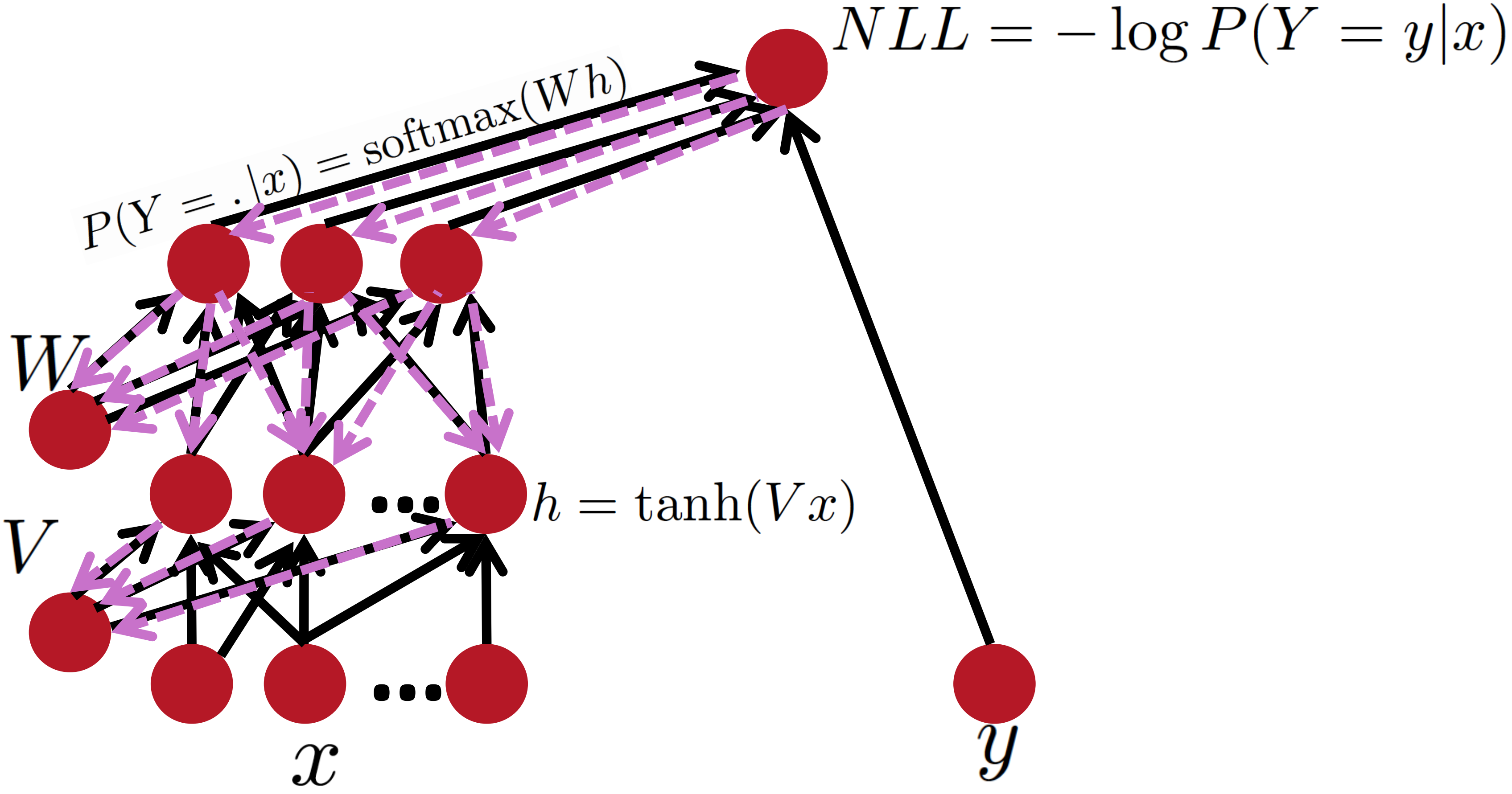
node = computation result

arc = computation dependency

$\{y_1,\ y_2,\ \ldots\ y_n\}$ = 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



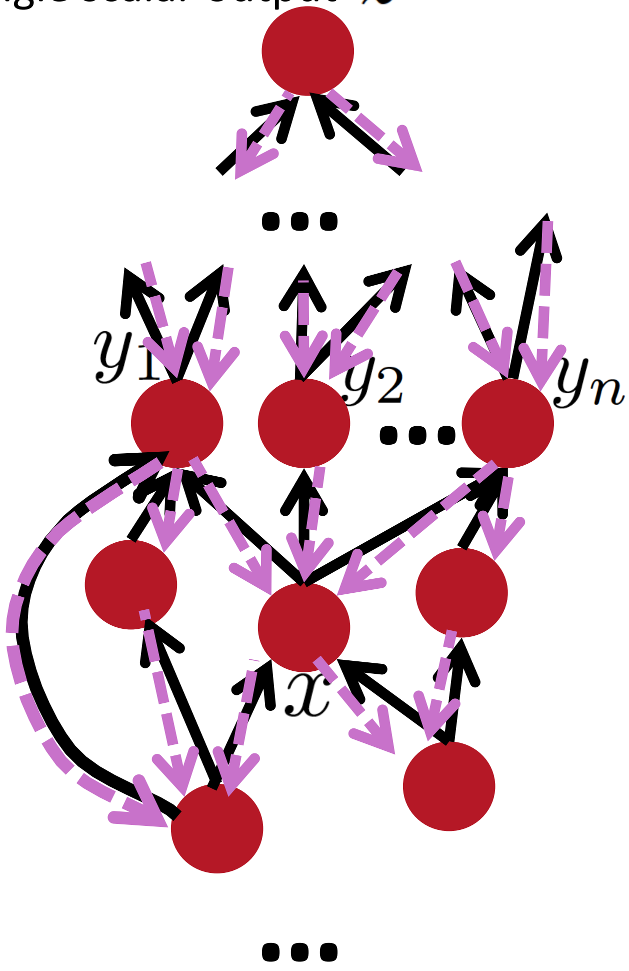
$NLL = -\log P(Y = y|x)$

$\mathrm{P}(\mathrm{Y} = .|x) = \mathrm{softmax}(\mathrm{W}\,\mathrm{h})$

$h = \tanh(Vx)$

Back-Prop in General Flow Graph

Single scalar output z



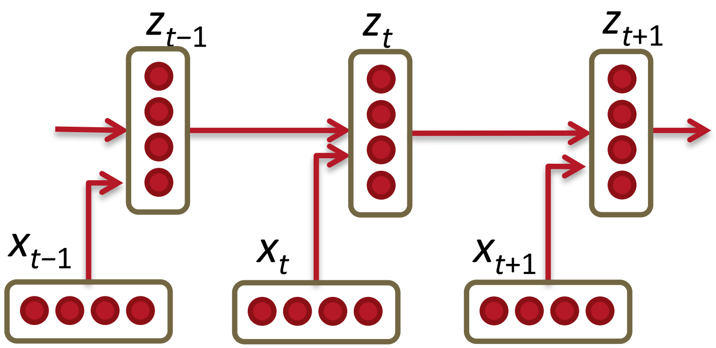
1. Fprop: visit nodes in topo-sort order
   - Compute value of node given predecessors
2. Bprop:
   - initialize output gradient = 1
   - visit nodes in reverse order:
     Compute gradient wrt each node using gradient wrt successors

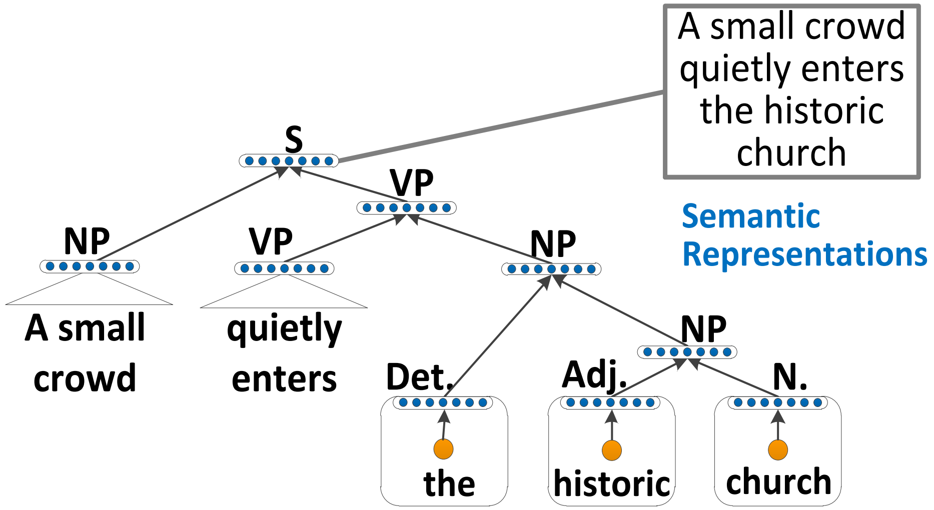
$\{y_1,\ y_2,\ \ldots\ y_n\}$ = successors of x

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

Back-Prop in Recurrent & Recursive Nets

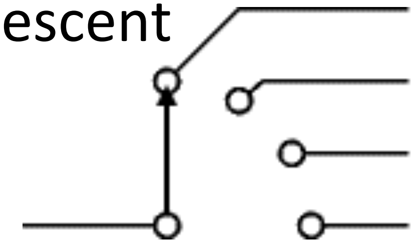
- Replicate a parameterized function over different time steps or nodes of a DAG
- Output state at one time-step / node is used as input for another time-step / node



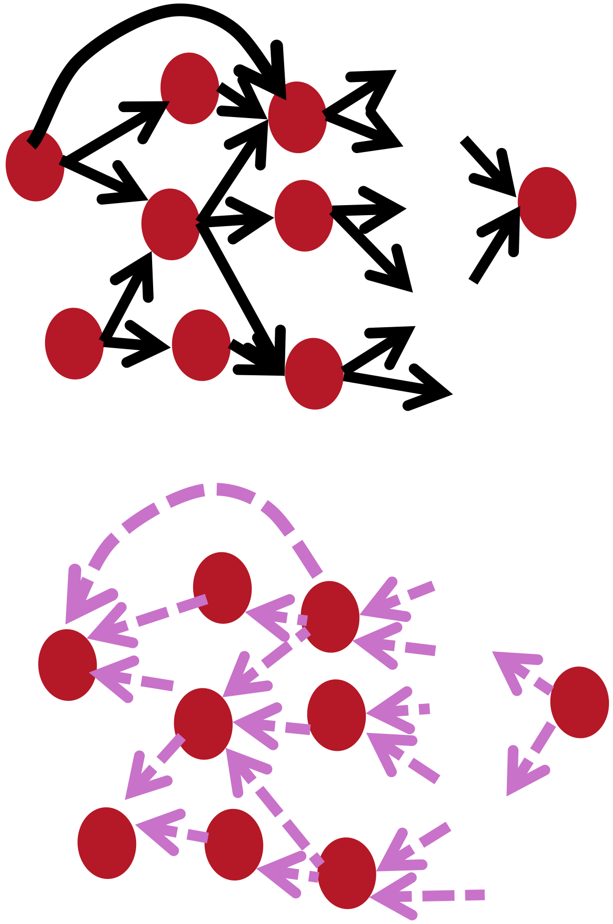


Backpropagation Through Structure

- 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
  - Max(a,b) is continuous.



Automatic Differentiation



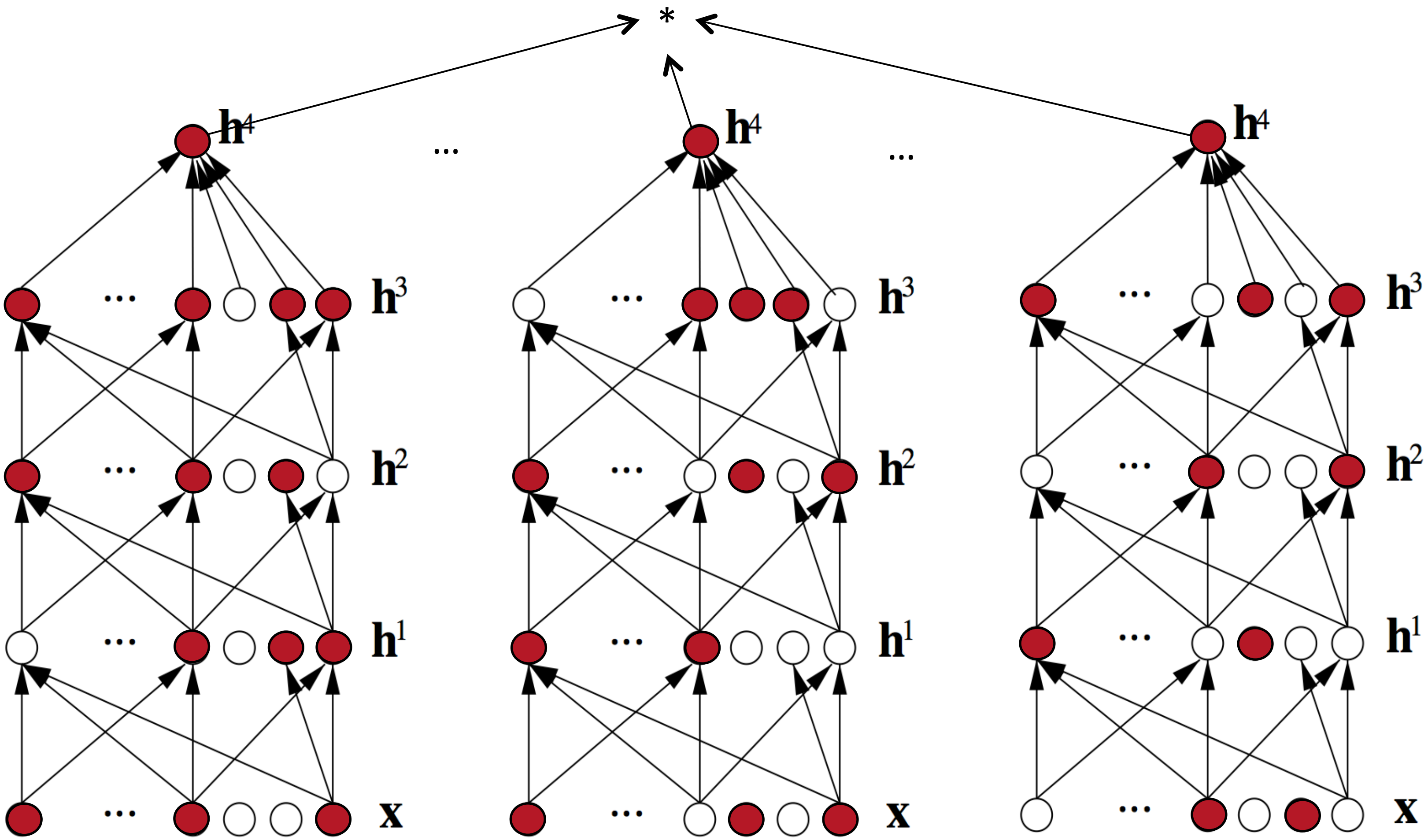
- The gradient computation can be automatically inferred from the symbolic expression of the fprop.
- Each node type needs to know how to compute its output and how to compute the gradient wrt its inputs given the gradient wrt its output.
- Easy and fast prototyping

theano

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

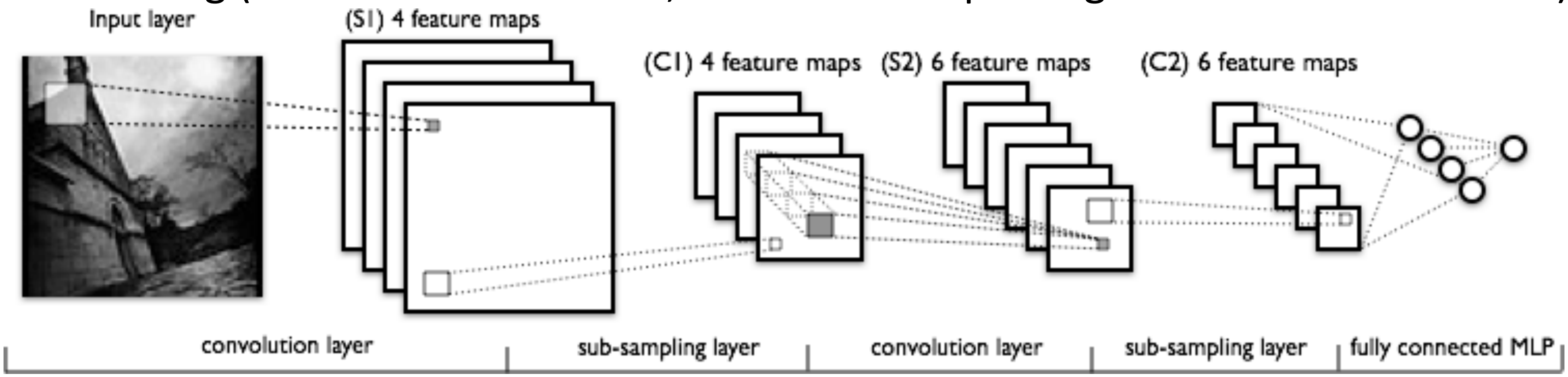
- **Dropouts** trick: during training multiply neuron output by random bit (p=0.5), during test by 0.5
- Used in deep supervised networks
- Similar to denoising auto-encoder, but corrupting every layer
- Works better with some non-linearities (rectifiers, maxout) (Goodfellow et al. ICML 2013)
- Equivalent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

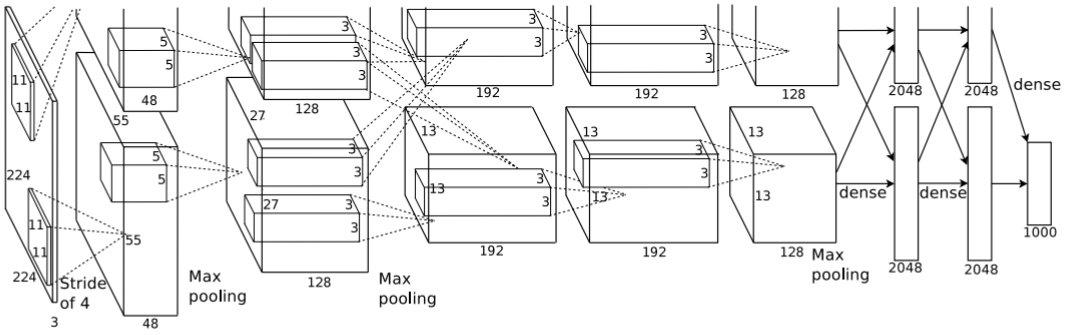
Dropout Regularizer: Super-Efficient Bagging

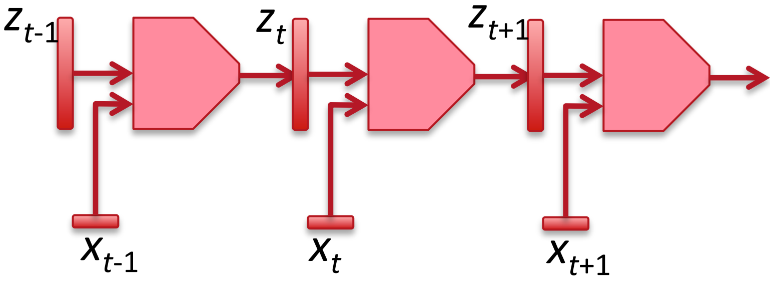


Temporal & Spatial Inputs: Convolutional & Recurrent Nets

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





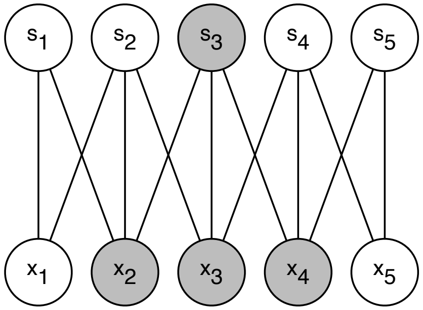


Recurrent nets (RNNs) can summarize information from the past

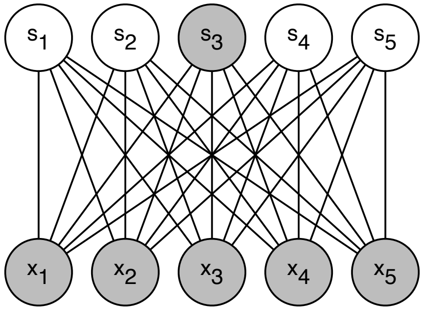
Bidirectional RNNs also summarize information from the future

Convolution = sparse connectivity + parameter sharing

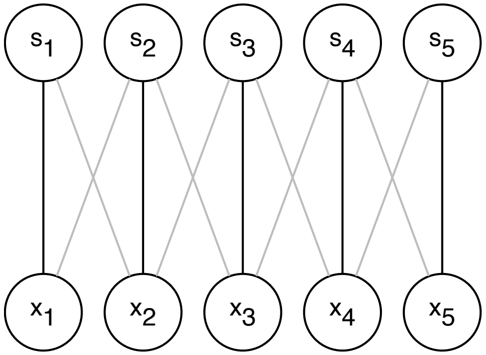
$$s[t] = (x * w)(t) = \sum_{a=-\infty}^{\infty} x[a]w[t-a]$$



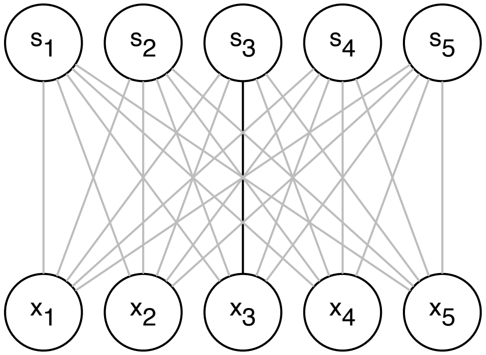
sparse



dense



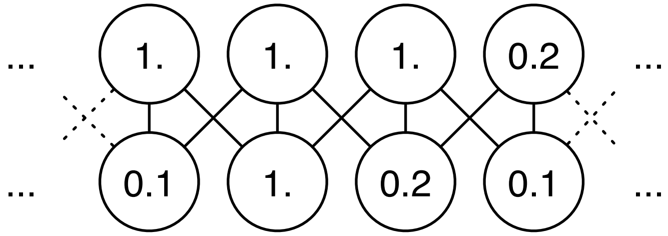
shared



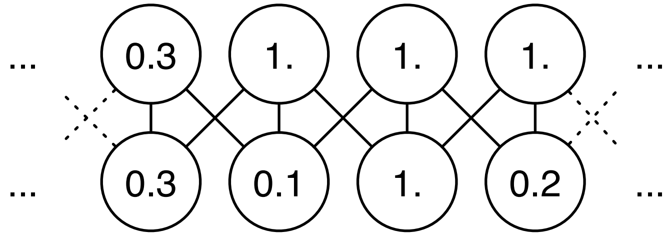
not shared

Pooling Layers

- Aggregate to achieve local invariance

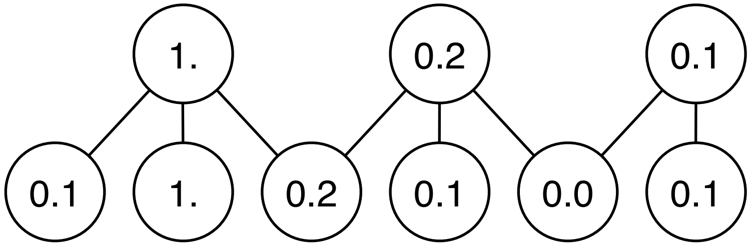


Max-pooling

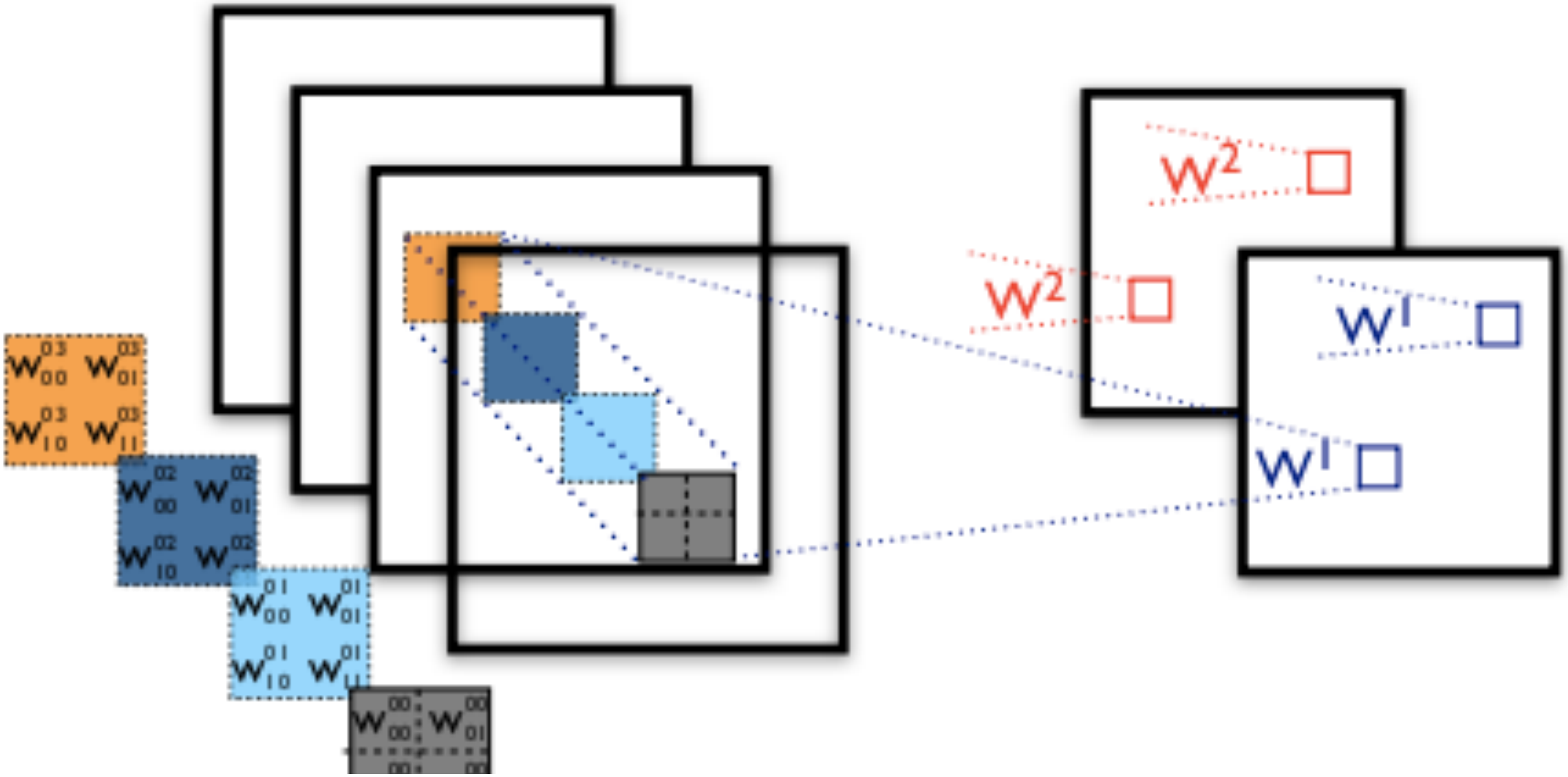


Effect of translation

- Subsampling to reduce temporal/spatial scale and computation

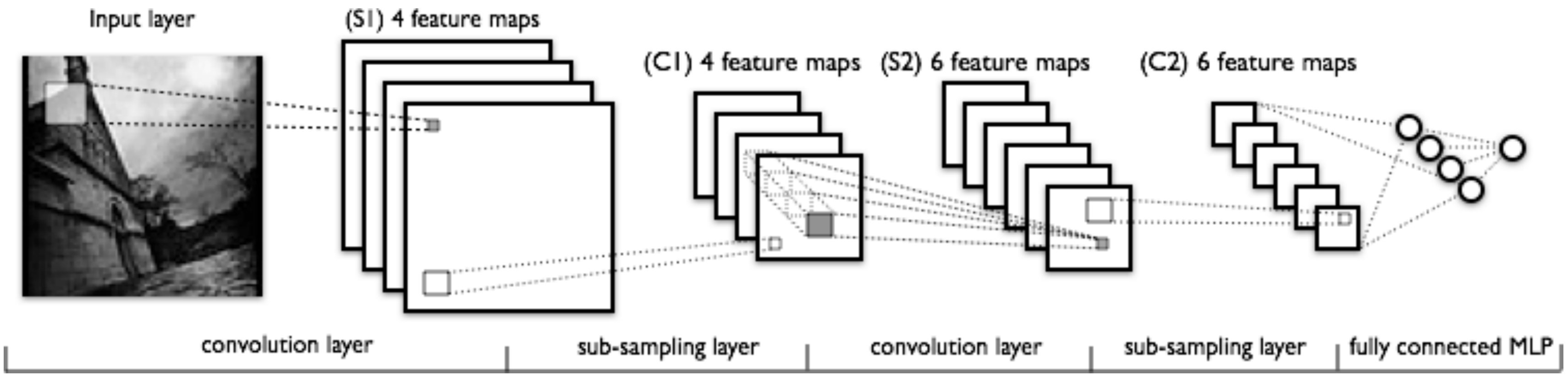


Multiple Convolutions: Feature Maps



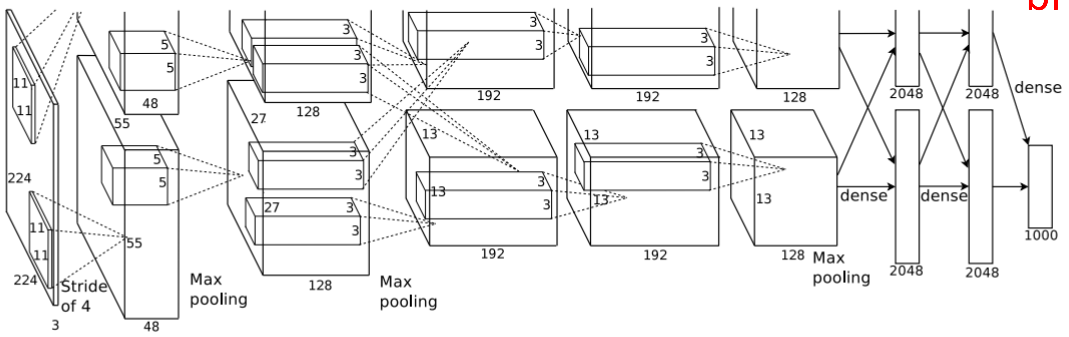
Alternating convolutions & pooling

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



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

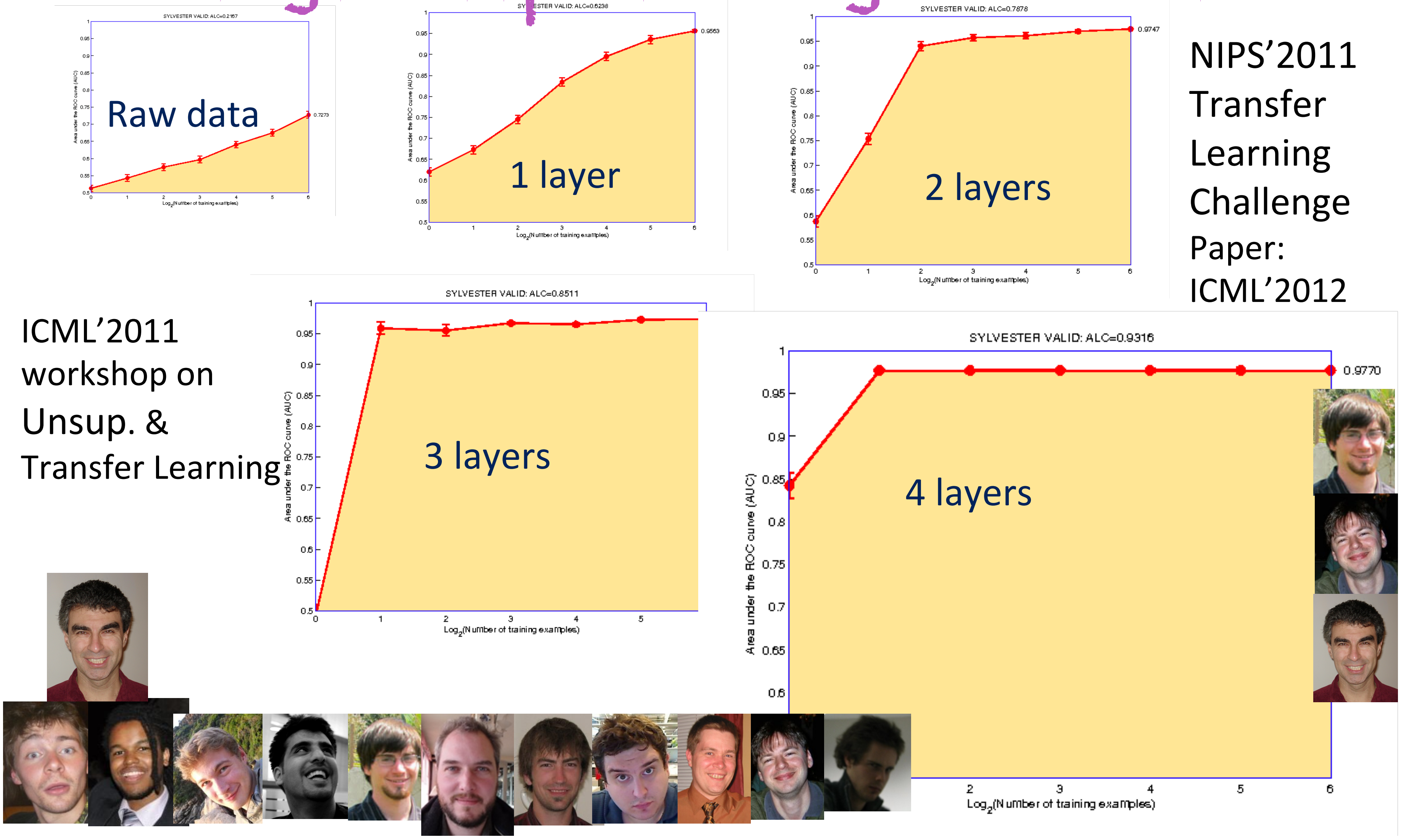
Krizhevsky, Sutskever & Hinton 2012
breakthrough in object recognition



Distributed Representations & Neural Nets:

How to do unsupervised training?

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

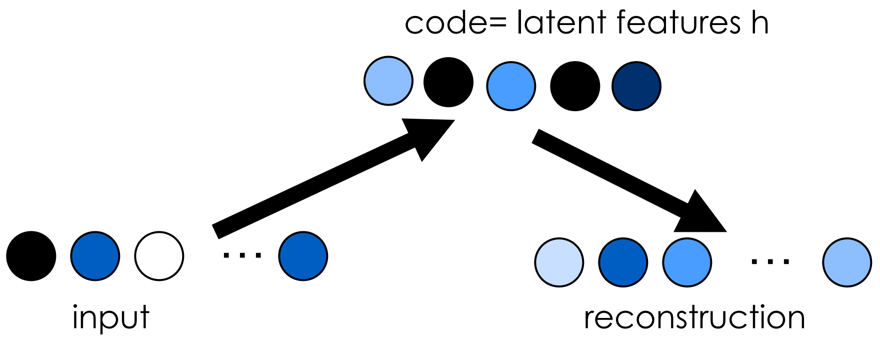


NIPS'2011 Transfer Learning Challenge Paper: ICML'2012

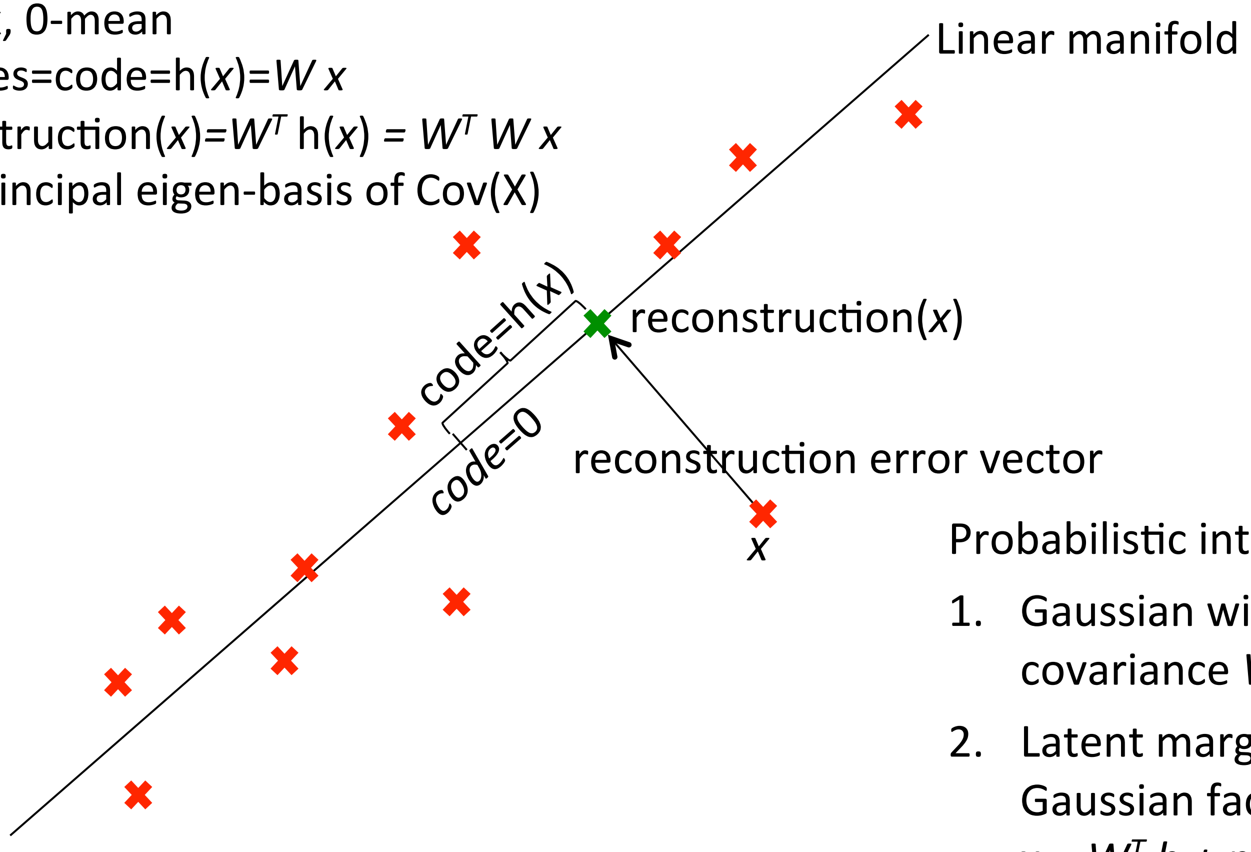
ICML'2011 workshop on Unsup. & Transfer Learning

PCA

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



input x, 0-mean
features=code=h(x)=$W\,x$
reconstruction(x)=W^T h(x) = $W^T\,W\,x$
W = principal eigen-basis of Cov(X)



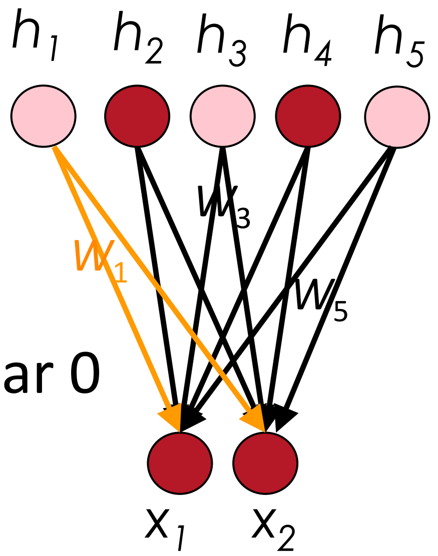
Probabilistic interpretations:

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

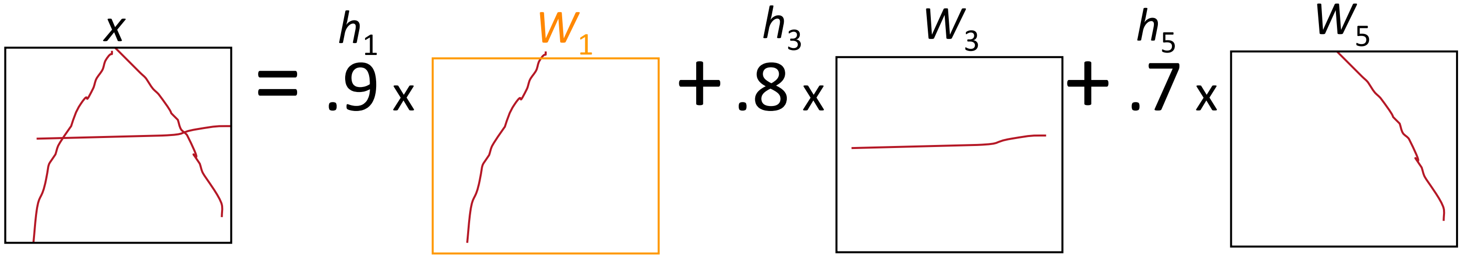
Directed Factor Models: $P(x,h)=P(h)P(x|h)$

factors prior likelihood

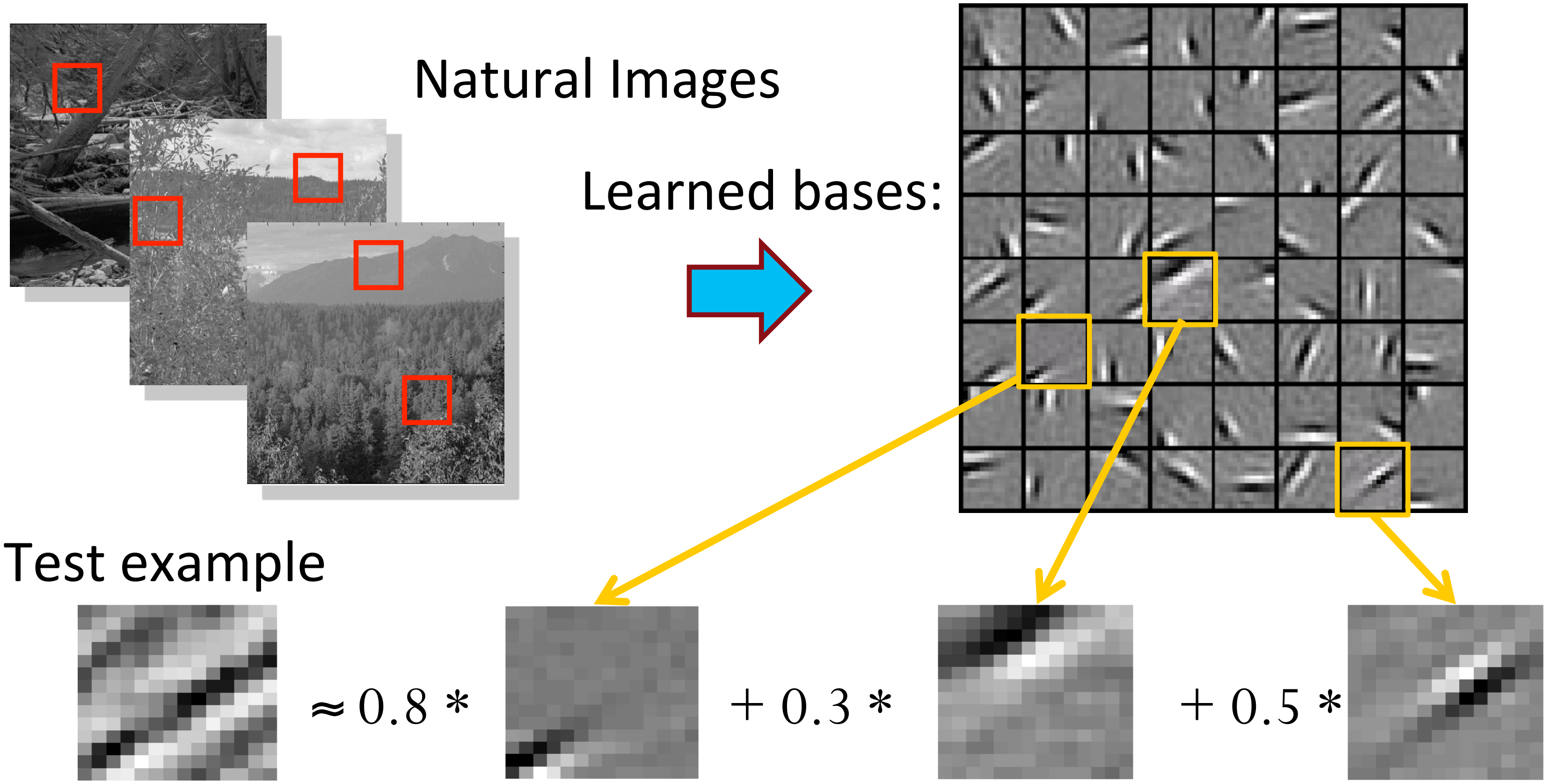
- $P(h)$ factorizes into $P(h_1)\,P(h_2)$…
- Different priors:
  - PCA: $P(h_i)$ is Gaussian
  - ICA: $P(h_i)$ is non-parametric
  - **Sparse coding**: $P(h_i)$ is concentrated near 0
- Likelihood is typically Gaussian $x \mid h$ with mean given by $W^T h$



- 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

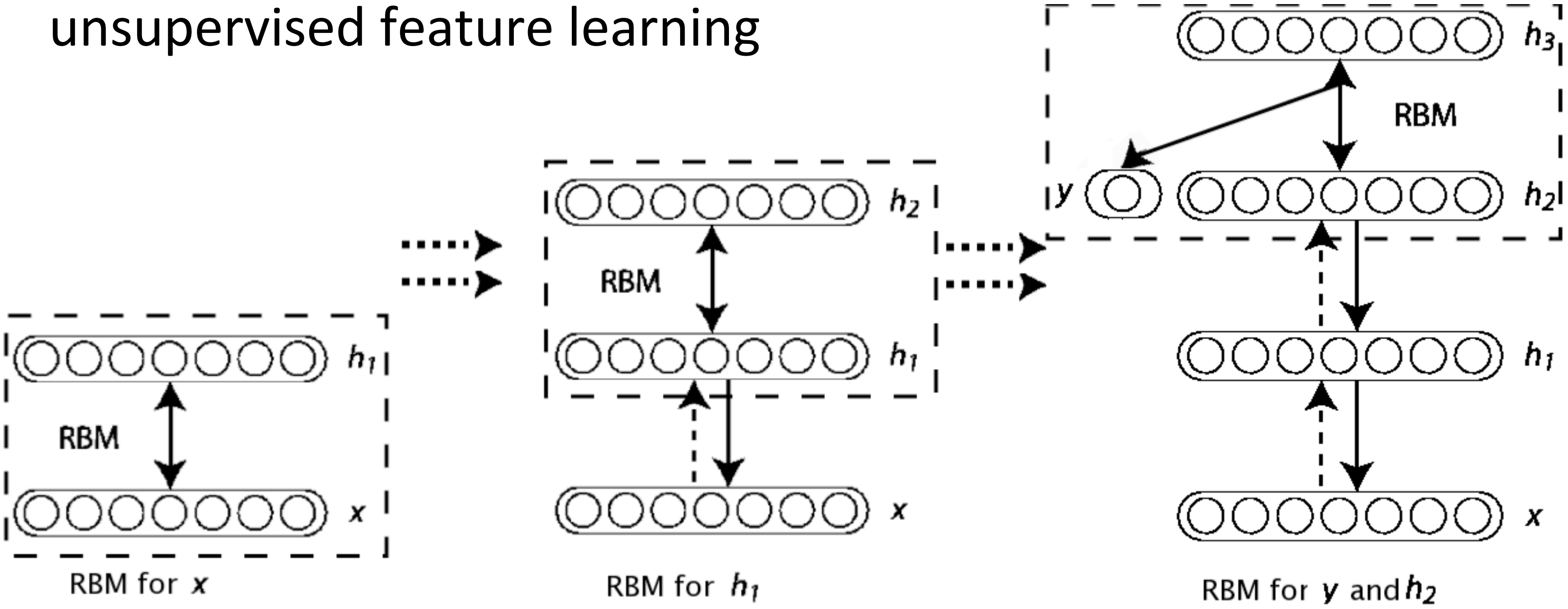
$\approx 0.8 * \quad + 0.3 * \quad + 0.5 *$

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

(feature representation)

Stacking Single-Layer Learners

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



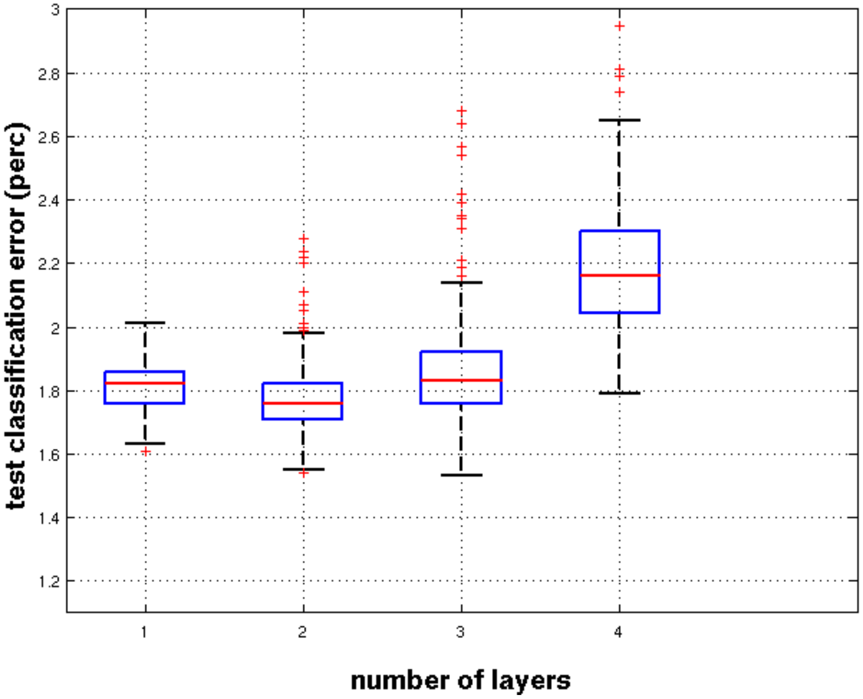
Stacking Restricted Boltzmann Machines (RBM) → Deep Belief Network (DBN)

Effective deep learning first became possible with unsupervised pre-training

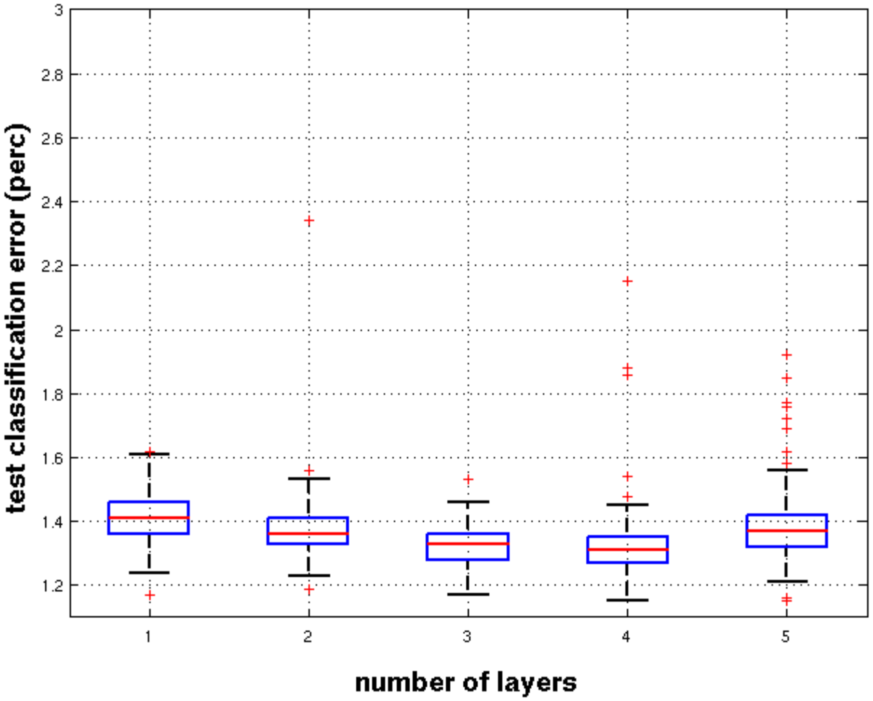
[Erhan et al., JMLR 2010]

(with RBMs and Denoising Auto-Encoders)

Purely supervised neural net



With unsupervised pre-training



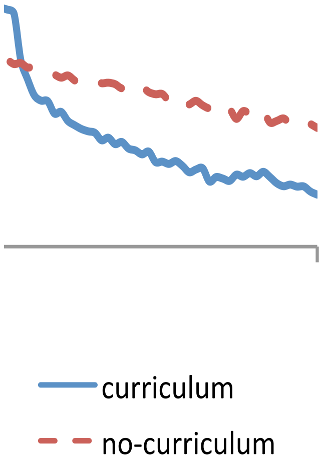
Optimizing Deep Non-Linear Composition of Functions Seems Hard

- Failure of training deep supervised nets before 2006
- Regularization effect + optimization effect of unsupervised pre-training
- Is optimization difficulty due to
  - ill-conditioning?
  - local minima?
  - something else?
- The jury is still out, but we now have success stories of training deep supervised nets without unsupervised pre-training

Order & Selection of Examples Matters

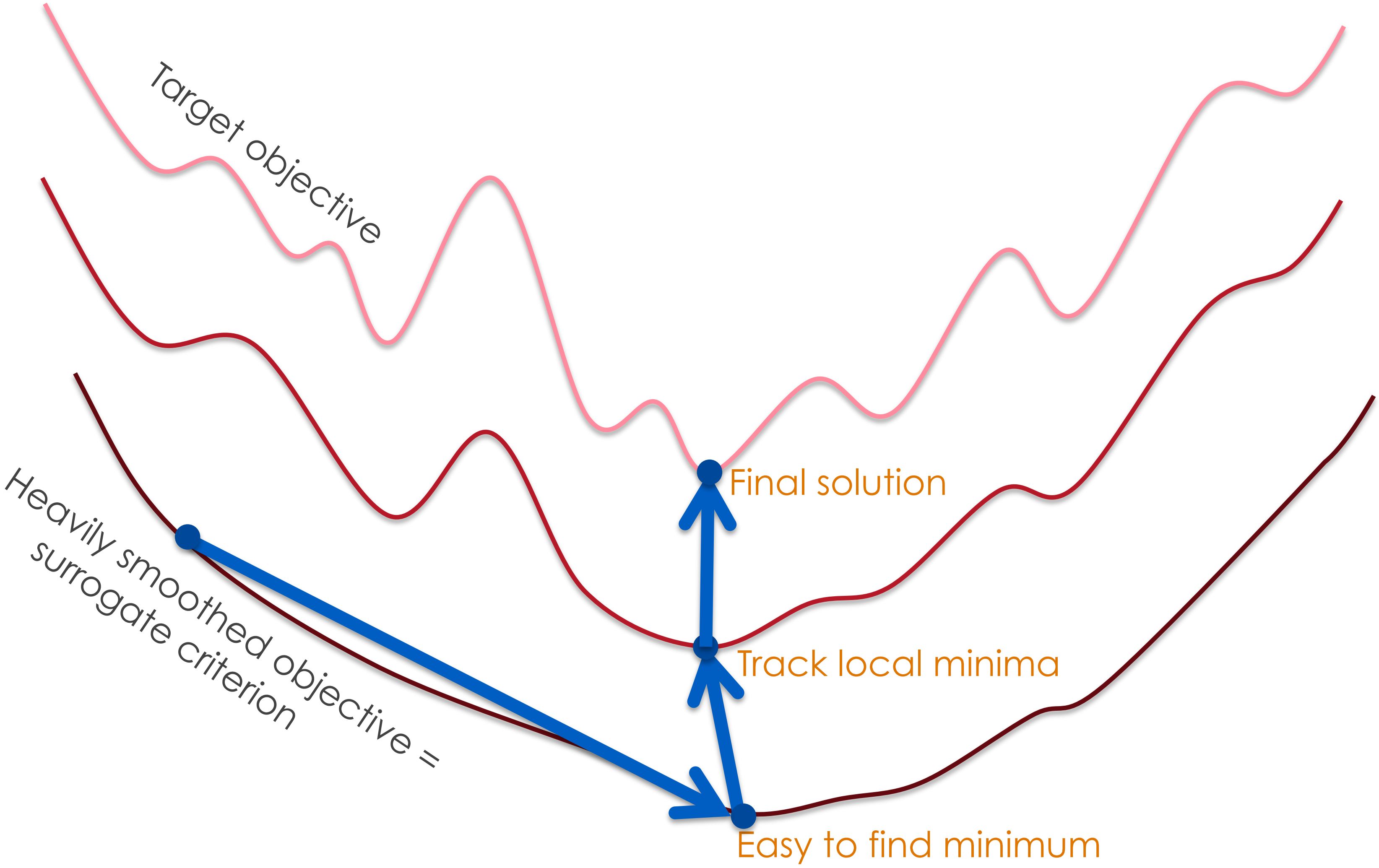
(Bengio, Louradour, Collobert & Weston, ICML'2009)

- Curriculum learning
- (Bengio et al 2009, Krueger & Dayan 2009)
- **Start with easier examples**

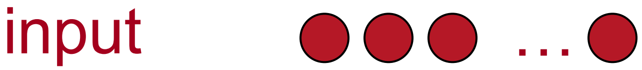


- Faster convergence to a better local minimum in deep architectures

Continuation Methods

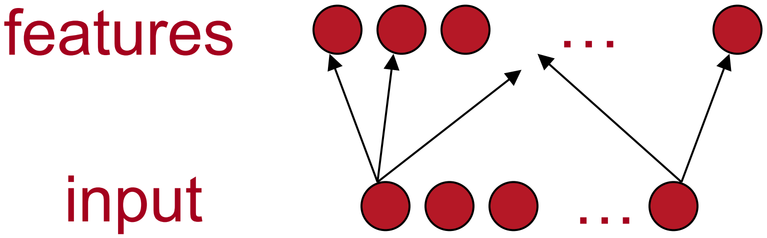


Layer-wise Unsupervised Learning

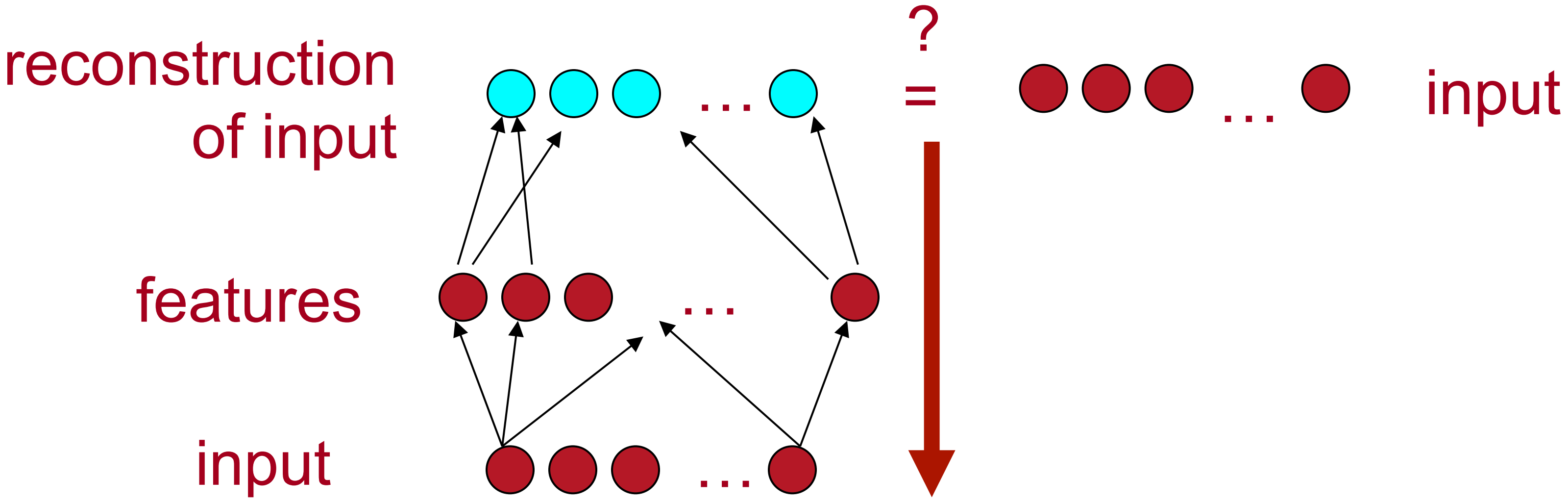


input …

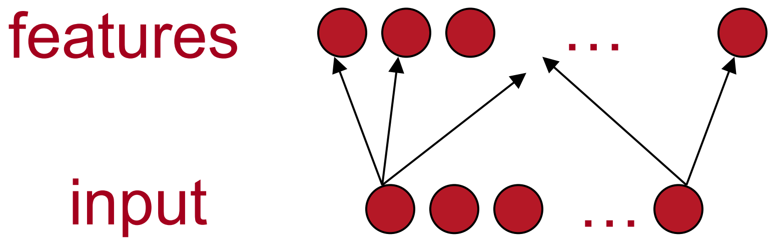
Layer-Wise Unsupervised Pre-training



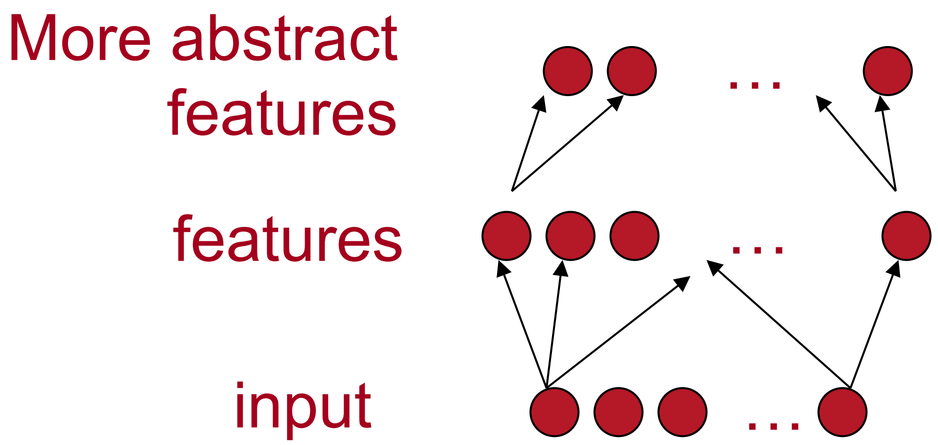
Layer-Wise Unsupervised Pre-training



Layer-Wise Unsupervised Pre-training



Layer-Wise Unsupervised Pre-training

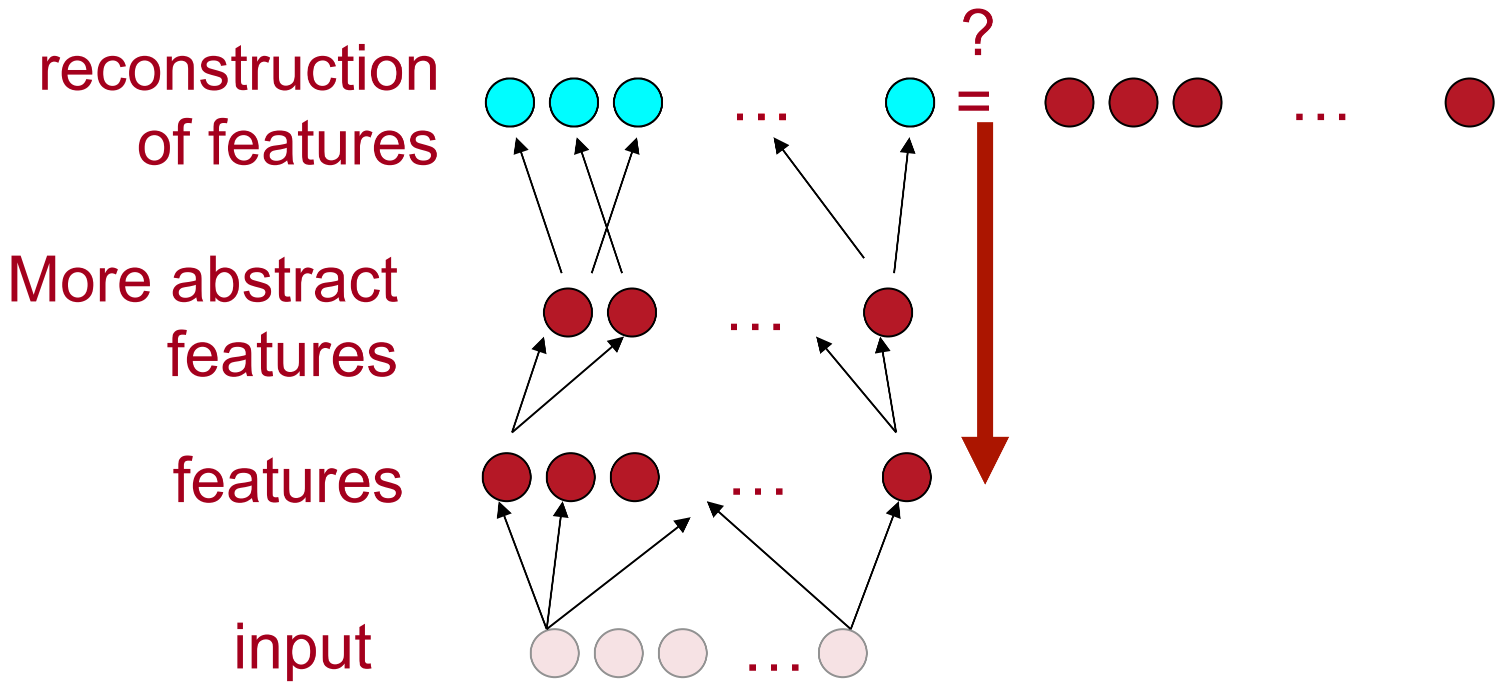


More abstract features

features

input

Layer-wise Unsupervised Learning



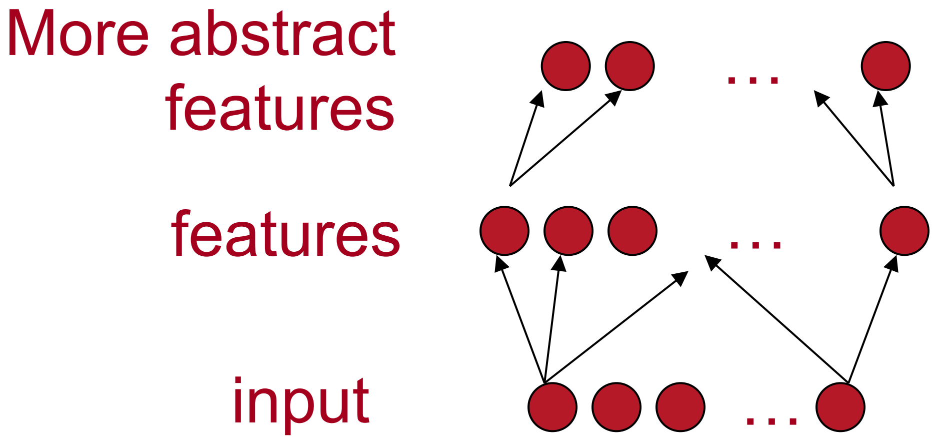
reconstruction of features

More abstract features

features

input

Layer-Wise Unsupervised Pre-training

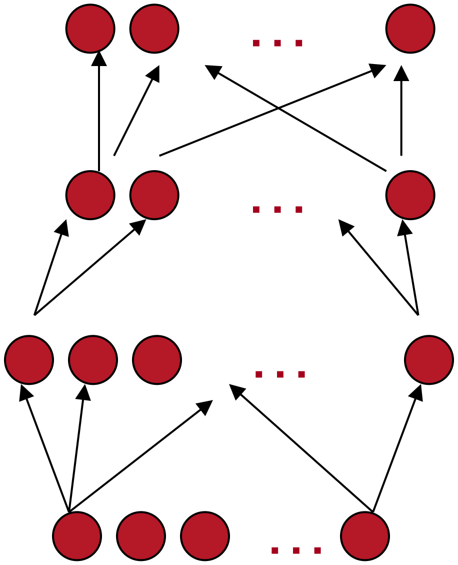


More abstract features

features

input

Layer-wise Unsupervised Learning



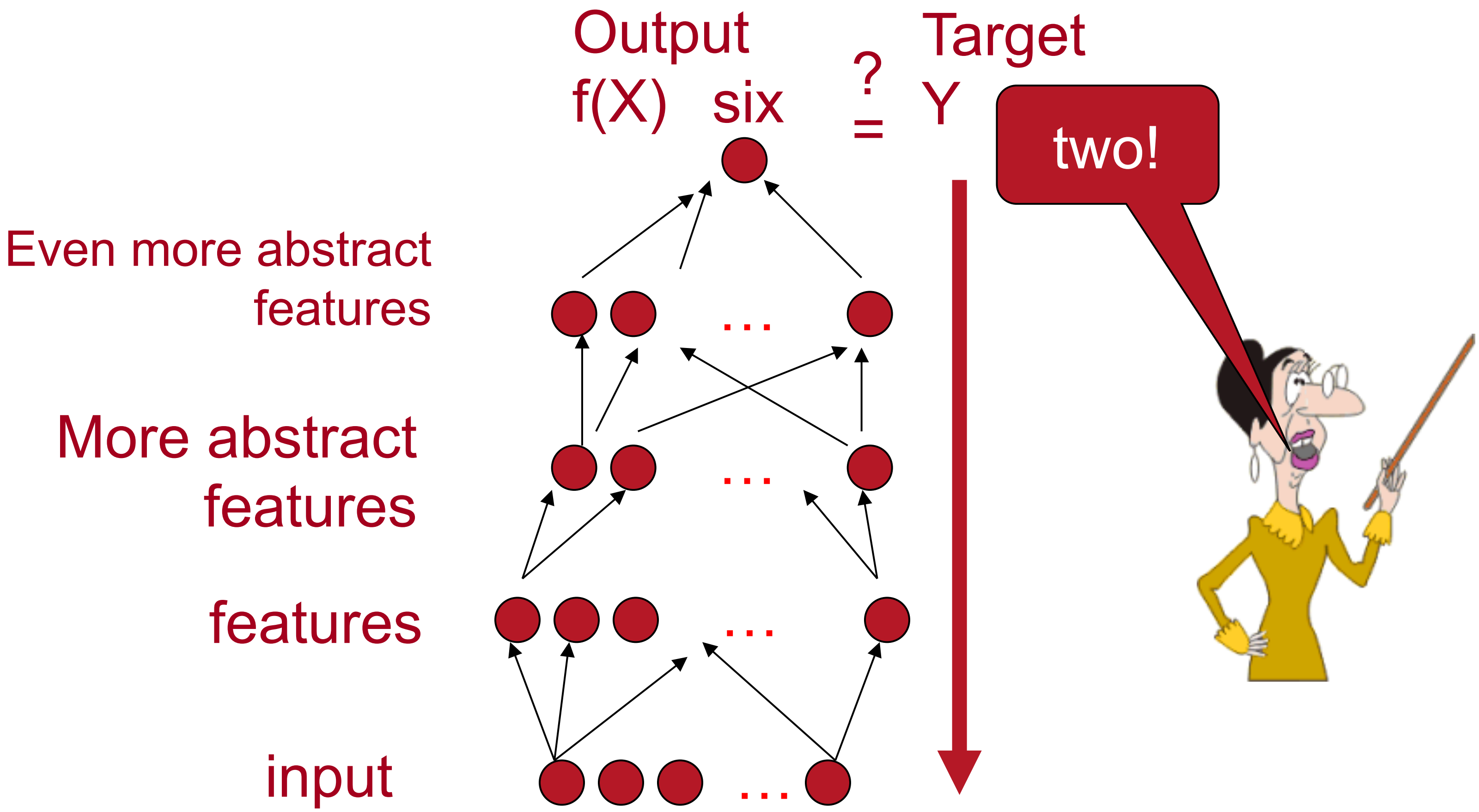
Even more abstract features

More abstract features

features

input

Supervised Fine-Tuning



Output

f(X) six $\overset{?}{=}$ Y

Target

two!

Even more abstract features

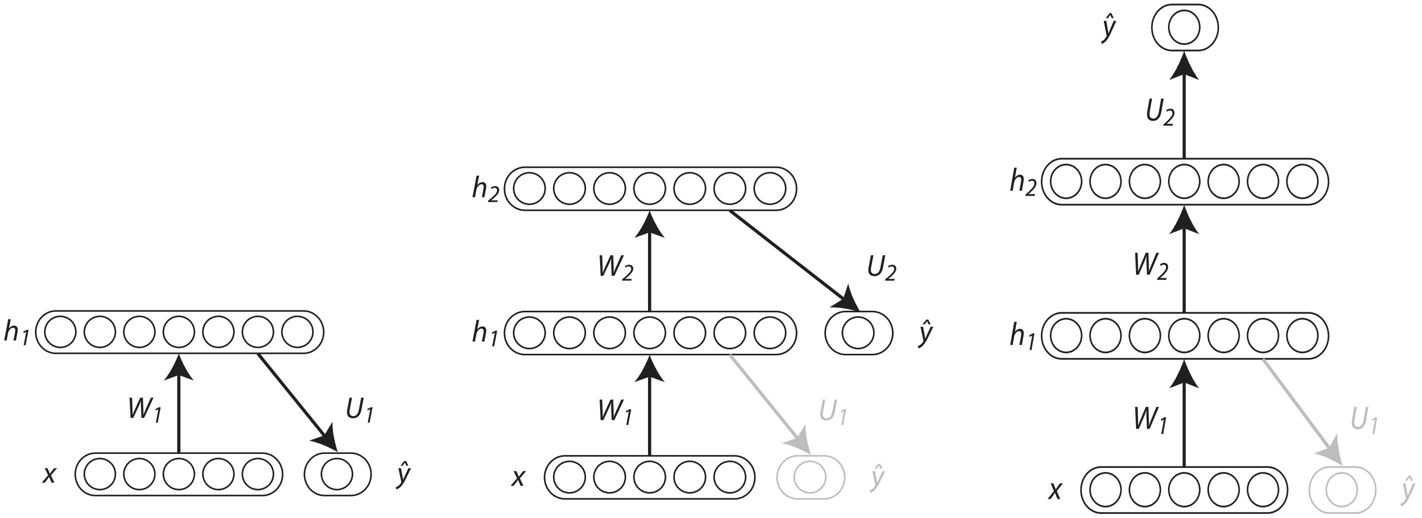
More abstract features

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.

Understanding the difficulty of training deep feedforward supervised neural networks

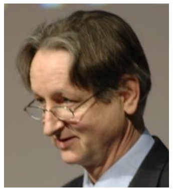
(Glorot & Bengio, AISTATS 2010)

Study the activations and gradients

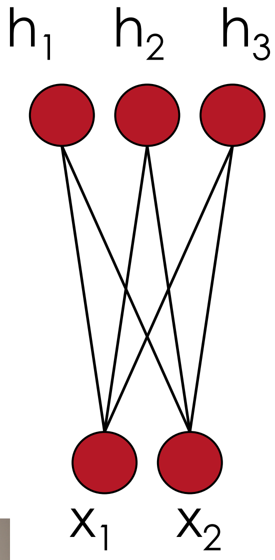
- wrt depth
- as training progresses
- for different initializations → big difference
- for different non-linearities → big difference

First demonstration that deep supervised nets can be successfully trained almost as well as with unsupervised pre-training, by setting up the optimization problem appropriately…

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"*

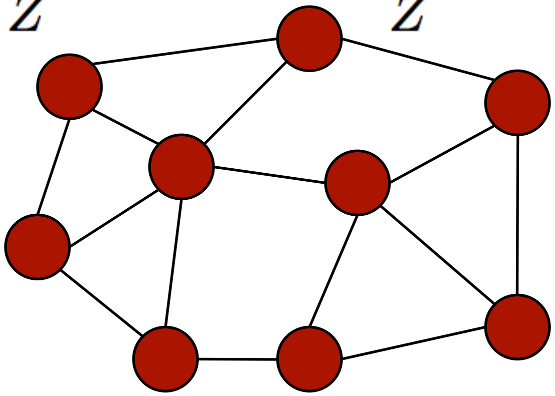
Boltzmann Machines & MRFs

- Boltzmann machines:

(Hinton 84)

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

- Markov Random Fields:



Undirected graphical models

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

Soft constraint / probabilistic statement

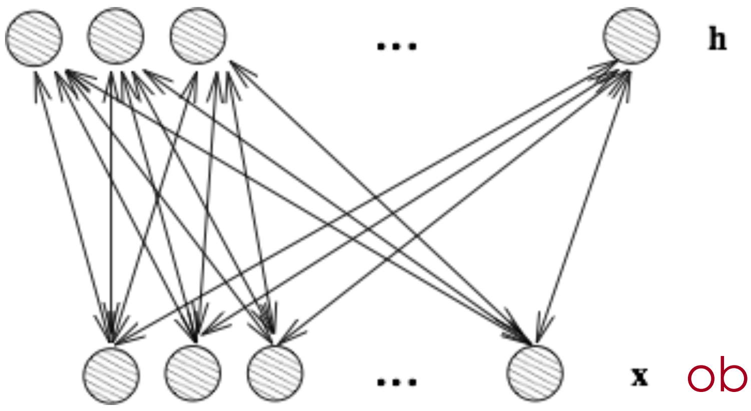
- More interesting with latent variables!

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

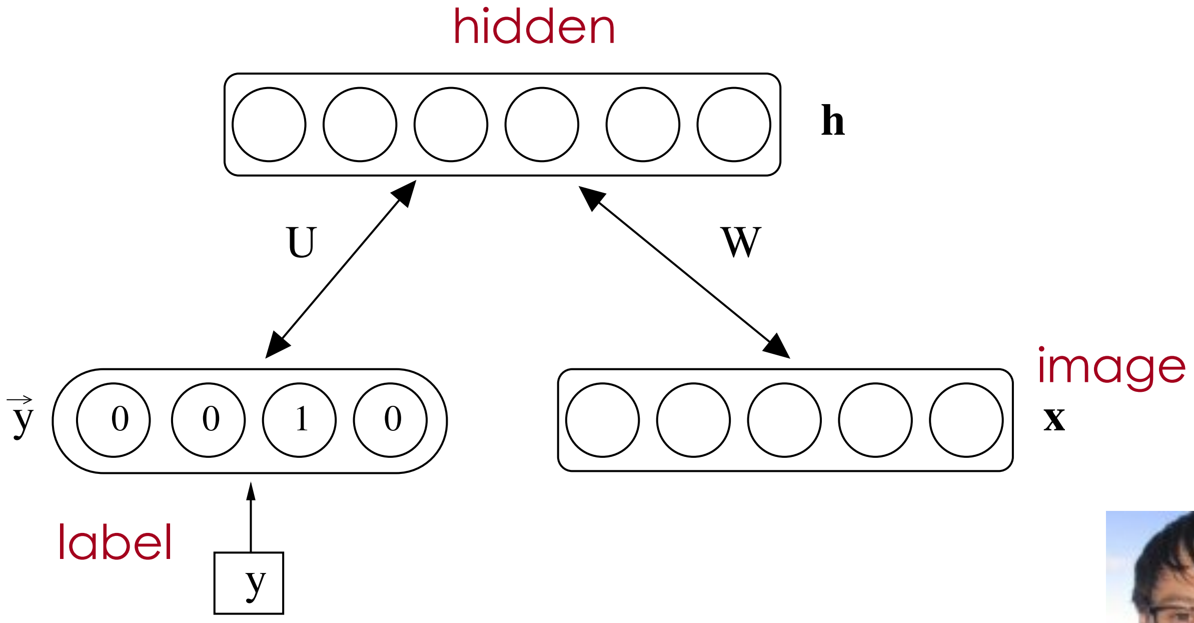
- **Bipartite** undirected graphical model



hidden

observed

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

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

- Gradient has two components:

“positive phase”

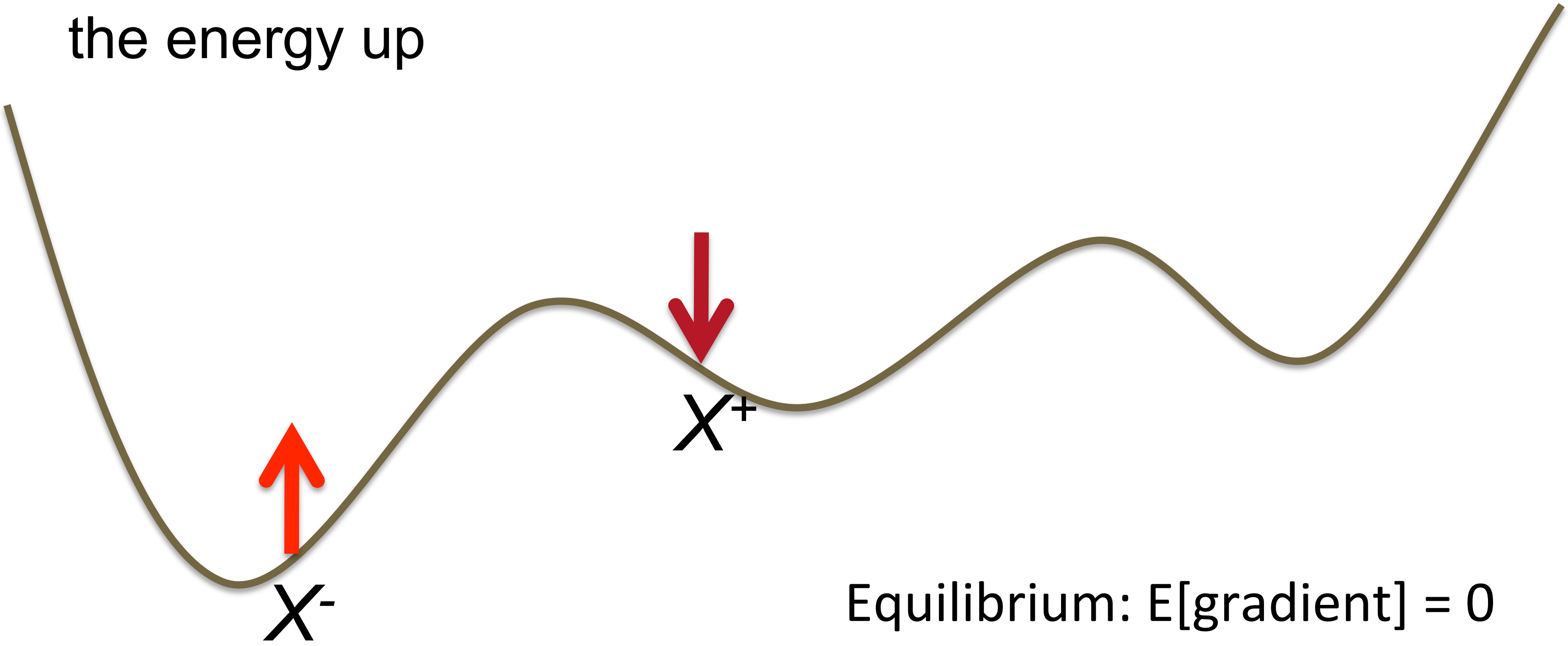
“negative phase”

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

- 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



Equilibrium: E[gradient] = 0

Training RBMs

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

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

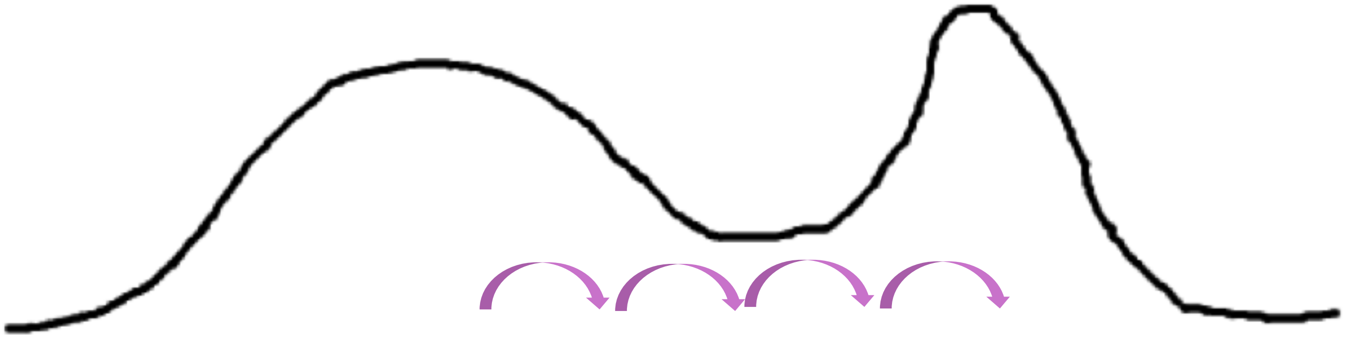
Fast PCD: two sets of weights, one with a large learning rate only used for negative phase, quickly exploring modes

Herding: 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

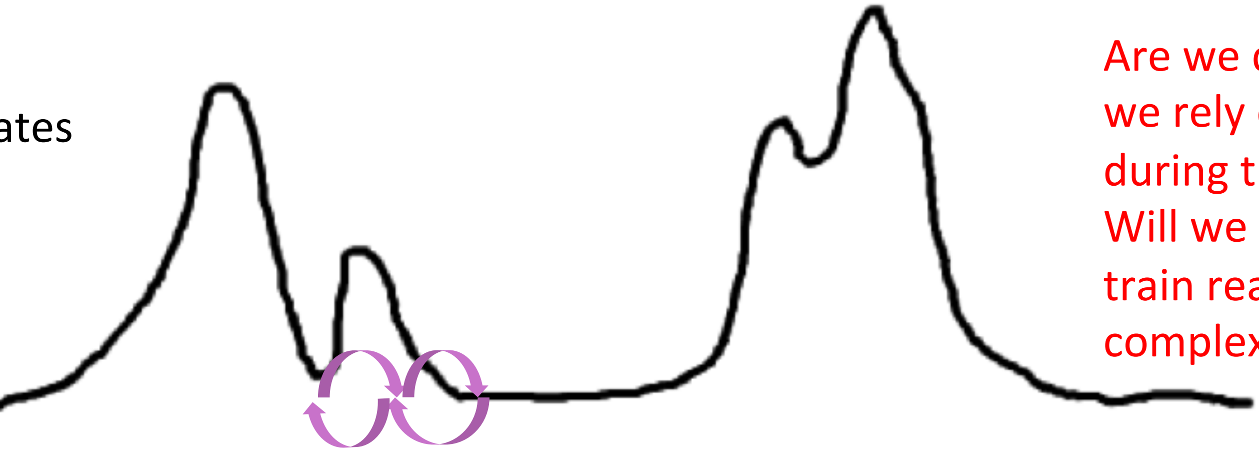


- Later on, hard to cross empty voids between modes

Training updates

vicious circle

Mixing

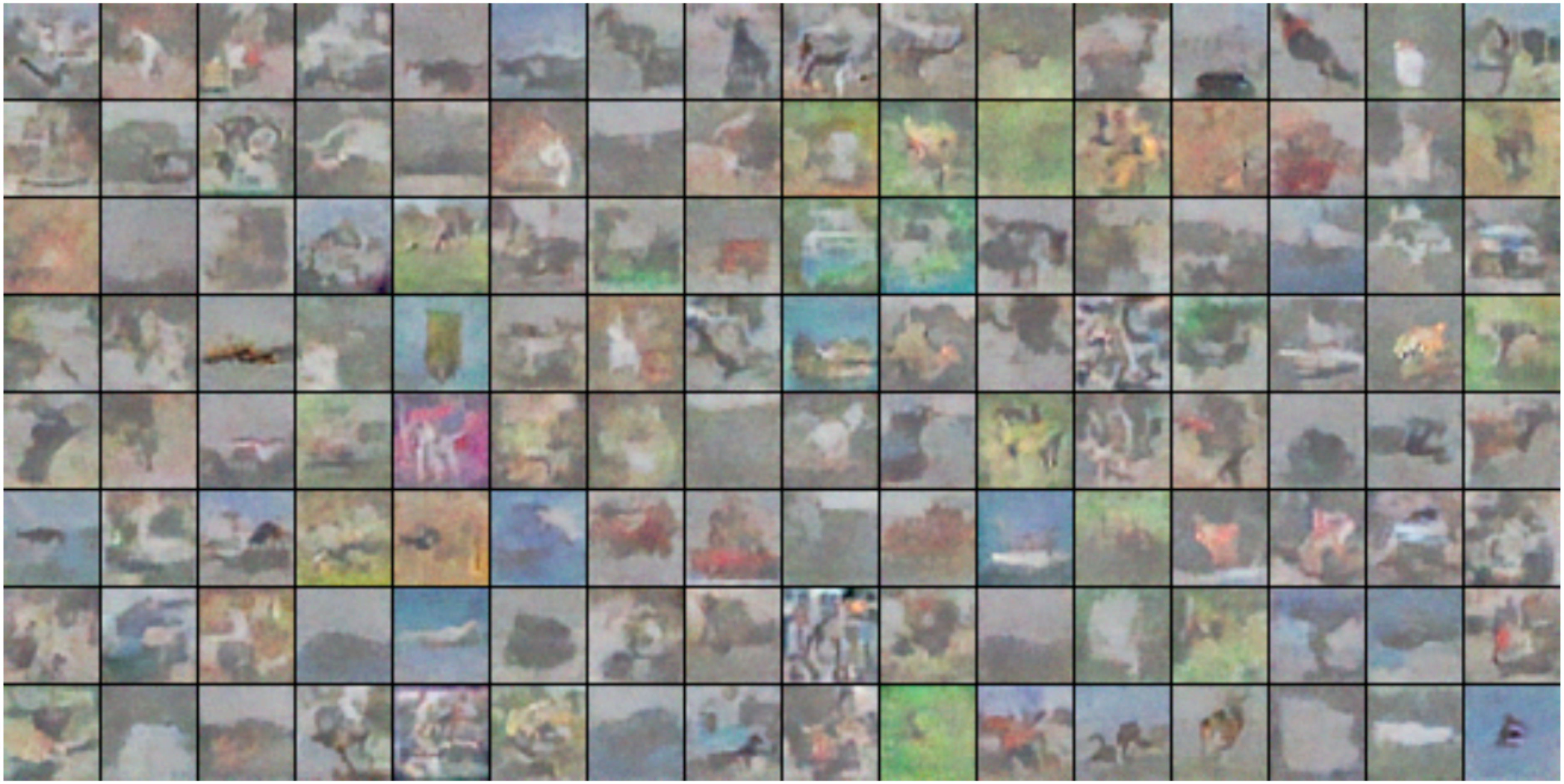


Are we doomed if we rely on MCMC during training? Will we be able to train really large & complex models?

Some RBM Variants

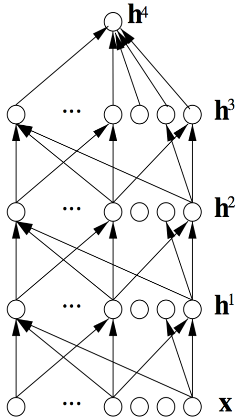
- Different energy functions and allowed values for the hidden and visible units:
  - Hinton et al 2006: binary-binary RBMs
  - Welling NIPS'2004: exponential family units
  - Ranzato & Hinton CVPR'2010: Gaussian RBM weaknesses (no conditional covariance), propose mcRBM
  - Ranzato et al NIPS'2010: mPoT, similar energy function
  - Courville et al ICML'2011: spike-and-slab RBM

Convolutionally Trained Spike & Slab RBMs Samples



Auto-Encoders & Variants: Learning a computational graph

Computational Graphs



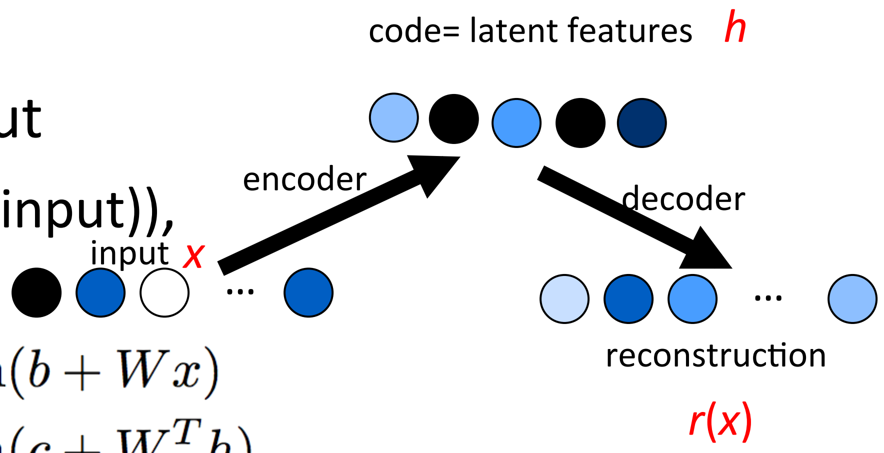
- Operations for particular task
- Neural nets' structure = computational graph for P(y|$\boldsymbol{x}$)
- Graphical model's structure ≠ computational graph for inference
- Recurrent nets & graphical models

  ➔ **family of computational graphs sharing parameters**

- *Could we have a parametrized family of computational graphs defining "the model"?*

Simple Auto-Encoders

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

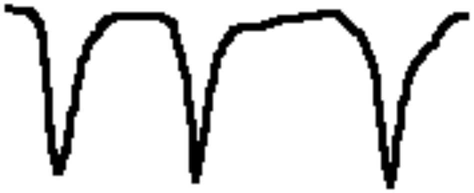


$$\begin{aligned} h &= \tanh(b + Wx) \\ \text{reconstruction} &= \tanh(c + W^T h) \\ \text{Loss } L(x, \text{reconstruction}) &= ||\text{reconstruction} - x||^2 \end{aligned}$$

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

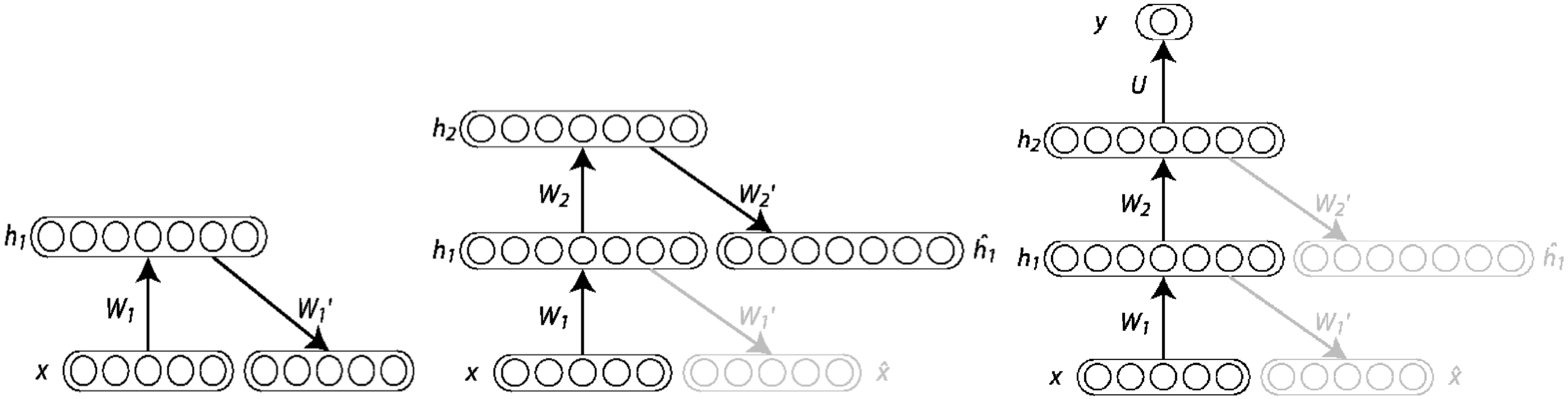
I finally understand what auto-encoders do!

- Try to carve holes in $||r(x)-x||^2$ or –log P(x | h(x)) at examples

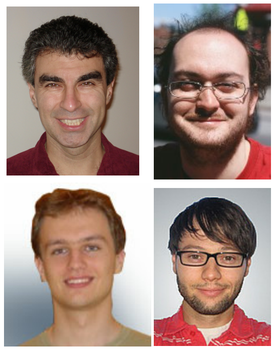


- Vector $r(x)-x$ points in direction of increasing prob., i.e. estimate score = d log $p(x)$ / dx: learn **score** vector field = **local mean**
- Generalize (*valleys*) in between above holes to form *manifolds*
  - $d\, r(x) / dx$ estimates the **local covariance** and is linked to the Hessian $d^2 \log p(x) / dx^2$
- **A Markov Chain associated with AEs estimates the data-generating distribution (Bengio et al, arxiv 1305.663, 2013)**

Stacking Auto-Encoders



Auto-encoders can be stacked successfully (Bengio et al NIPS'2006) to form highly non-linear representations, which with fine-tuning overperformed purely supervised MLPs



(Auto-Encoder) Reconstruction Loss

- Discrete inputs: cross-entropy for binary inputs
  - $- \Sigma_i x_i \log r_i(x) + (1-x_i) \log(1-r_i(x))$ (with $0<r_i(x)<1$)

  or log-likelihood reconstruction criterion, e.g., for a multinomial (one-hot) input

  - $- \Sigma_i x_i \log r_i(x)$ (where $\Sigma_i r_i(x)=1$, summing over subset of inputs associated with this multinomial variable)

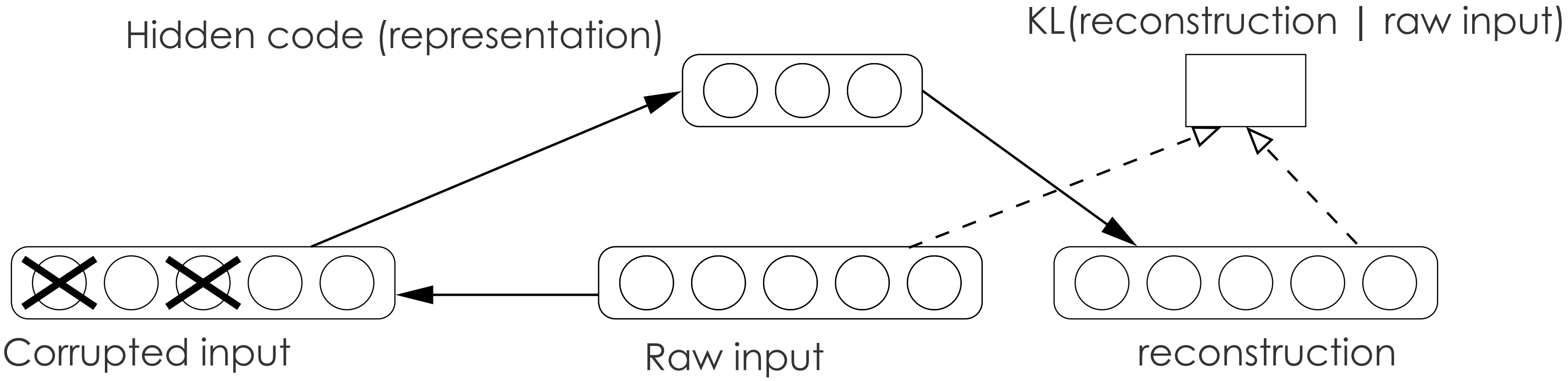
- In general: consider what are appropriate loss functions to predict each of the input variables,

  typically, **reconstruction neg. log-likelihood –log P(x|h(x))**

Denoising Auto-Encoder

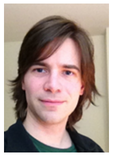
(Vincent et al 2008)

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



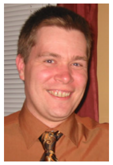
- Encoder & decoder: any parametrization
- As good or better than RBMs for unsupervised pre-training

Denoising Auto-Encoder



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

$$r(x)\text{-}x \propto \mathrm{dlogp}(x)/\mathrm{dx}$$



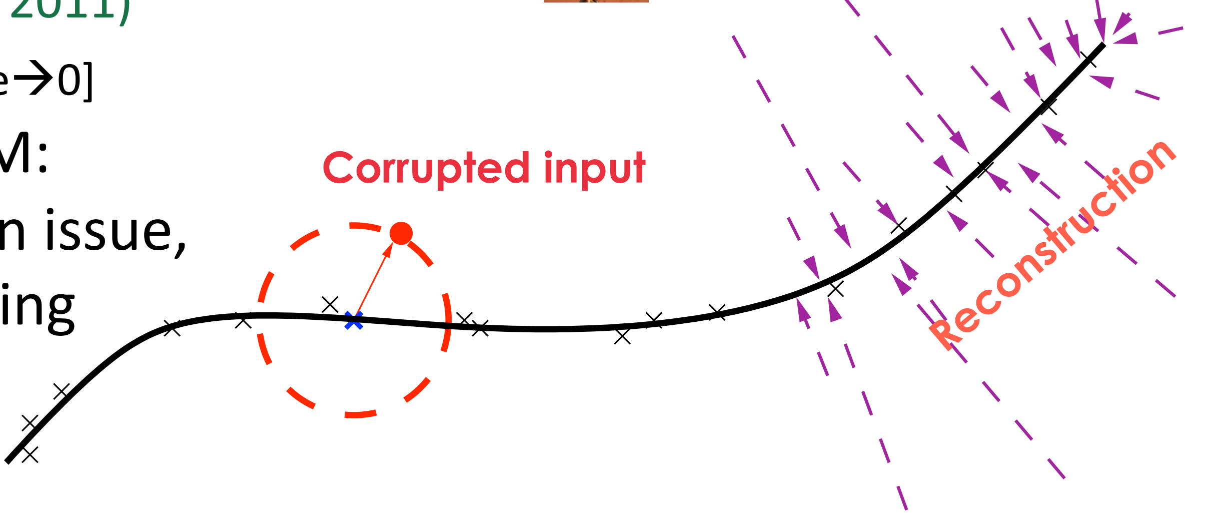
- Some DAEs correspond to a kind of Gaussian RBM with *regularized* Score Matching (Vincent 2011)

  [equivalent when noise→0]

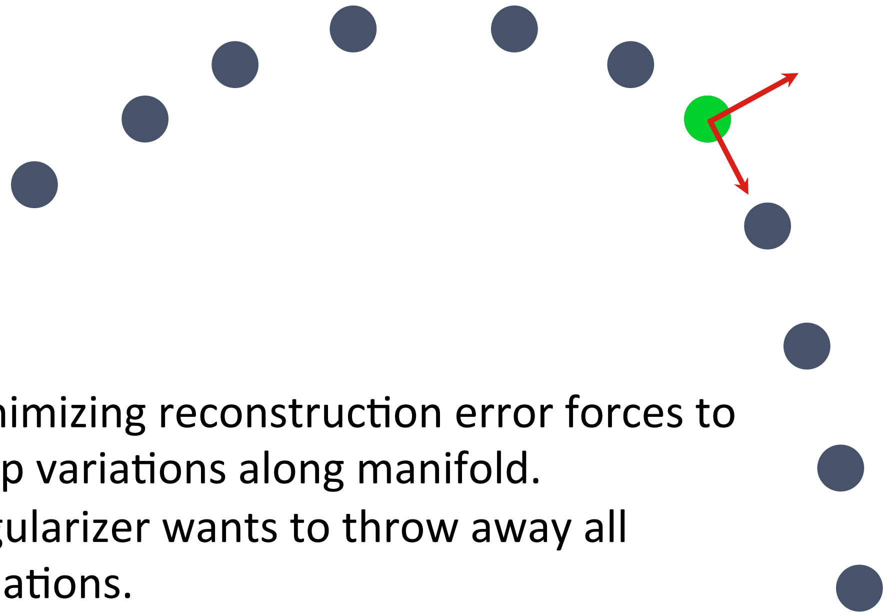
- Compared to RBM:

No partition function issue,
+ can measure training criterion

prior: examples concentrate near a lower dimensional "manifold"



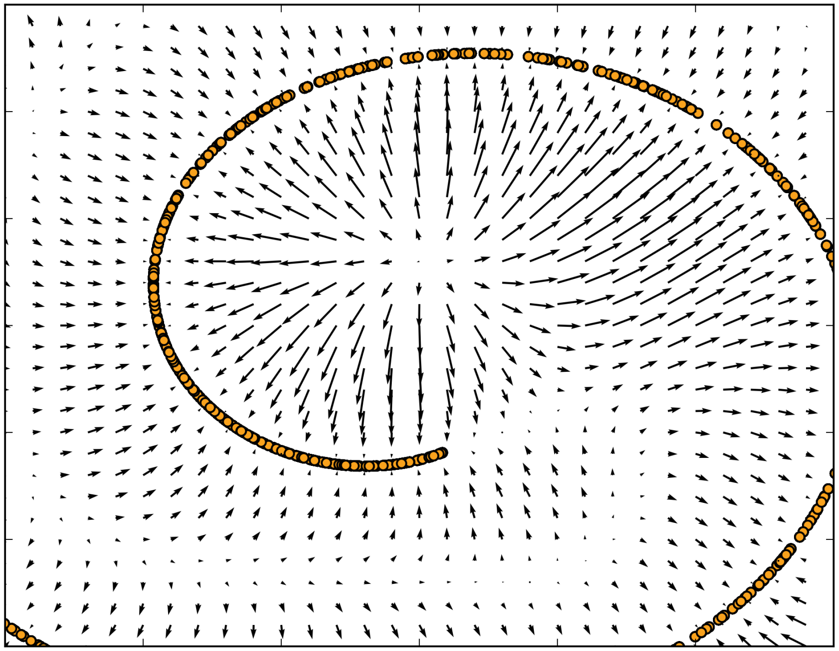
Auto-Encoders Learn Salient Variations, like a non-linear PCA

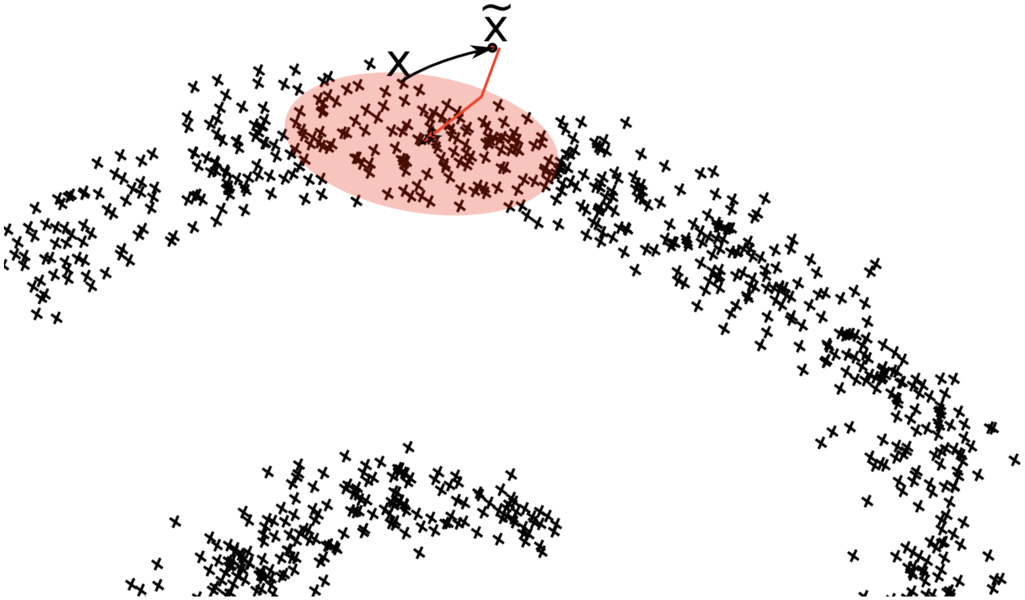


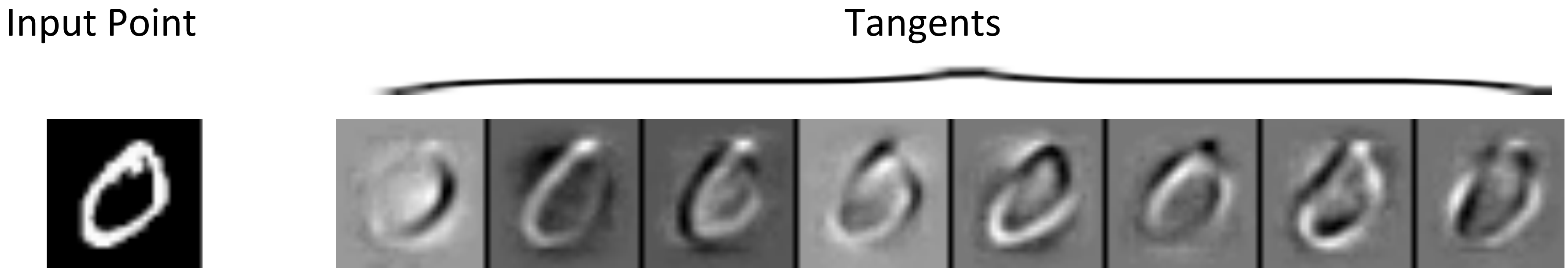
- Minimizing reconstruction error forces to keep variations along manifold.
- Regularizer wants to throw away all variations.
- With both: keep ONLY sensitivity to variations ON the manifold.

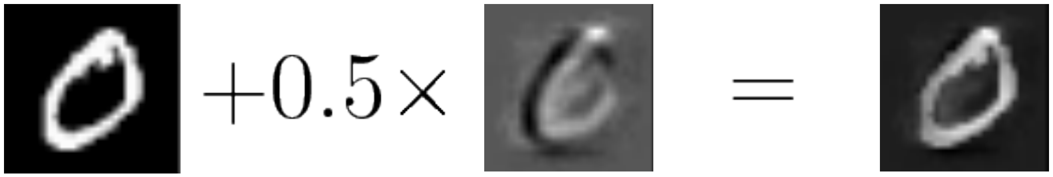
Regularized Auto-Encoders Learn a Vector Field or a Markov Chain Transition Distribution

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

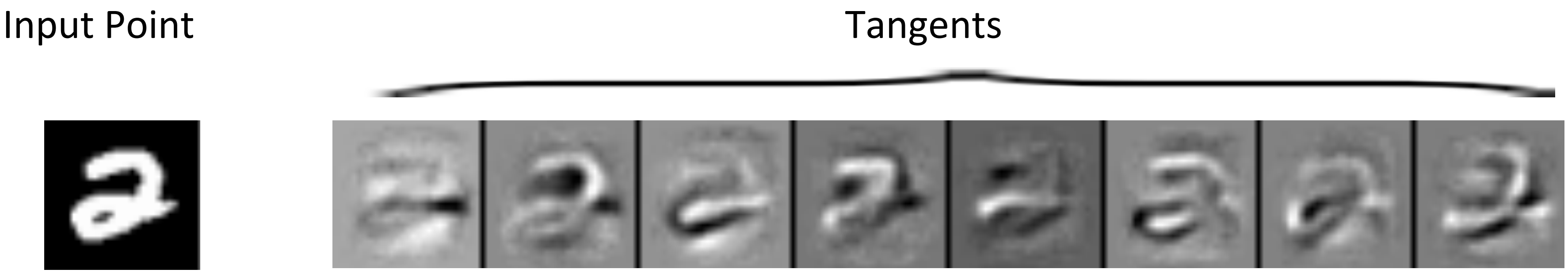








MNIST

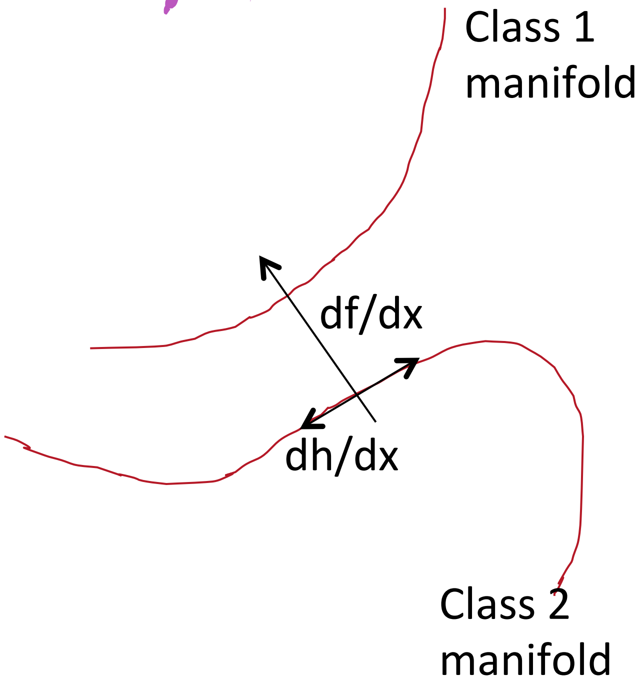


MNIST Tangents

Learned Tangent Prop: the Manifold Tangent Classifier

Makes classifier f(x) insensitive to variations on manifold at x

Tangent plane characterized by dh(x)/dx



(Rifai et al NIPS'2012)

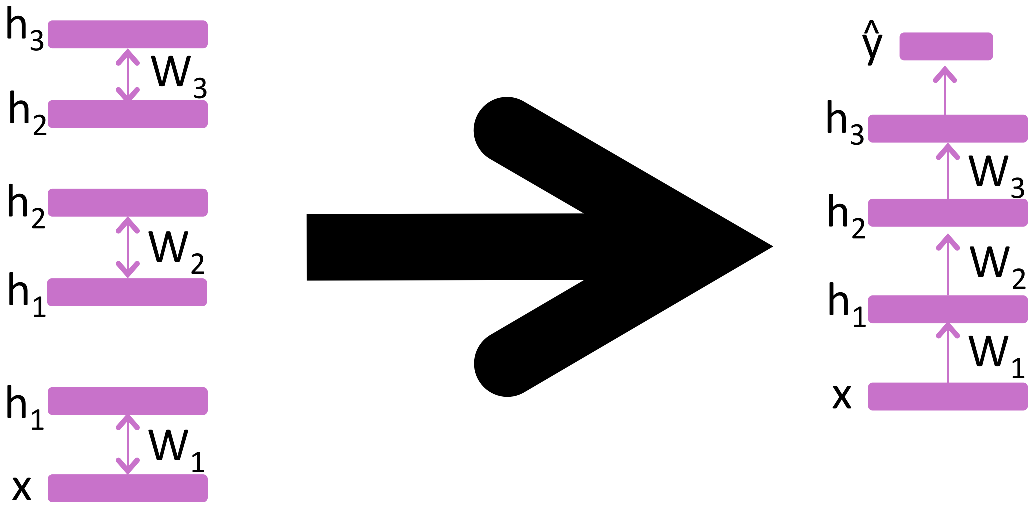
Deep Variants

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

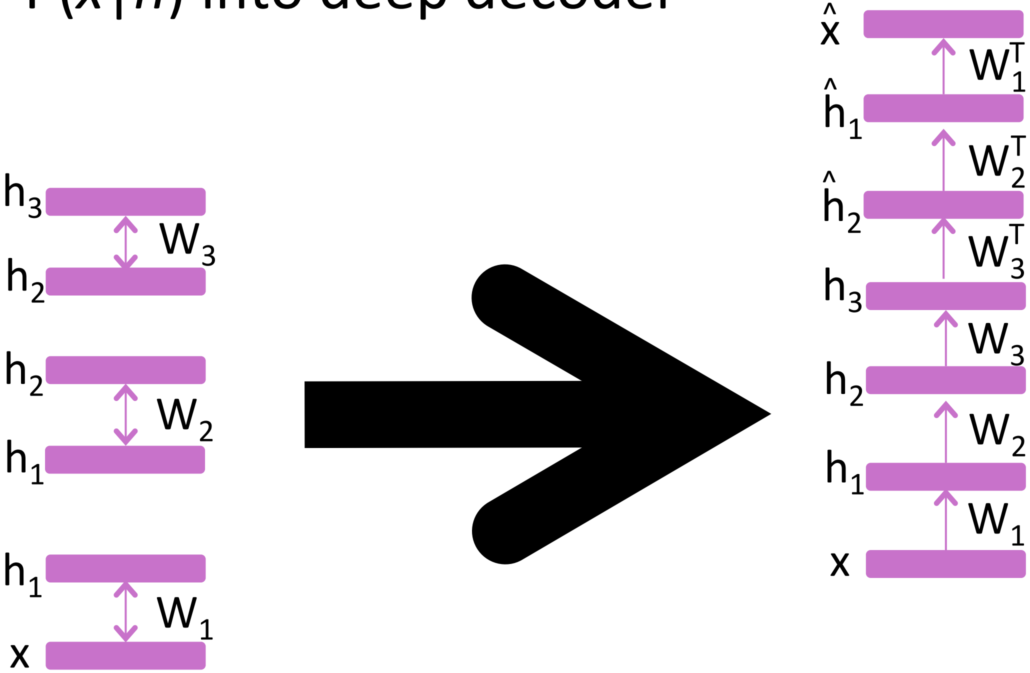
- Encoder or P(*h*|*v*) becomes MLP layer



Stack of RBMs / AEs → Deep Auto-Encoder

(Hinton & Salakhutdinov 2006)

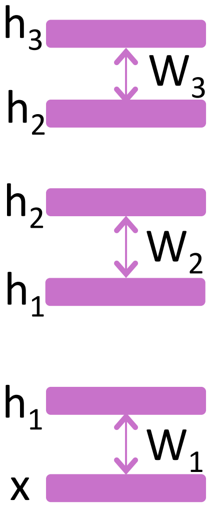
- Stack encoders / P(h|x) into deep encoder
- Stack decoders / P(x|h) into deep decoder



Stack of RBMs / AEs → Deep Recurrent Auto-Encoder

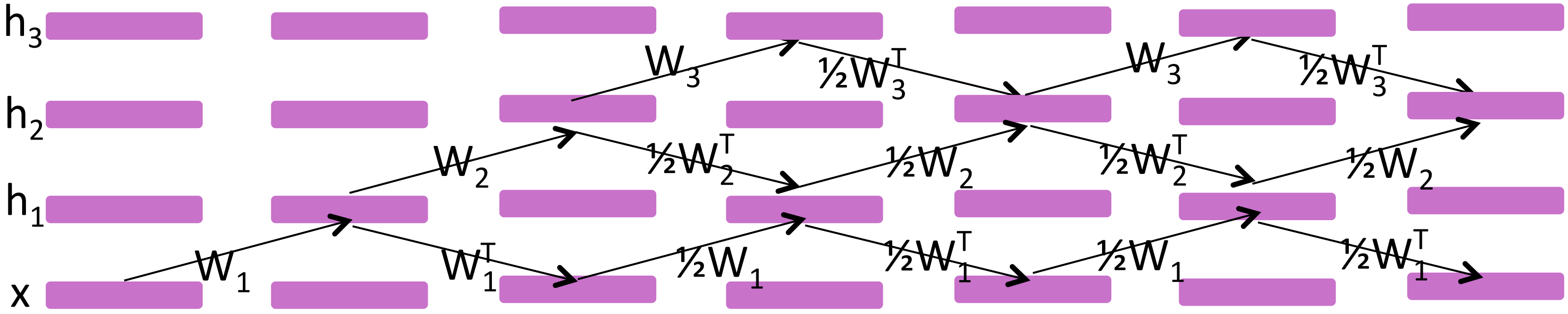
(Savard 2011)

(Bengio & Laufer, arxiv 2013)

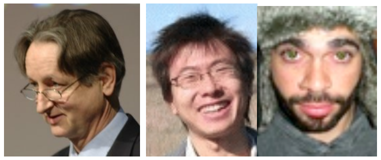


- Each hidden layer receives input from below and above
- Deterministic (mean-field) recurrent computation (Savard 2011)
- Stochastic (injecting noise) recurrent computation: Deep Generative Stochastic Networks (GSNs)

(Bengio & Laufer arxiv 2013)

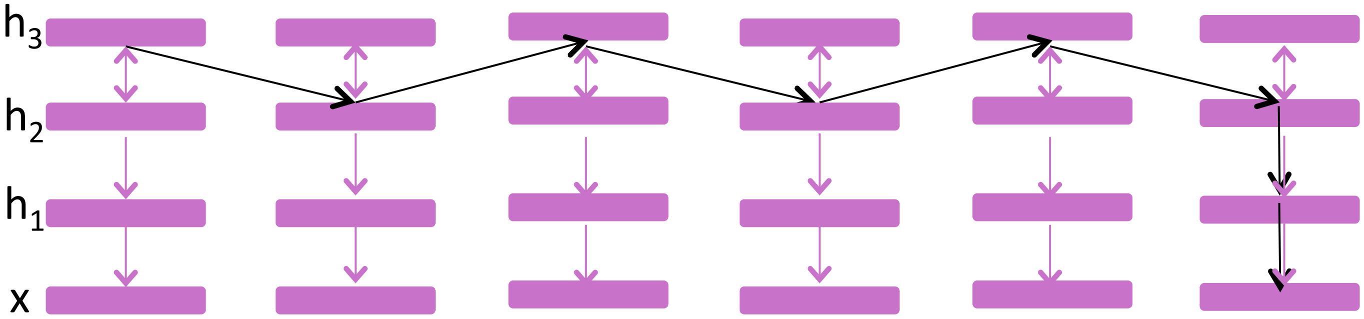


Stack of RBMs → Deep Belief Net



(Hinton et al 2006)

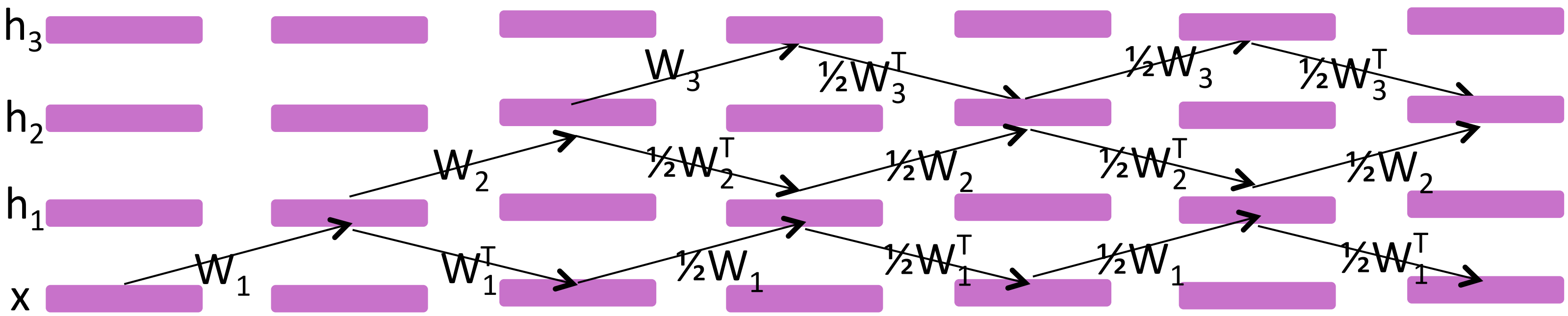
- Stack lower levels RBMs' P(*x*|*h*) along with top-level RBM
- $P(x, h_1, h_2, h_3) = P(h_2, h_3)\ P(h_1|h_2)\ P(x \mid h_1)$
- Sample: Gibbs on top RBM, propagate down



Stack of RBMs → Deep Boltzmann Machine

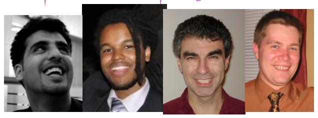
(Salakhutdinov & Hinton AISTATS 2009)

- 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



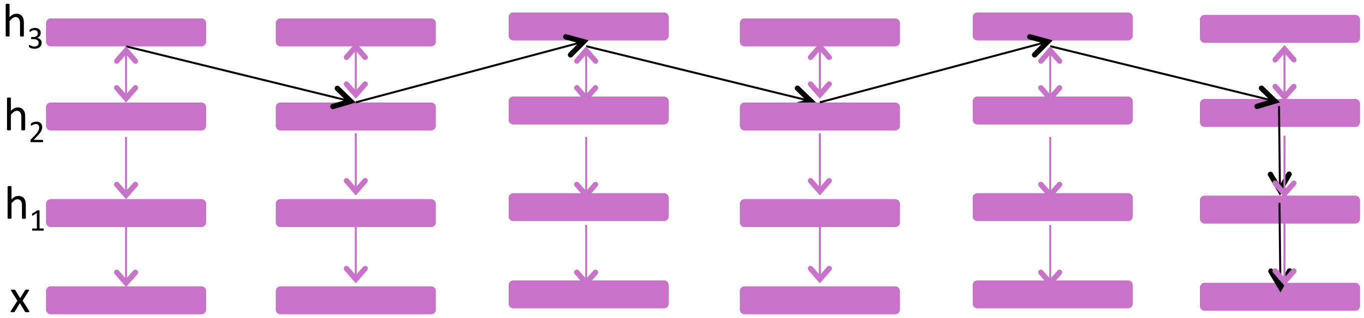
Stack of Auto-Encoders → Deep Generative Auto-Encoder

(Rifai et al ICML 2012)



- MCMC on top-level auto-encoder
  - h_{t+1} = encode(decode(h_t))+σ noise

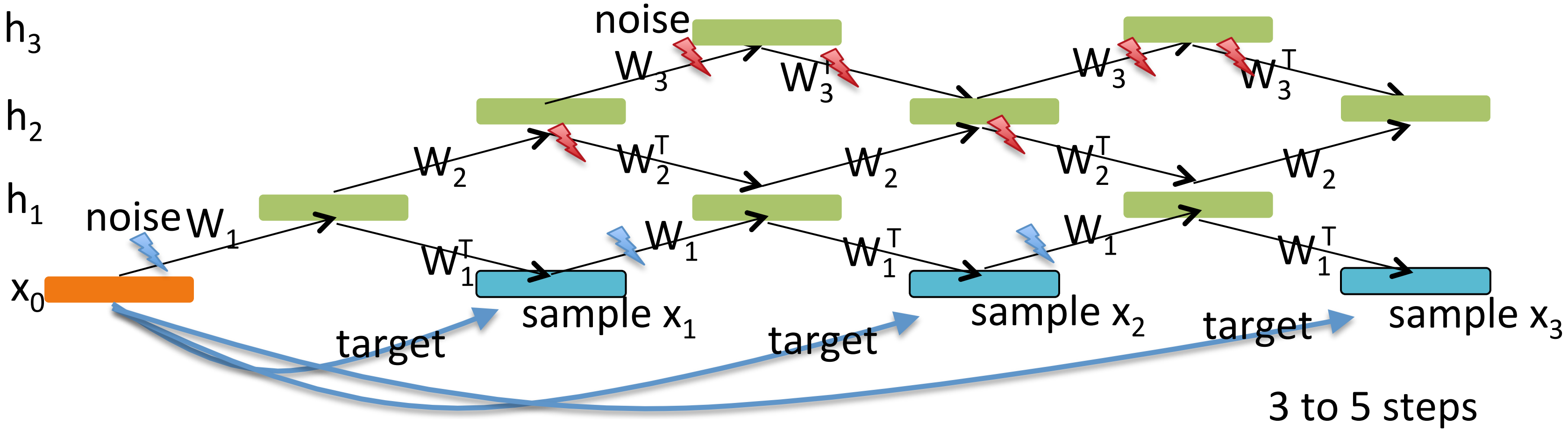
  where noise is Normal(0, d/dh encode(decode(h_t)))
- Then deterministically propagate down with decoders



Generative Stochastic Networks (GSN)

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

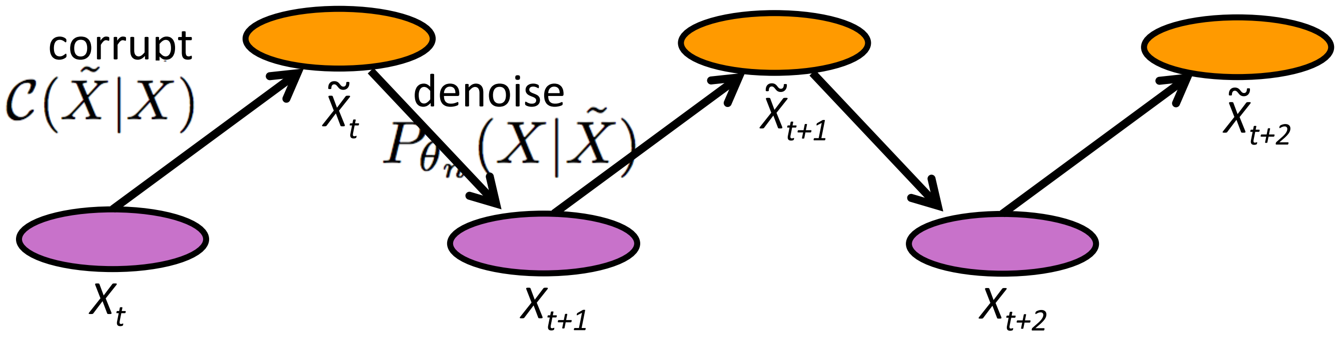
- **Recurrent parametrized stochastic computational graph that defines a transition operator for a Markov chain whose asymptotic distribution is implicitly estimated by the model**
- Noise injected in input and hidden layers
- Trained to max. reconstruction prob. of example at each step
- **Example**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



3 to 5 steps

Denoising Auto-Encoder Markov Chain

- $\mathcal{P}(X)$: true data-generating distribution
- $\mathcal{C}(\tilde{X}|X)$: corruption process
- $P_{\theta_n}(X|\tilde{X})$: denoising auto-encoder trained with *n* examples $X, \tilde{X}$ from $\mathcal{C}(\tilde{X}|X)\mathcal{P}(X)$, probabilistically "inverts" corruption
- T_n : Markov chain over *X* alternating $\tilde{X} \sim \mathcal{C}(\tilde{X}|X)$, $X \sim P_{\theta_n}(X|\tilde{X})$



New Theoretical Results (Bengio et al NIPS 2013)

- Denoising AE are consistent estimators of the data-generating distribution through their Markov chain, so long as they consistently estimate the conditional denoising distribution and the Markov chain converges.

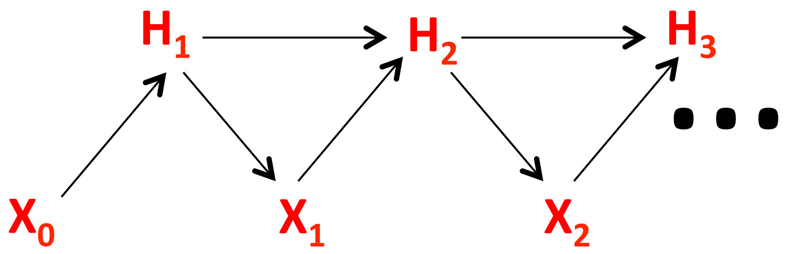
$$\text{Making } \underset{\substack{\nearrow \\ \text{denoising distr.}}}{P_{\theta_n}}(X|\tilde{X}) \text{ match } \underset{\substack{\nearrow \\ \text{truth}}}{\mathcal{P}}(X|\tilde{X}) \text{ makes } \underset{\substack{\nearrow \\ \text{stationary distr.}}}{\pi_n}(X) \text{ match } \underset{\substack{\nearrow \\ \text{truth}}}{\mathcal{P}}(X)$$

Generative Stochastic Networks

Bengio et al, ICML 2014

- Generalizes the denoising auto-encoder training scheme
  - Introduce latent variables in the Markov chain (over X,H)
  - Instead of a fixed corruption process, have a deterministic function with parameters θ_1 and a noise source Z as input

$$H_{t+1} = f_{\theta_1}(X_t, Z_t, H_t)$$



$$\begin{aligned} H_{t+1} &\sim P_{\theta_1}(H|H_t, X_t) \\ X_{t+1} &\sim P_{\theta_2}(X|H_{t+1}) \end{aligned}$$

A Proper Generative Model for Dependency Networks, MP-DBMs, and efficient deep NADE sampling

- Dependency nets (Heckerman et al 2000) estimate $P_{\theta_i}(X_i \mid X_{-i})$ not guaranteed to be conditionals of a unique joint
- Heckerman et al's sampling iterates over i: not ergodic?
- Randomly choosing i: proper GSN
- Defines a unique joint distribution = stationary distr. of chain (which averages out over resampling orders)
- Generalized to estimators of P(subset(X) | X \ subset(X)) and justify efficient sampling schemes for MP-DBMs and deep NADE.

Applications

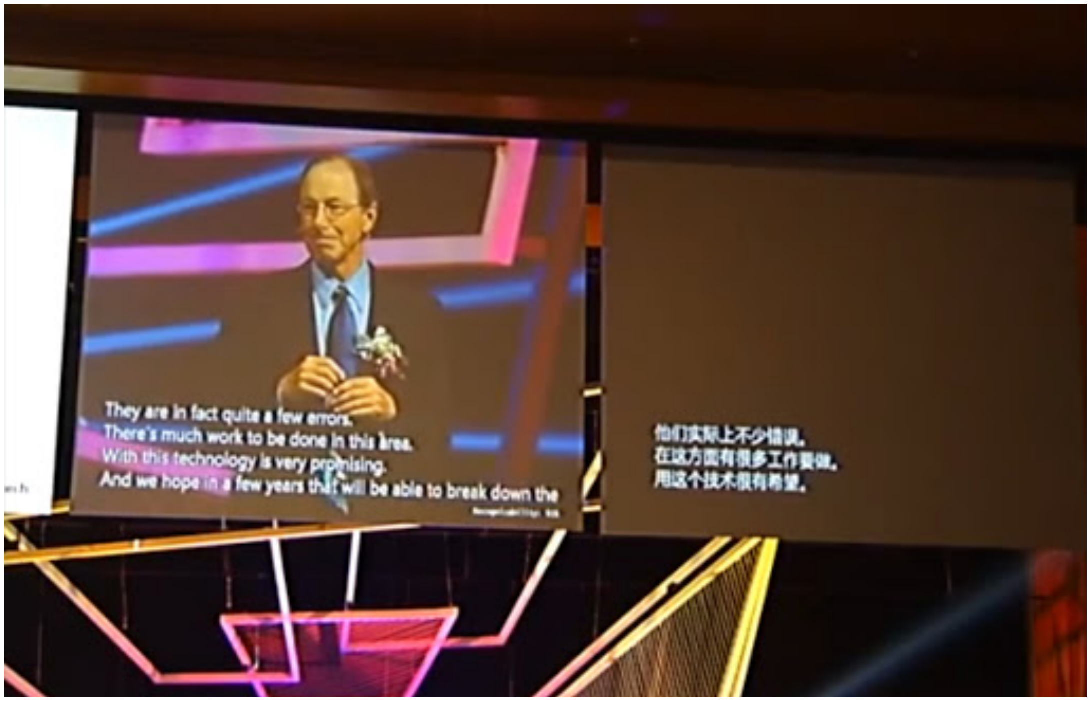
AI Tasks

- 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

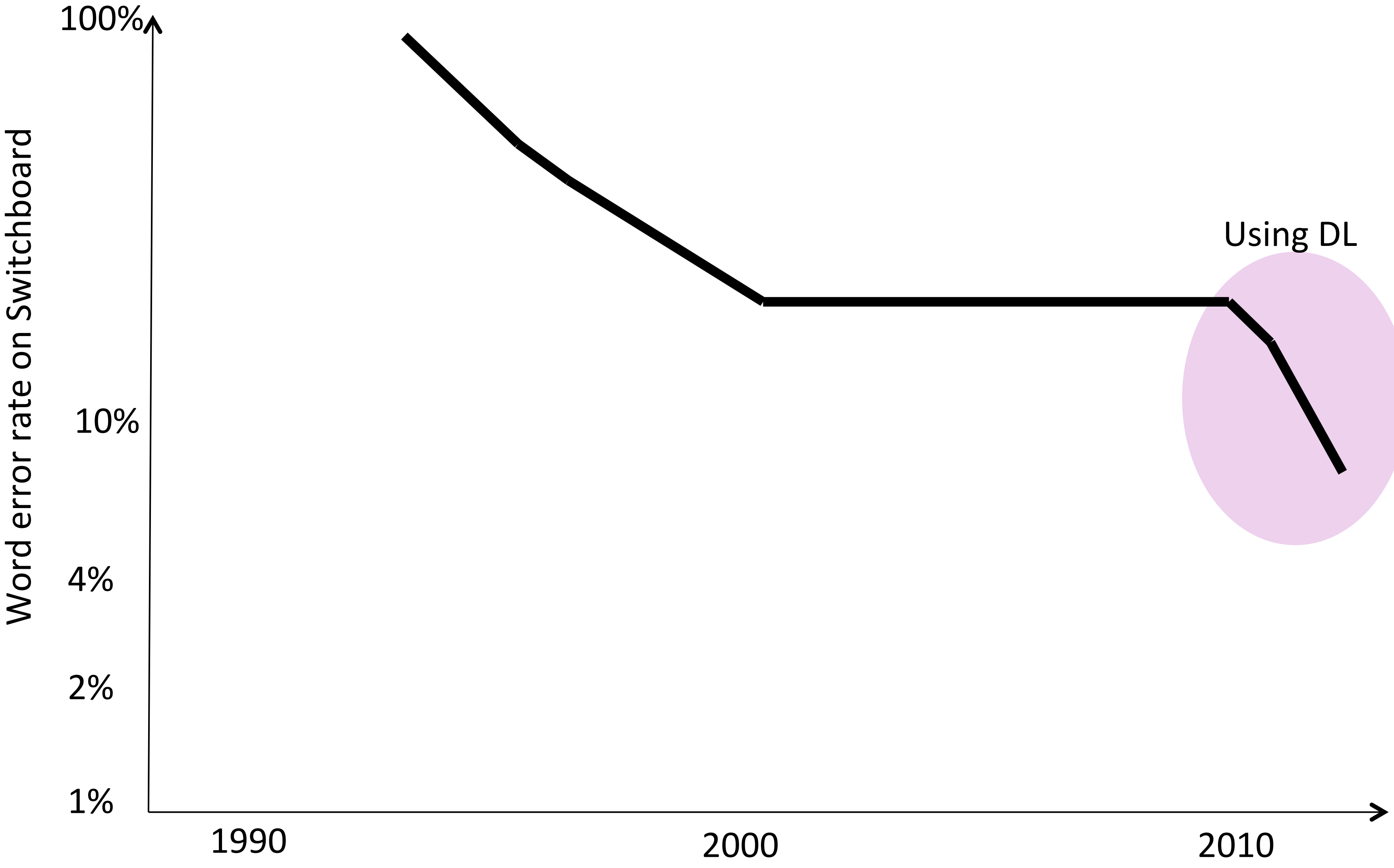
- Google uses DL in their android speech recognizer (both server-side and on some phones with enough memory)
- Microsoft uses DL in their speech recognizer
- Error reductions on the order of 30%, a major progress



They are in fact quite a few errors.
There's much work to be done in this area.
With this technology is very promising.
And we hope in a few years that will be able to break down the

他们实际上不少错误。
在这方面有很多工作要做。
用这个技术很有希望。

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	

(numbers taken from Geoff Hinton's June 22, 2012 Google talk)

Some Applications of DL

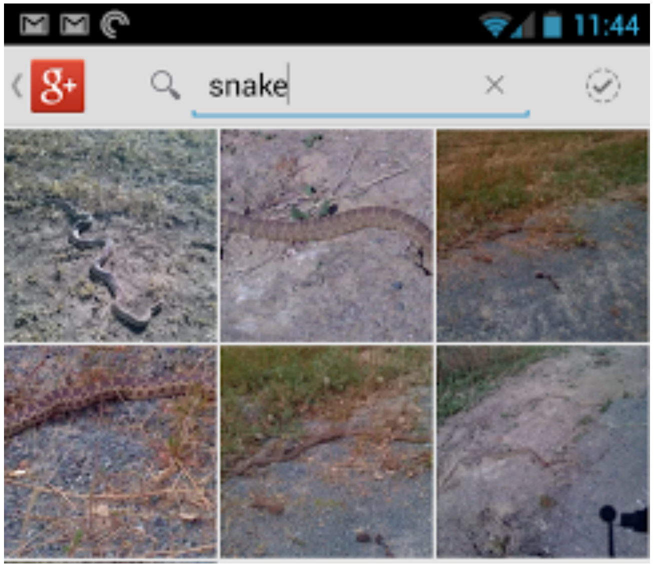
- **Language Modeling** (Speech Recognition, Machine Translation)
- **Acoustic Modeling** (**speech recognition**, music modeling)
- **NLP syntactic/semantic tagging** (Part-Of-Speech, chunking, Named Entity Recognition, Semantic Role Labeling, Parsing)
- **NLP applications**: sentiment analysis, paraphrasing, 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

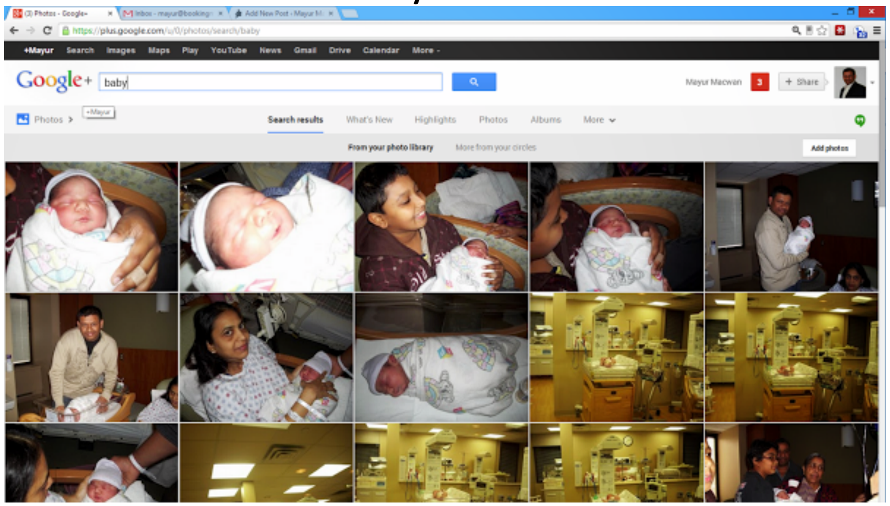
- Krizhevsky, Sutskever & Hinton NIPS 2012

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

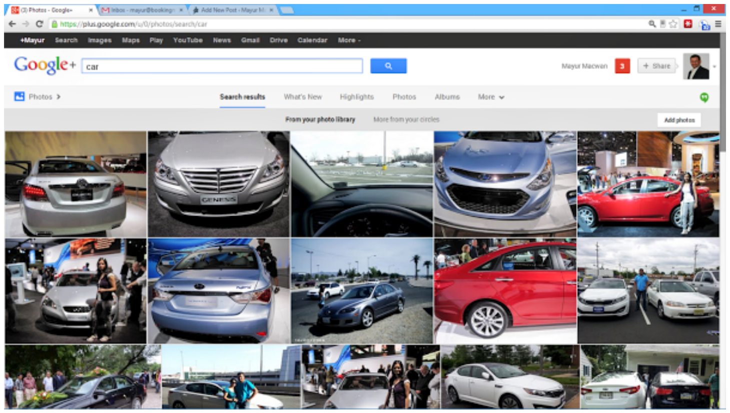
- **Google incorporates DL in Google+ photo search**, "A step across the semantic gap" (Google Research blog, June 12, 2013)
- Baidu now offers similar services



baby

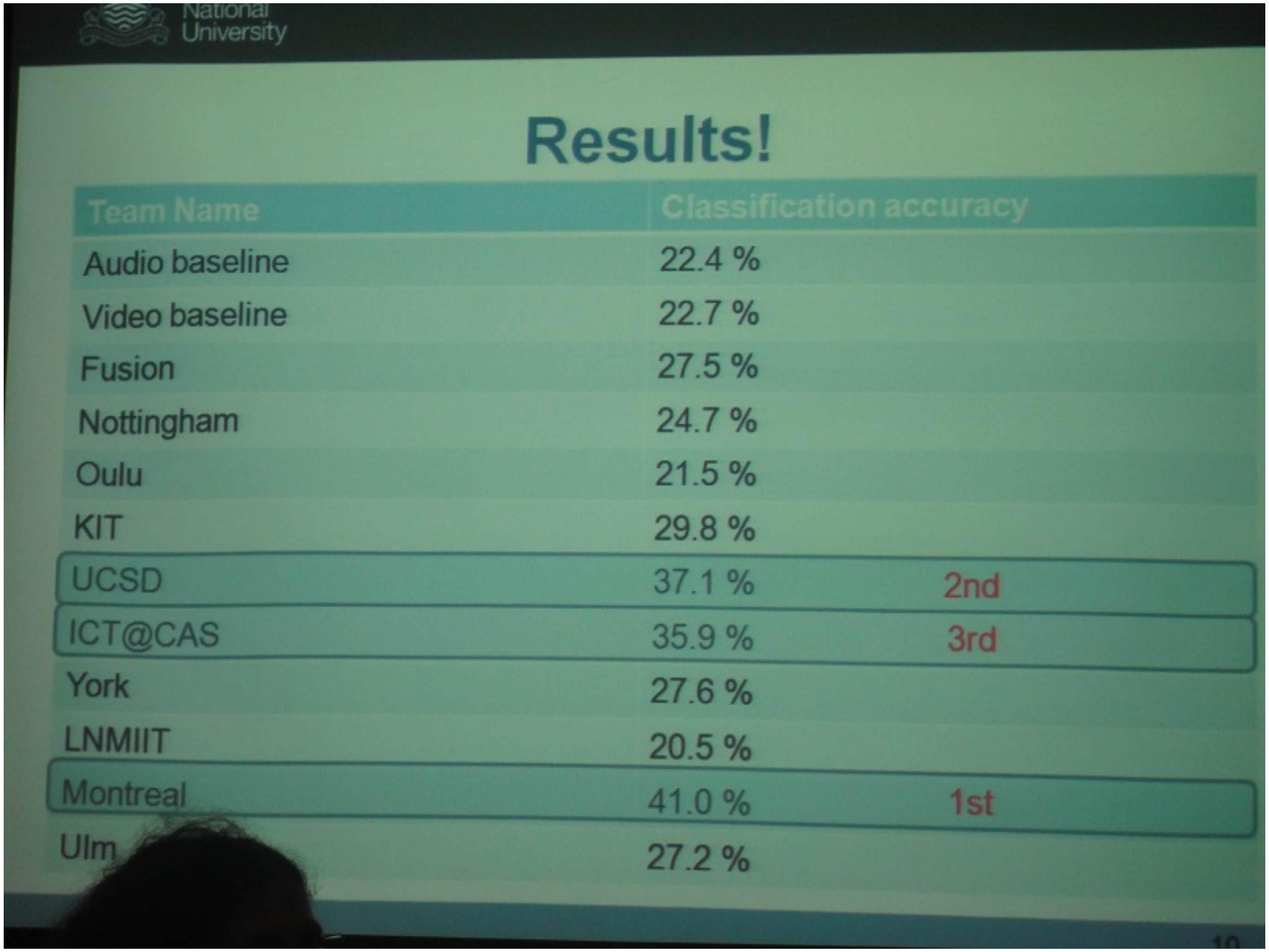


car



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 %	2nd
ICT@CAS	35.9 %	3rd
York	27.6 %	
LNMIIT	20.5 %	
Montreal	41.0 %	1st
Ulm	27.2 %	

More Successful Applications

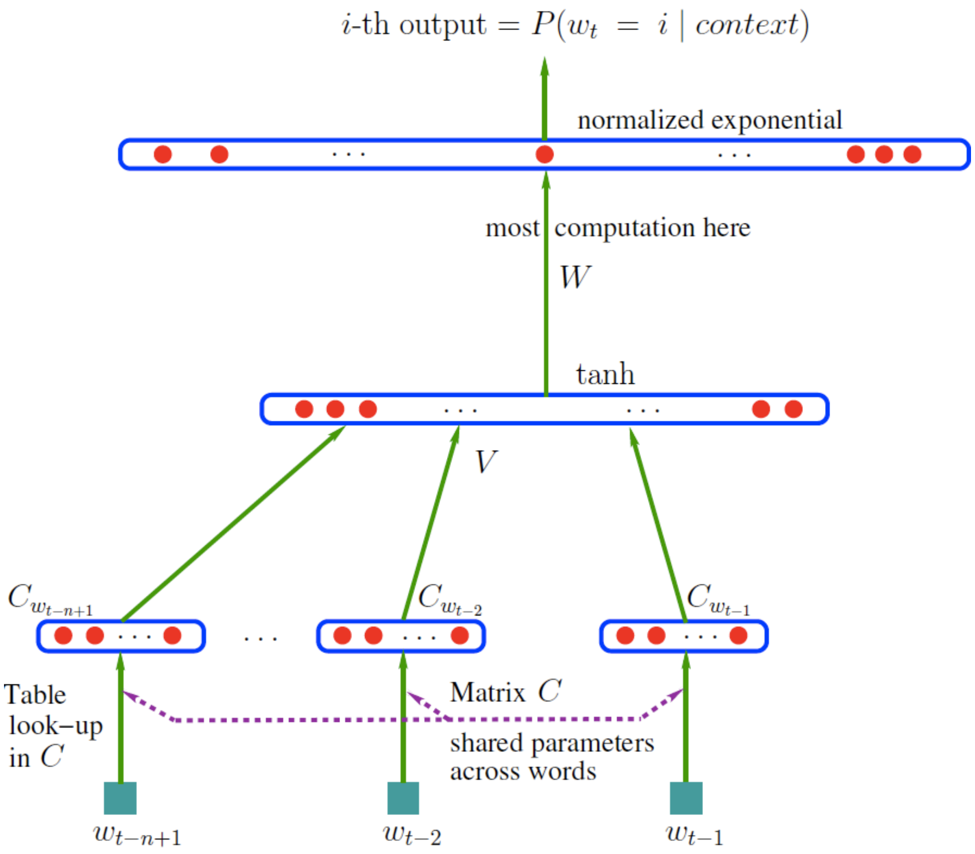
- Microsoft uses DL for speech rec. service (audio video indexing), based on Hinton/Toronto's DBNs (Mohamed et al 2012)
- Google uses DL in its Google Goggles service, using Ng/Stanford DL systems, and in its Google+ photo search service, using deep convolutional nets
- NYT talks about these: http://www.nytimes.com/2012/06/26/technology/in-a-big-network-of-computers-evidence-of-machine-learning.html?_r=1
- 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

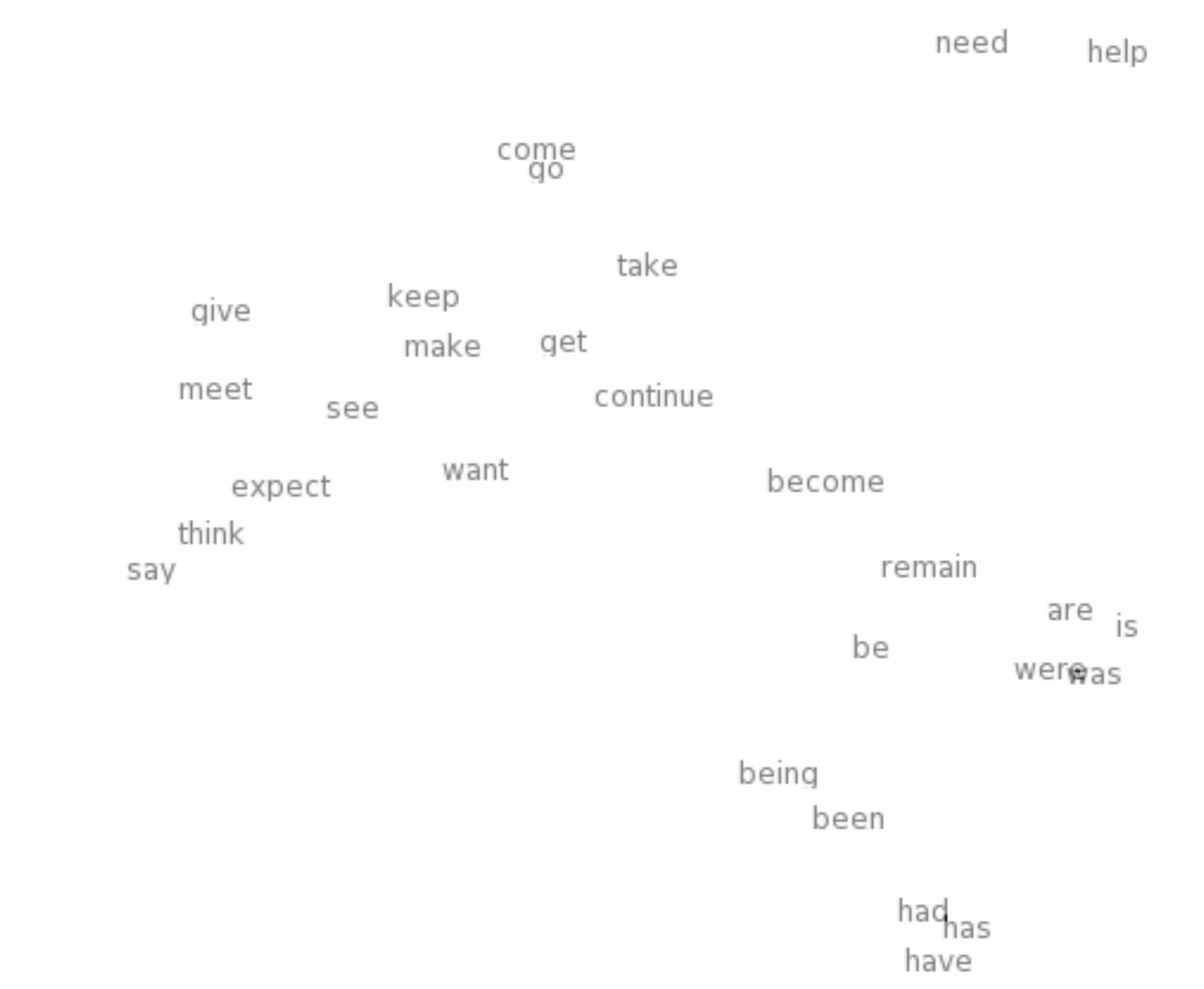
- Language Modeling (Speech Recognition, Machine Translation)
- Acoustic Modeling
- Part-Of-Speech Tagging
- Chunking
- Named Entity Recognition
- Semantic Role Labeling
- Parsing
- Sentiment Analysis
- Paraphrasing
- Question-Answering
- Word-Sense Disambiguation

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

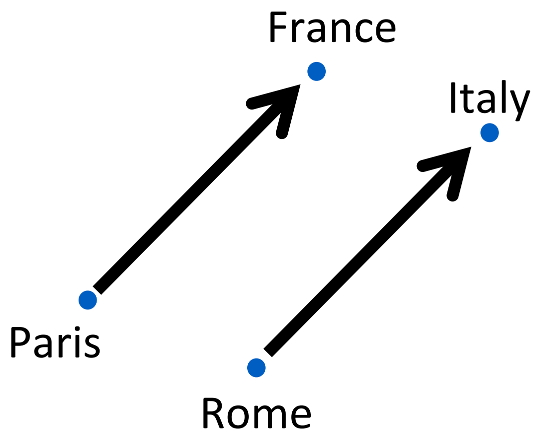


Neural word embeddings - visualization



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

- Semantic relations appear as linear relationships in the space of learned representations
- King – Queen ≈ Man – Woman
- Paris – France + Italy ≈ 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, θ = parameter vector, and ε_t = learning rate.
- Ordinary gradient descent is a batch method, very slow, should never be used. 2nd 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 ε_0 and τ.
- 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

- In very deep networks such as **recurrent networks** (or possibly recursive ones), the gradient is a product of Jacobian matrices, each associated with a step in the forward computation. This can become very small or very large quickly [Bengio et al 1994], and the locality assumption of gradient descent breaks down.

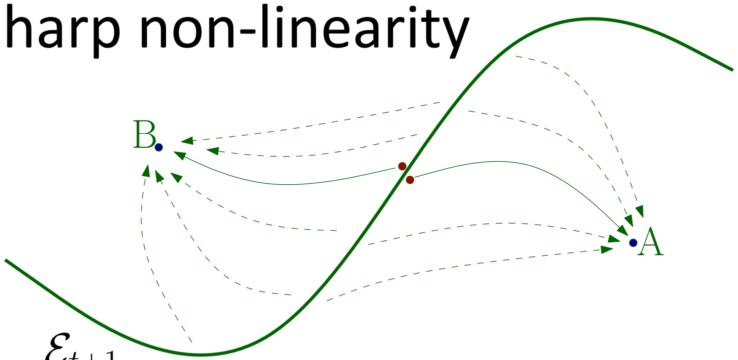
$$L = L(s_T(s_{T-1}(\ldots s_{t+1}(s_t, \ldots))))$$

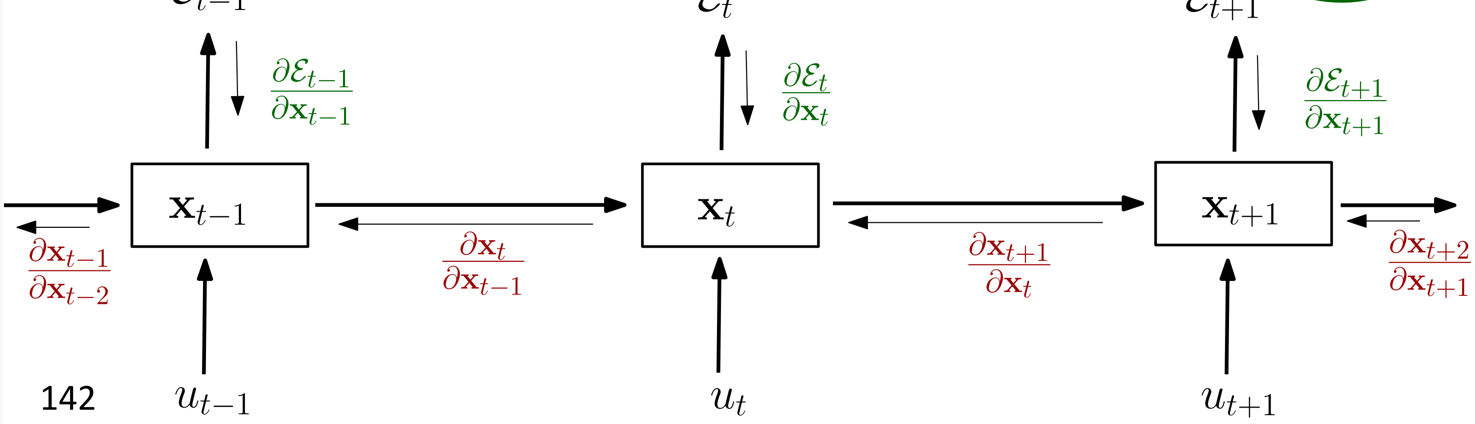
$$\frac{\partial L}{\partial s_t} = \frac{\partial L}{\partial s_T} \frac{\partial s_T}{\partial s_{T-1}} \cdots \frac{\partial s_{t+1}}{\partial s_t}$$

- Two kinds of problems:
  - sing. values of Jacobians > 1 → gradients explode
  - or sing. values < 1 → gradients shrink & vanish

The Optimization Challenge in Deep / Recurrent Nets

- Higher-level abstractions require highly non-linear transformations to be learned
- Sharp non-linearities are difficult to learn by gradient
- Composition of many non-linearities = sharp non-linearity
- Exploding or vanishing gradients

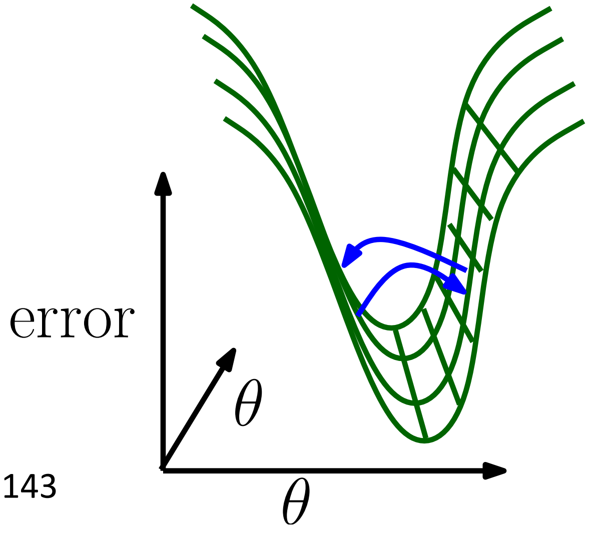


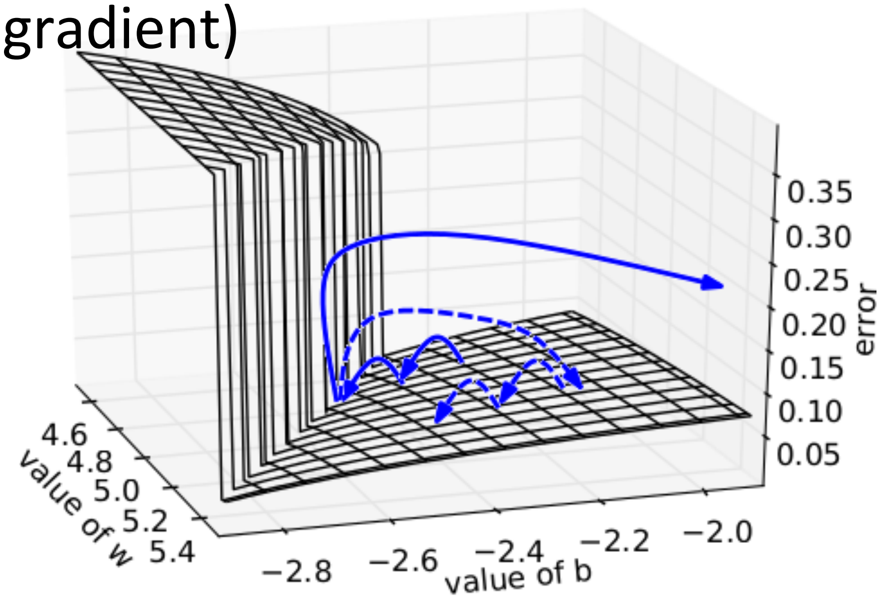

$\mathcal{E}_{t-1}$
$\frac{\partial \mathcal{E}_{t-1}}{\partial \mathbf{x}_{t-1}}$
$\mathcal{E}_t$
$\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t}$
$\mathcal{E}_{t+1}$
$\frac{\partial \mathcal{E}_{t+1}}{\partial \mathbf{x}_{t+1}}$
$\frac{\partial \mathbf{x}_{t-1}}{\partial \mathbf{x}_{t-2}}$
$\mathbf{x}_{t-1}$
$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_{t-1}}$
$\mathbf{x}_t$
$\frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{x}_t}$
$\mathbf{x}_{t+1}$
$\frac{\partial \mathbf{x}_{t+2}}{\partial \mathbf{x}_{t+1}}$
u_{t-1}
u_t
u_{t+1}

RNN Tricks

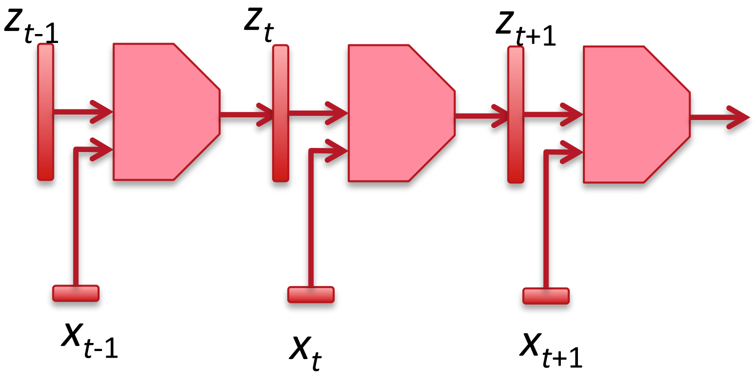
(Pascanu, Mikolov, Bengio, ICML 2013; Bengio, Boulanger & Pascanu, ICASSP 2013)

- Clipping gradients (avoid exploding gradients)
- Leaky integration (propagate long-term dependencies)
- Momentum (cheap 2nd order)
- Initialization (start in right ballpark avoids exploding/vanishing)
- Sparse Gradients (symmetry breaking)
- Gradient propagation regularizer (avoid vanishing gradient)
- LSTM self-loops (avoid vanishing gradient)





Long-Term Dependencies and Clipping Trick



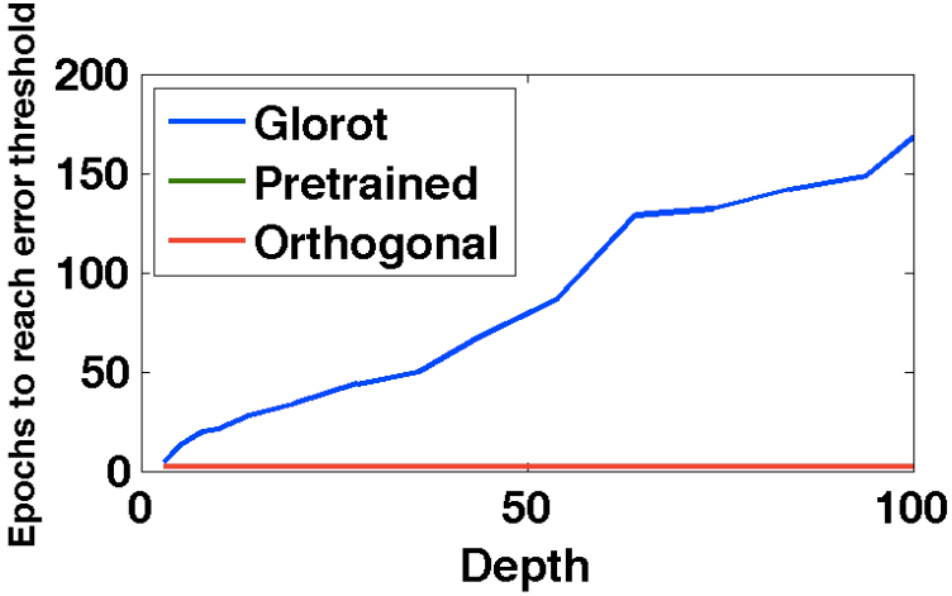
Trick first introduced by Mikolov is to clip gradients to a maximum NORM value.

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.

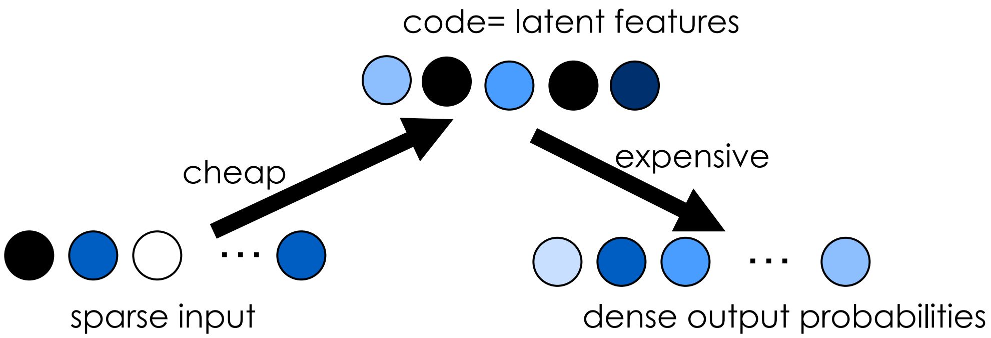
Orthogonal Initialization Works Even Better

- 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

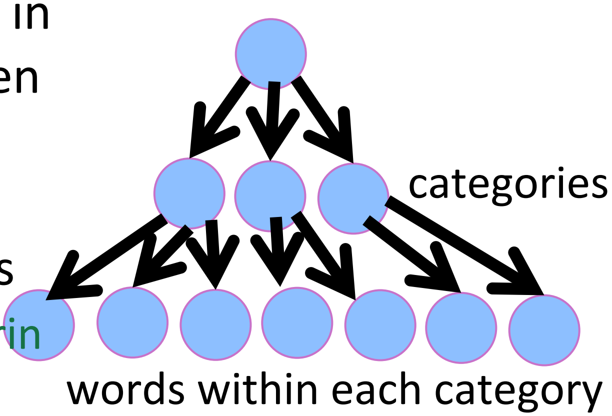


Handling Large Output Spaces

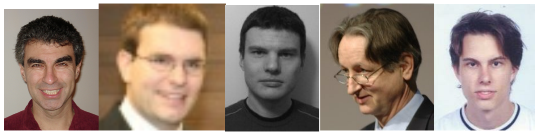
- Auto-encoders and RBMs reconstruct the input, which is sparse and high-dimensional; Language models have a huge output space (1 unit per word).



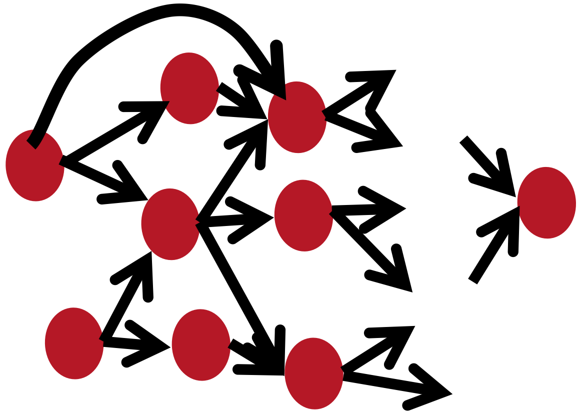
- (Dauphin et al, ICML 2011) Reconstruct the non-zeros in the input, and reconstruct as many randomly chosen zeros, + importance weights



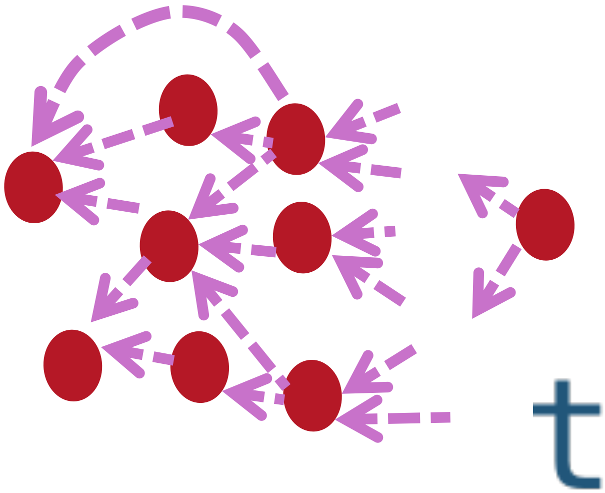
- (Collobert & Weston, ICML 2008) sample a ranking loss
- Decompose output probabilities hierarchically (Morin & Bengio 2005; Blitzer et al 2005; Mnih & Hinton 2007,2009; Mikolov et al 2011)



Automatic Differentiation



- 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.



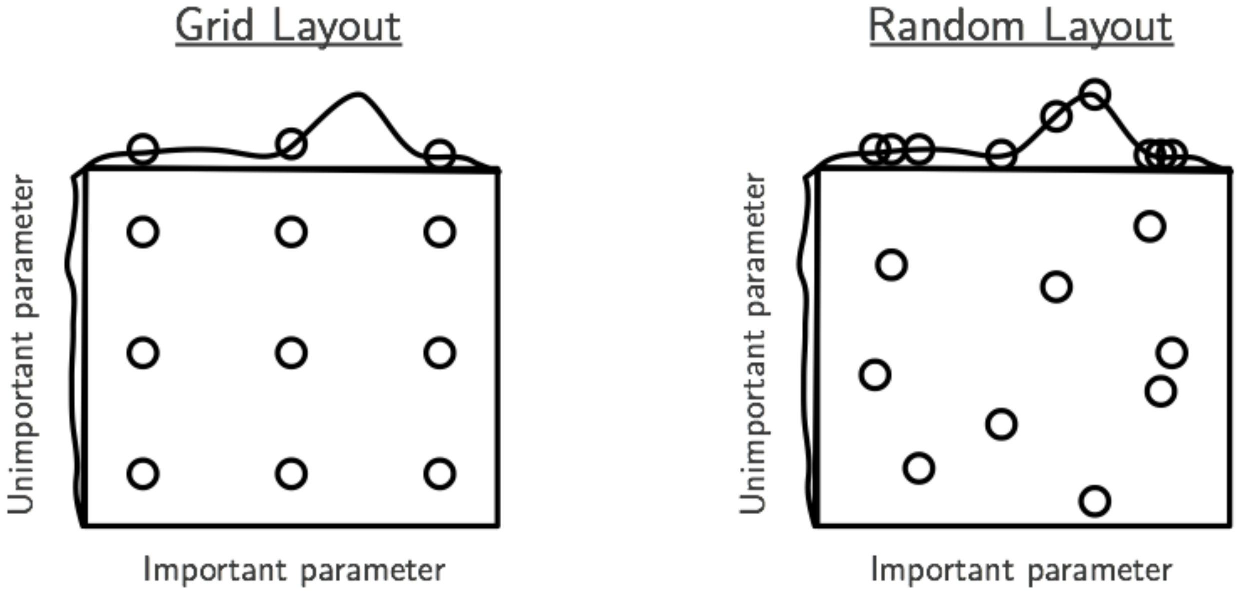
(Bergstra et al SciPy'2010)

theano

Random Sampling of Hyperparameters

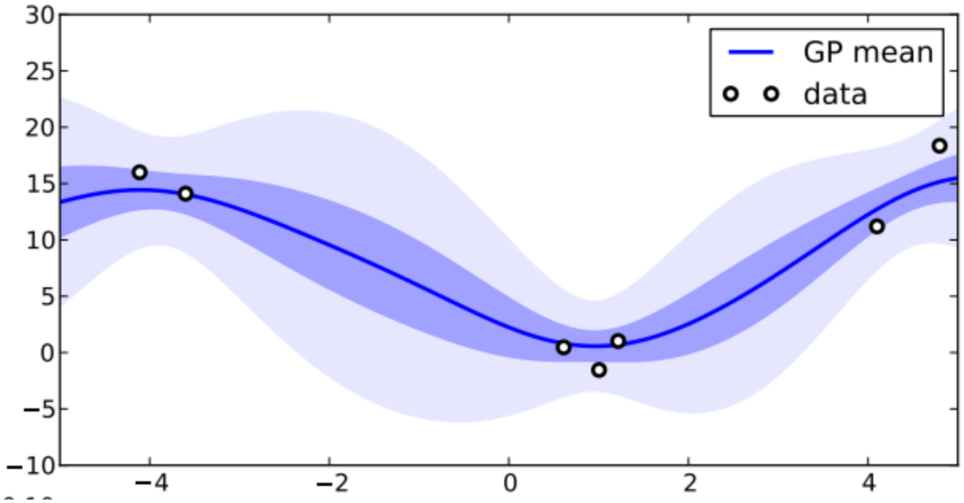
(Bergstra & Bengio 2012)

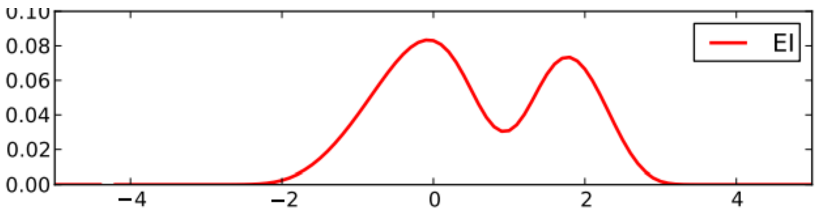
- Common approach: manual + grid search
- Grid search over hyperparameters: simple & wasteful
- Random search: simple & efficient
  - Independently sample each HP, e.g. l.rate~exp(U[log(.1),log(.0001)])
  - Each training trial is iid
  - If a HP is irrelevant grid search is wasteful
  - More convenient: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



Sequential Model-Based Optimization of Hyper-Parameters

- (Hutter et al JAIR 2009; Bergstra et al NIPS 2011; Thornton et al arXiv 2012; Snoek et al NIPS 2012)
- Iterate
  - Estimate P(valid. err | hyper-params config x, D)
  - choose optimistic x, e.g. $\max_x$ P(valid. err < current min. err | x)
  - train with config x, observe valid. err. v, D ← D U {(x,v)}





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

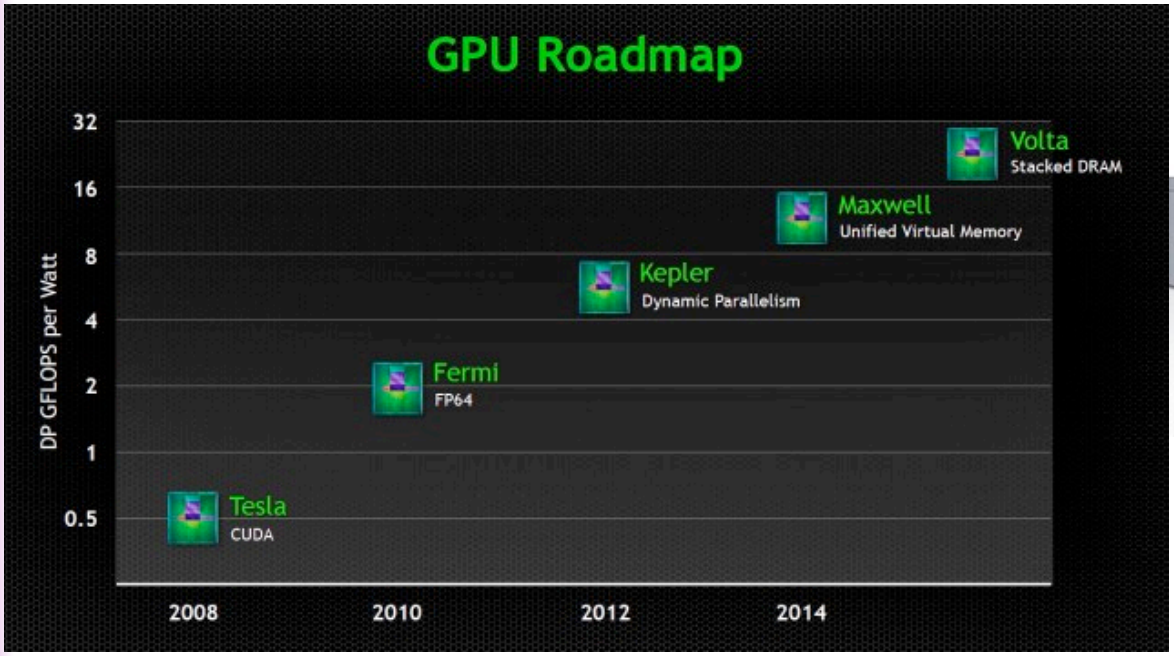
Deep Learning Challenges

(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

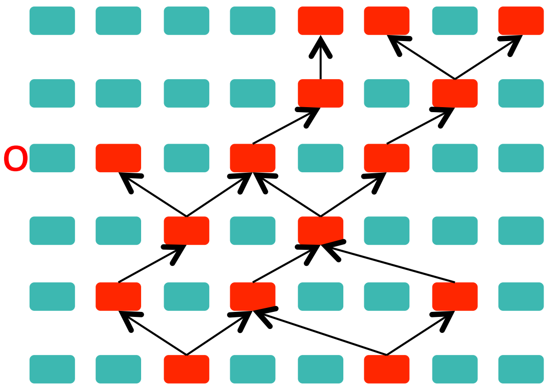
Challenge: Computational Scaling

- Recent breakthroughs in speech, object recognition and NLP hinged on faster computing, GPUs, and large datasets
- A 100-fold speedup is possible without waiting another 10 yrs?
  - Challenge of distributed training
  - Challenge of conditional computation

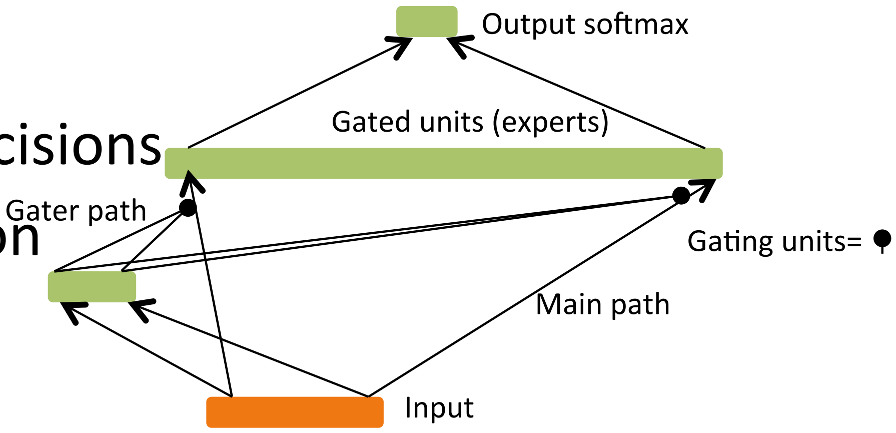


Conditional Computation: only visit a small fraction of parameters / example

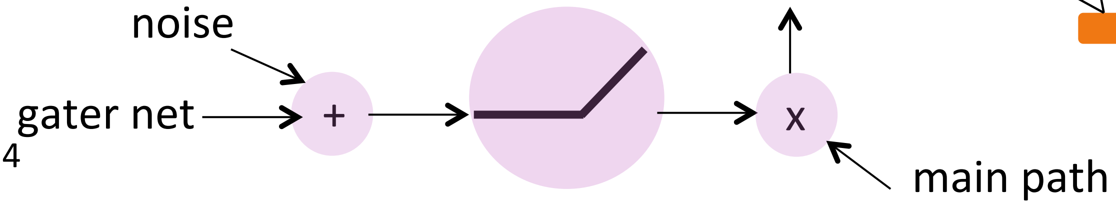
- Deep nets vs decision trees
- Hard mixtures of experts (Collobert, Bengio & Bengio 2002)
- Conditional computation for deep nets: sparse distributed gaters selecting combinatorial subsets of a deep net



- Challenges:
  - Credit assignment for hard decisions
  - Gated architectures exploration



- Noisy rectifiers work well



Distributed Training

- Minibatches
- Large minibatches + 2nd order & natural gradient methods
- Asynchronous SGD (Bengio et al 2003, Le et al ICML 2012, Dean et al NIPS 2012)
  - Bottleneck: sharing weights/updates among nodes, to avoid node-models to move too far from each other
- Ideas forward:
  - Low-resolution sharing only where needed
  - Specialized conditional computation (each computer specializes in updates to some cluster of gated experts, and prefers examples which trigger these experts)

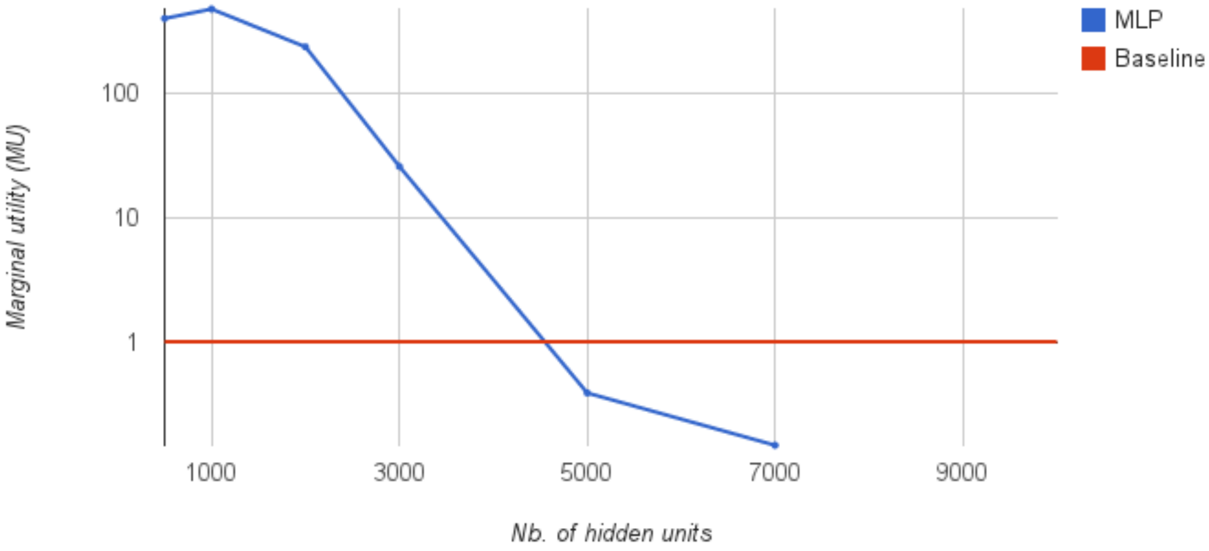
Deep Learning Challenges

(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

Optimization & Underfitting

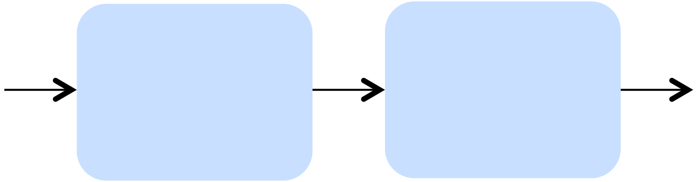
- On large datasets, major obstacle is underfitting
- **Marginal utility** of wider MLPs decreases quickly below memorization baseline



- Current limitations: local minima, ill-conditioning or else?

Guided Training, Intermediate Concepts

- In (Gulcehre & Bengio ICLR'2013) we set up a task that seems almost impossible to learn by shallow nets, deep nets, SVMs, trees, boosting etc
- Breaking the problem in two sub-problems and pre-training each module separately, then fine-tuning, nails it
- *Need prior knowledge to decompose the task*
- Guided pre-training allows to find much better solutions, escape effective local minima



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

Why Unsupervised Learning?

- Recent progress mostly in supervised DL
- ∃ 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

Basic Challenge with Probabilistic Models: marginalization

- Joint and marginal likelihoods involve intractable sums over configurations of random variables (inputs x, latent h, outputs y) e.g.

$$P(x) = \Sigma_h P(x,h)$$

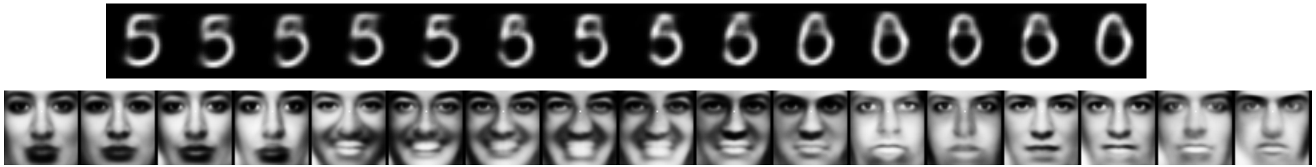
$$P(x,h) = e^{-\text{energy}(x,h)} / Z$$

$$Z = \Sigma_{x,h} e^{-\text{energy}(x,h)}$$

- MCMC methods can be used for these sums, by sampling from a chain of x's (or of (x,h) pairs) approximately from P(x,h)

Two Fundamental Problems with Probabilistic Models with Many Random Variables

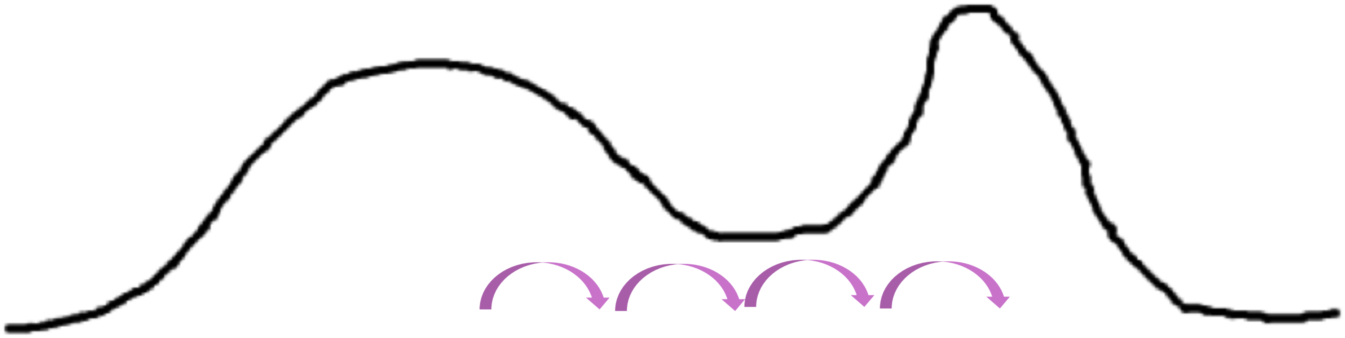
1. MCMC mixing between modes
   (manifold hypothesis)



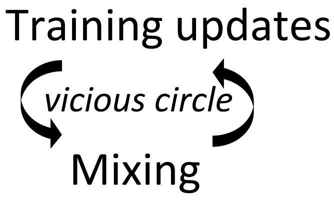
2. Many non-negligeable modes
   (both in posterior & joint distributions)

For gradient & inference: More difficult to mix with better trained models

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



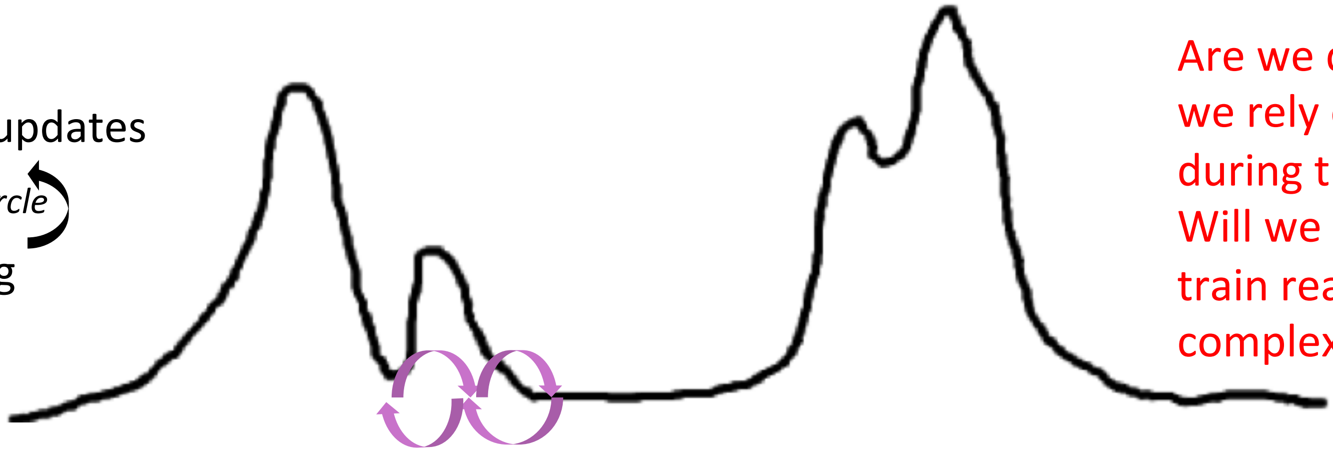
- Later on, hard to cross empty voids between modes



Training updates

vicious circle

Mixing

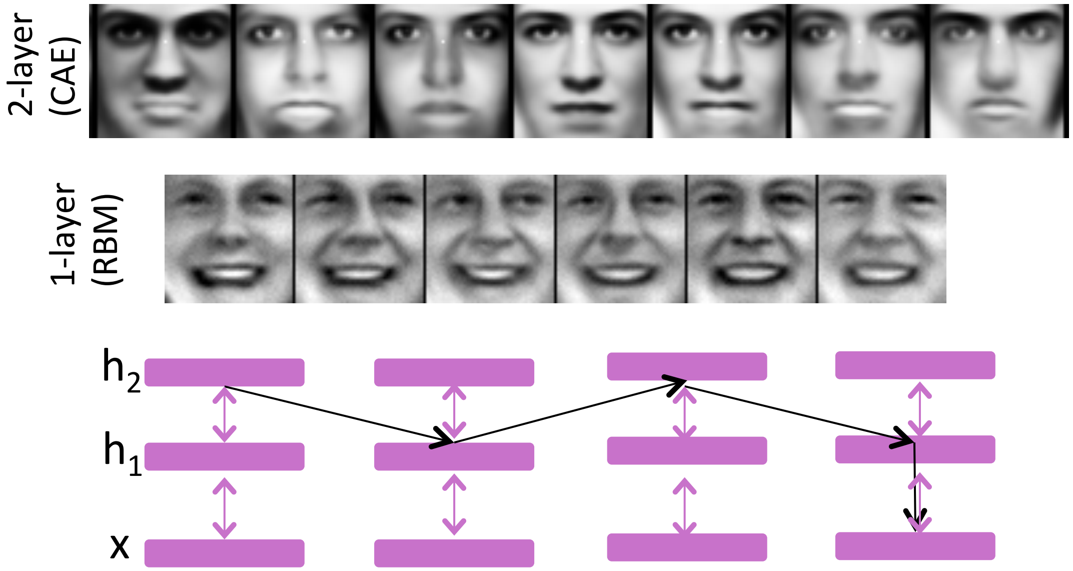


Are we doomed if we rely on MCMC during training? Will we be able to train really large & complex models?

Poor Mixing: Depth to the Rescue

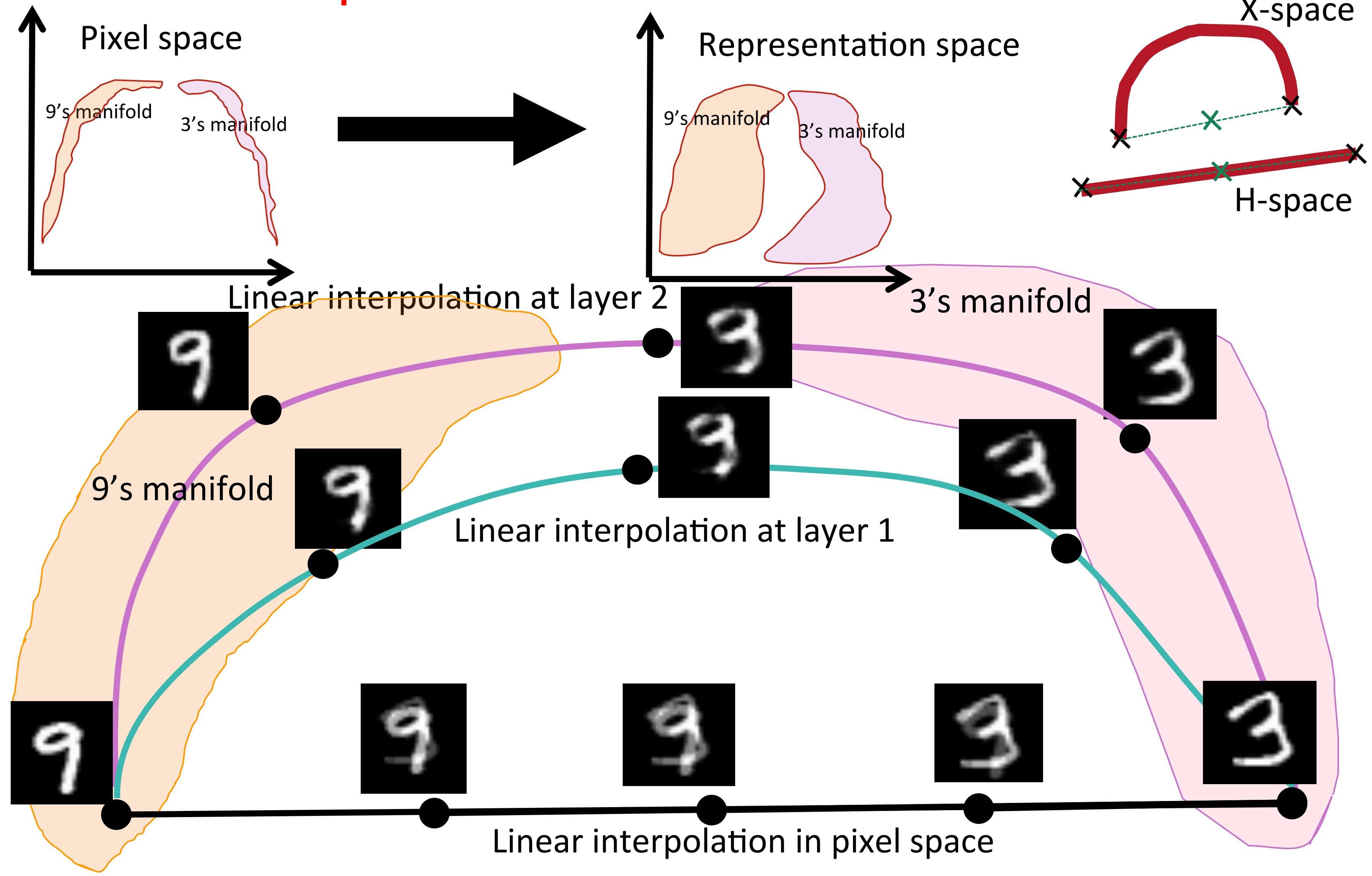
(Bengio et al ICML 2013)

- 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



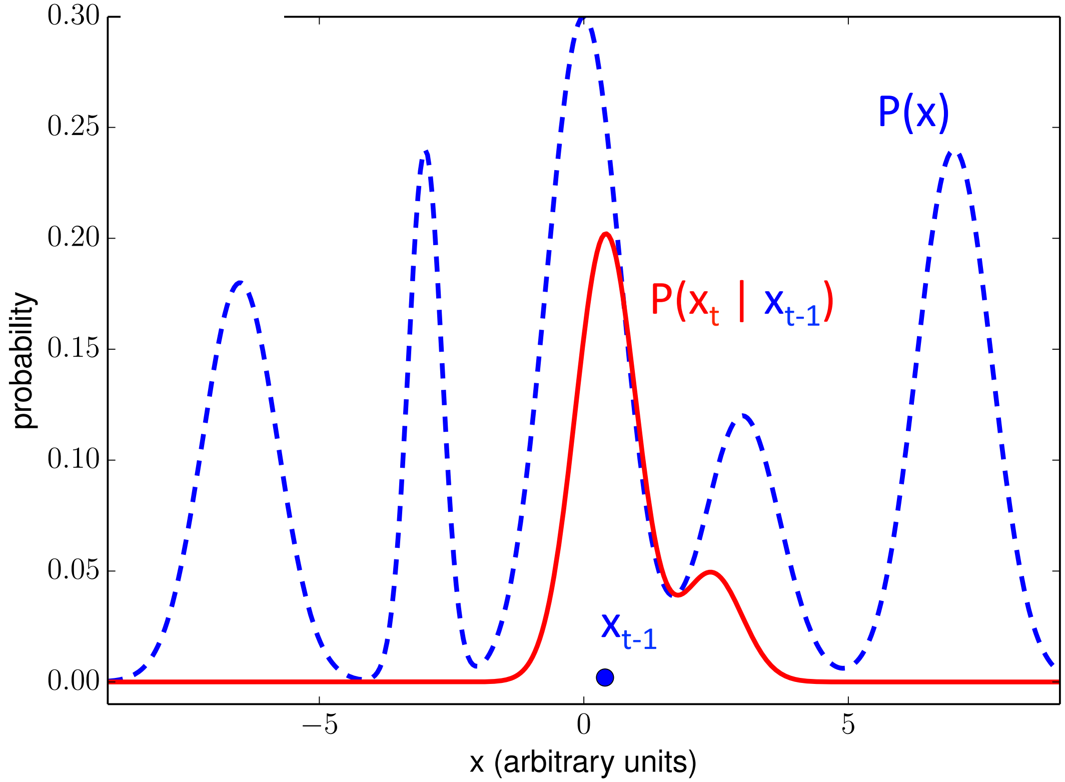
Space-Filling in Representation-Space

- **Deeper representations ➔ abstractions ➔ disentangling**
- **Manifolds are expanded and flattened**



Many Modes Challenge: Instead of Learning P(x) directly, Learn Markov chain operator $P(x_t \mid x_{t-1})$

- P(x) may have many modes, making the normalization constant intractable, and MCMC approximations poor
- Partition fn of $P(x_t \mid x_{t-1})$ much simpler because most of the time a local move, might even be well approximated by unimodal



Deep Learning Challenges

(Bengio, arxiv 1305.0445 Deep Learning of representations: looking forward)

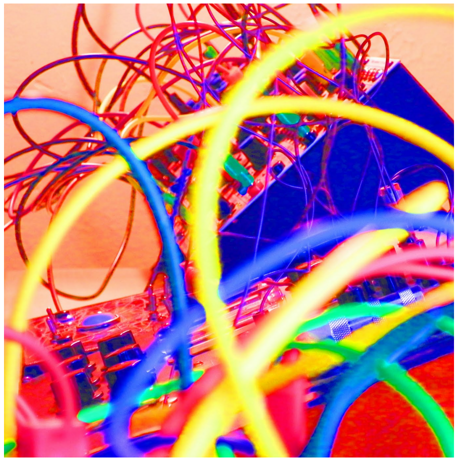
- Computational Scaling
- Optimization & Underfitting
- Intractable Marginalization, Approximate Inference & Sampling
- Disentangling Factors of Variation
- Reasoning & One-Shot Learning of Facts

Disentangling the Underlying Factors

- How could a learner disentangle the unknown underlying factors of variation?
  - Statistical structure present in the data
  - Hints = priors

- Good disentangling →

  avoid the curse of dimensionality

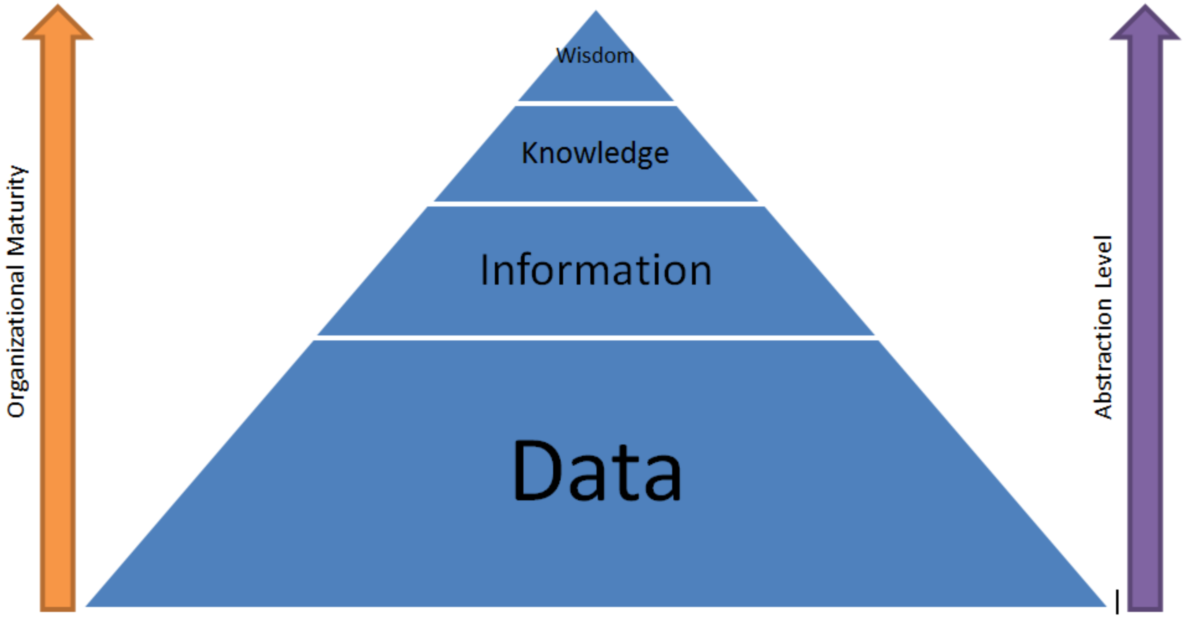


Broad Priors as Hints to Disentangle the Factors of Variation

- *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)

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 & 2014 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...

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

LISA team: **Merci! Questions?**