Scaling up Deep Learning

KDD'2014 Tutorial

Yoshua Bengio

August 24, 2014

Université de Montréal

LISA

Ultimate Goal

- Understand the principles giving rise to intelligence

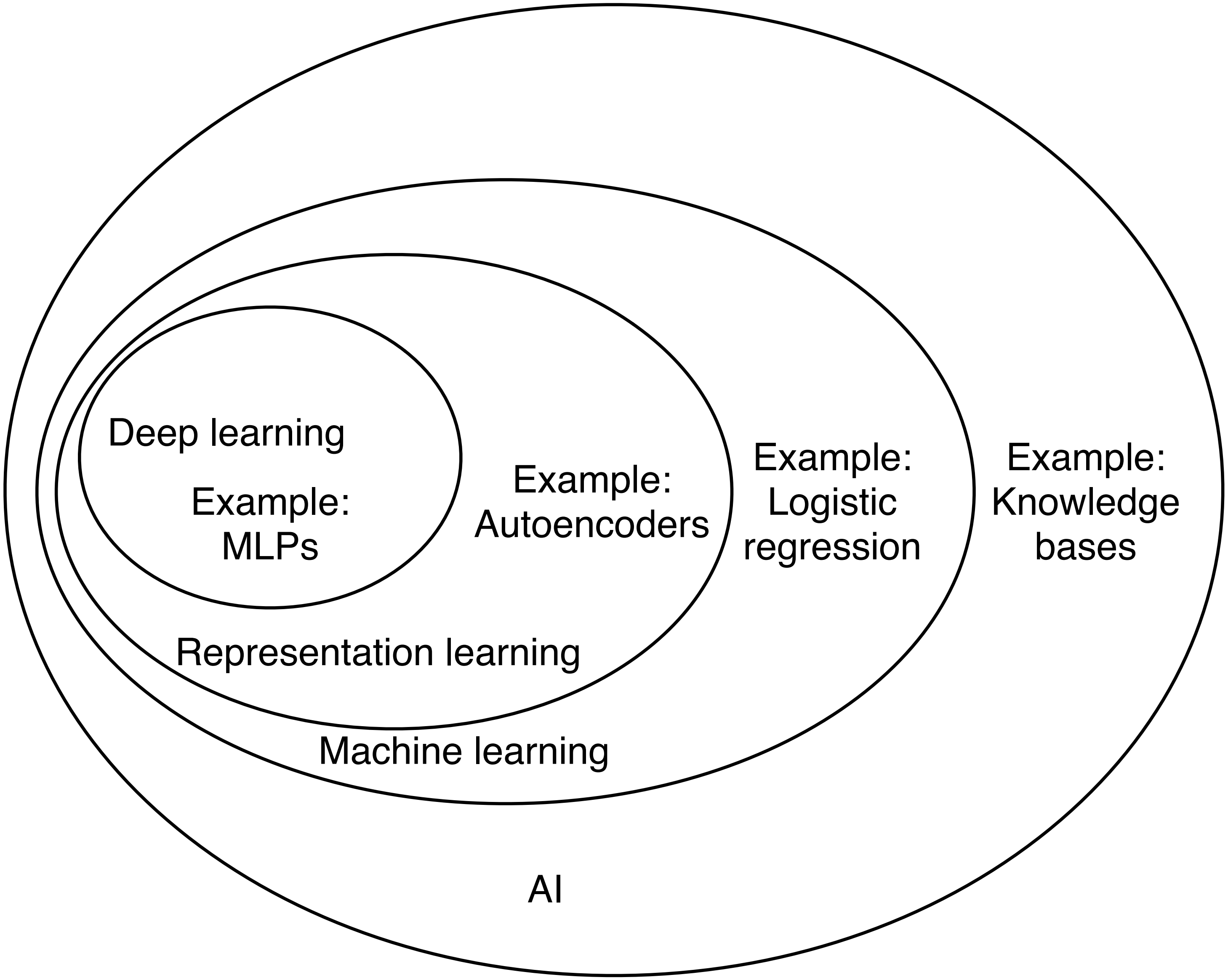
Focus

- **Learning:** mathematical and computational principles allowing one to learn from examples in order to acquire knowledge

Breakthrough

- **Deep Learning**: machine learning algorithms inspired by brains, based on learning multiple levels of representation / abstraction.

Deep Learning within ML & AI



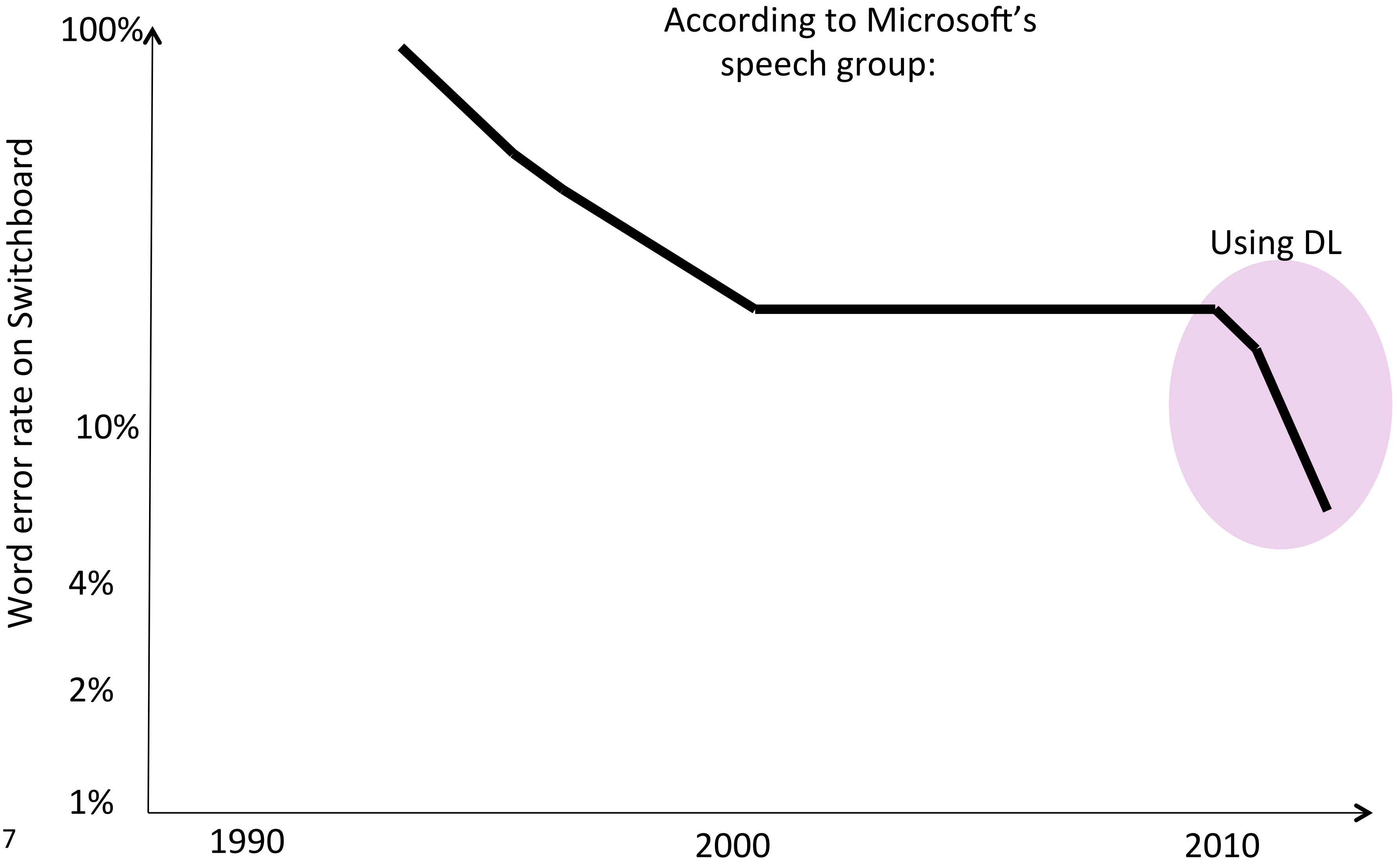
Impact

Deep learning has revolutionized

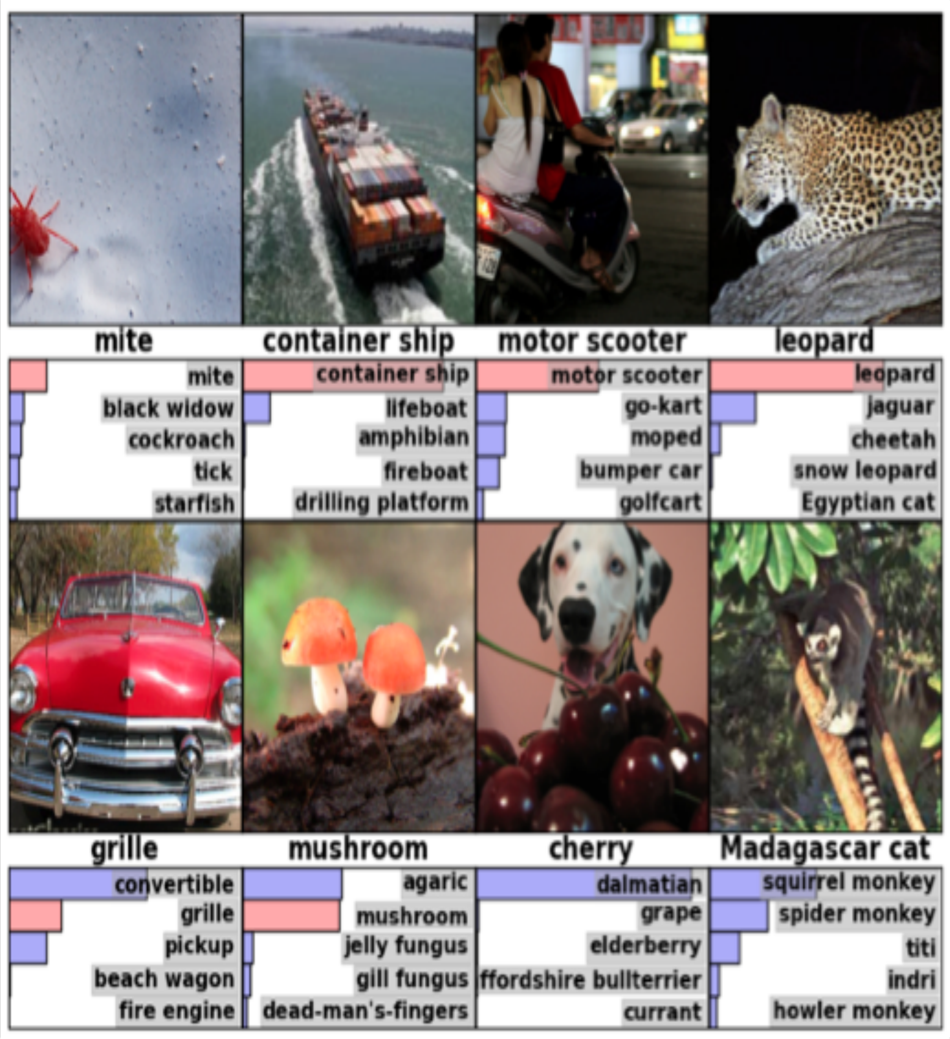
- Speech recognition
- Object recognition

More coming, including other areas of computer vision, NLP, dialogue, reinforcement learning…

The dramatic impact of Deep Learning on Speech Recognition



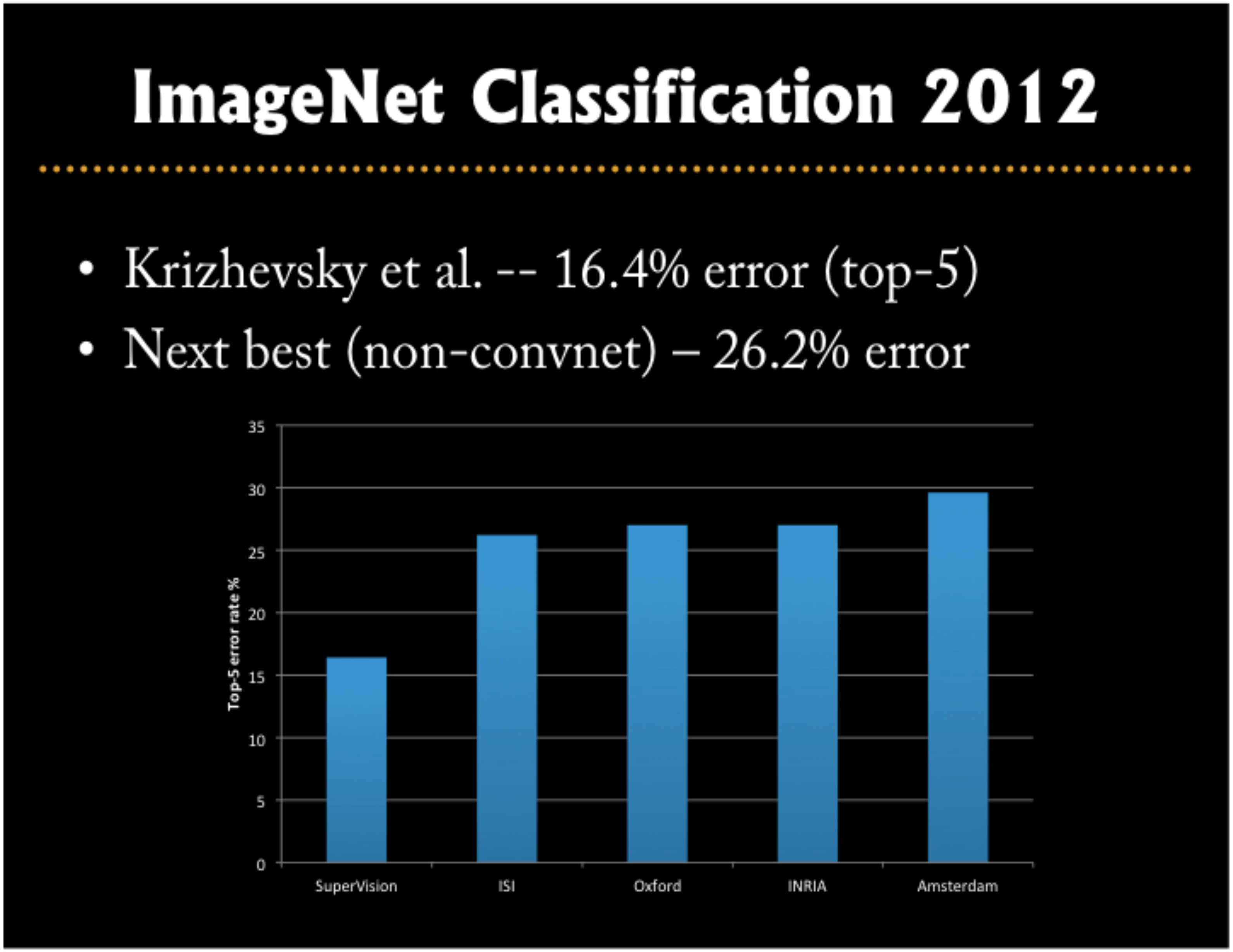
Object Recognition Breakthrough



ImagetNet Breakthrough

- Achieves state-of-the-art on many object recognition tasks.

See: deeplearning.cs.toronto.edu



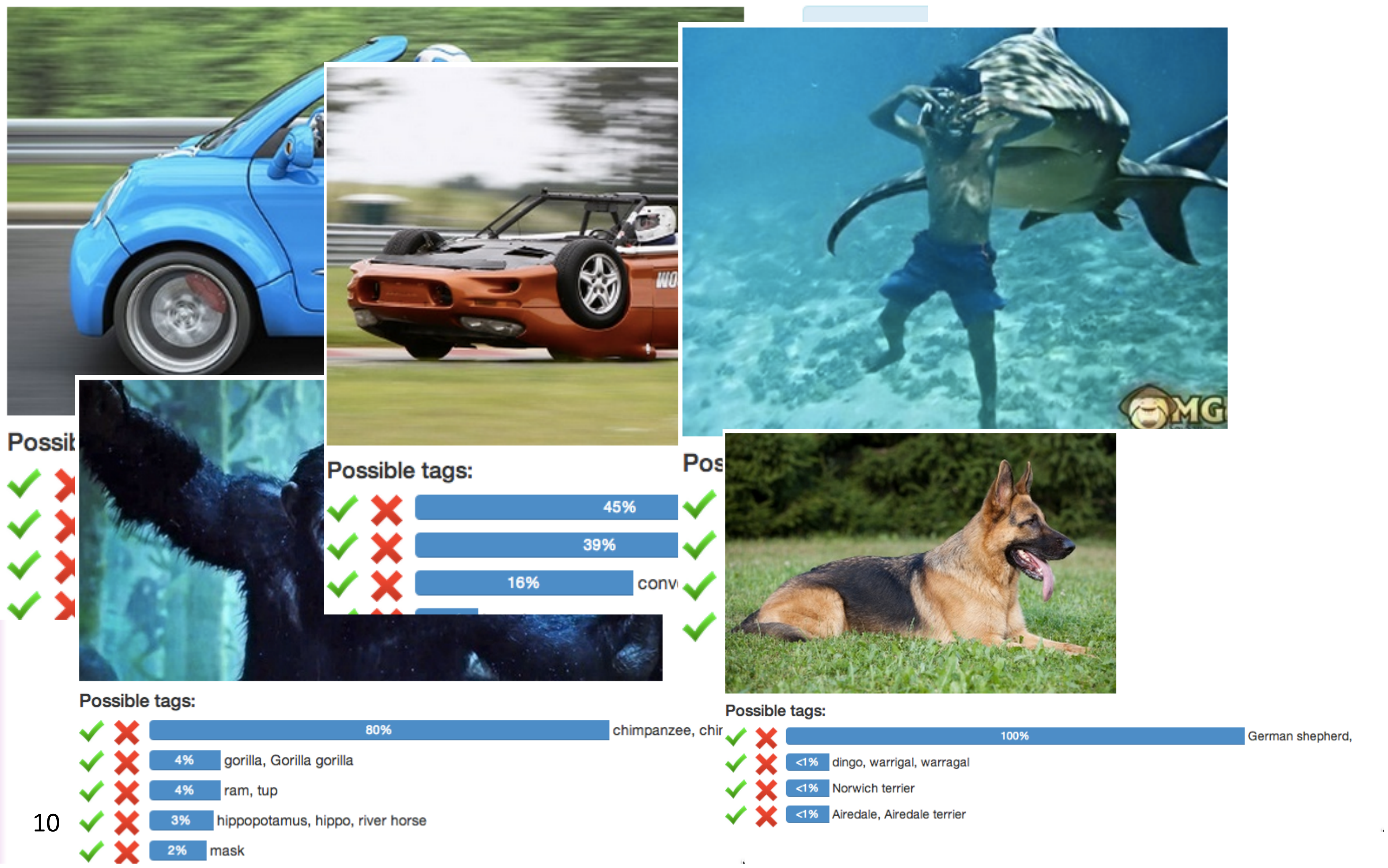
ImageNet Classification 2012

- Krizhevsky et al. -- 16.4% error (top-5)
- Next best (non-convnet) – 26.2% error

Slide from Rob Fergus, NIPS tutorial, 2012

Object Recognition Works

- Try it at http://deeplearning.cs.toronto.edu



Possib

Possible tags:

✓	✗	45%	
✓	✗	39%	
✓	✗	16%	conv

Pos

Possible tags:

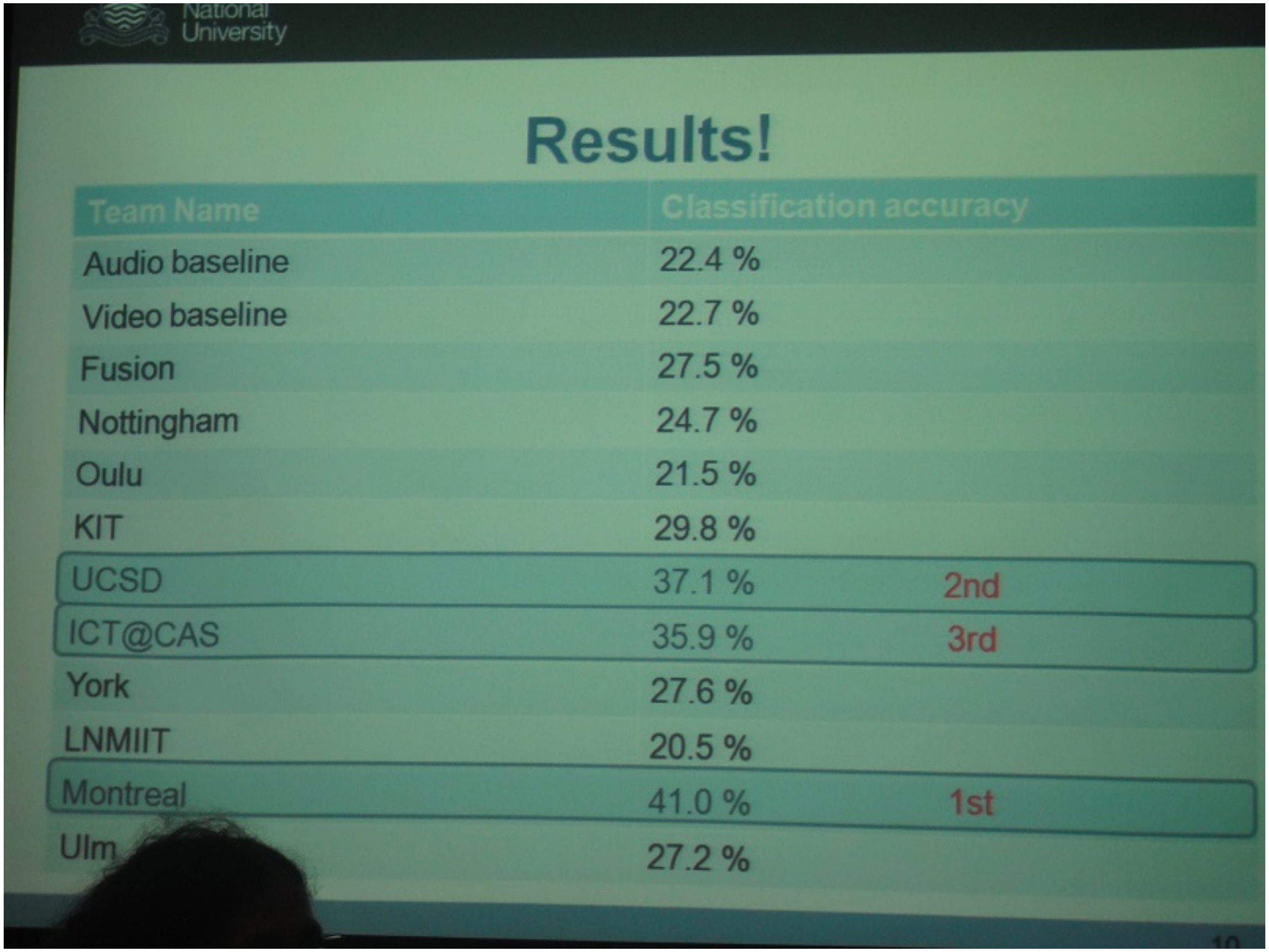
✓	✗	80%	chimpanzee, chir
✓	✗	4%	gorilla, Gorilla gorilla
✓	✗	4%	ram, tup
✓	✗	3%	hippopotamus, hippo, river horse
✓	✗	2%	mask

Possible tags:

✓	✗	100%	German shepherd,
✓	✗	<1%	dingo, warrigal, warragal
✓	✗	<1%	Norwich terrier
✓	✗	<1%	Airedale, Airedale terrier

Montreal Deep Nets Win Emotion Recognition in the Wild Challenge

Predict emotional expression from video (using images + audio)

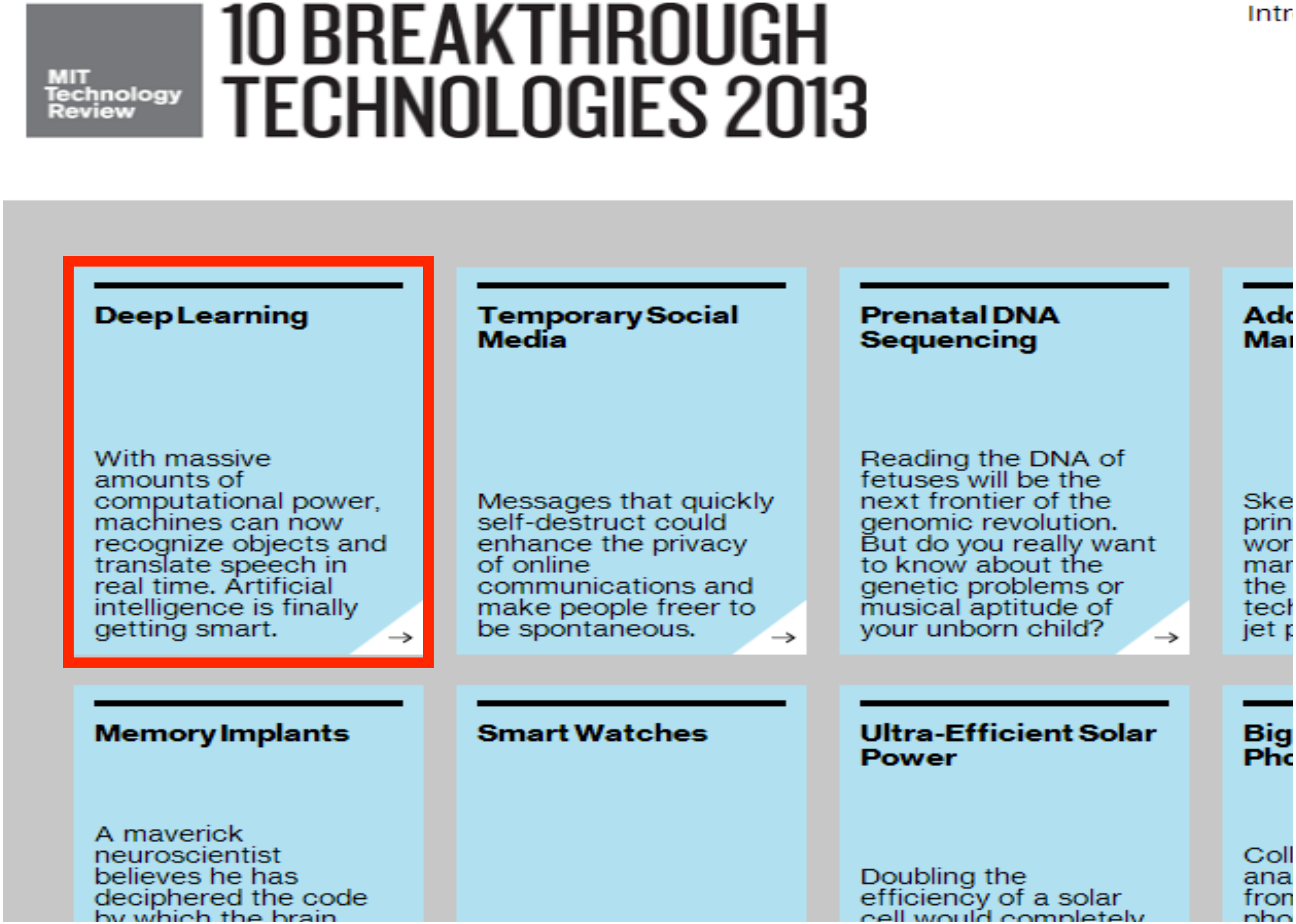


National University

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 %	

Dec. 9, 2013



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?

Ad
Ma

Ske
prin
wor
mar
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
fron
pho

Deep Learning in the News

The New York Tim

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

e Hired to Make A.I. a Reality

WEEKS

EXCLUSIVE

Facebook, Google in 'Deep Learning' Arms Race

Yann LeCun, an NYU artificial intelligence researcher who now works for Facebook. *Photo: Josh Valcarcel/WIRED*

NEWS BULLETIN

Google Beat Facebook for DeepMind

Google Acquires Artificial Intelligence Startup DeepMind For More Than $500M

Posted Jan 26, 2014 by Catherine Shu (@catherineshu)

Challenges

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

- **Scaling up Unsupervised Learning & Structured outputs**
  - ***Intractable* computations with latent variables**
  - **Key to more adaptable models, exploiting tons of unlabeled data, and complex output decisions**
- **Scaling up Computation**
- **Scaling up Numerical Optimization**

Potential Outcome: AI

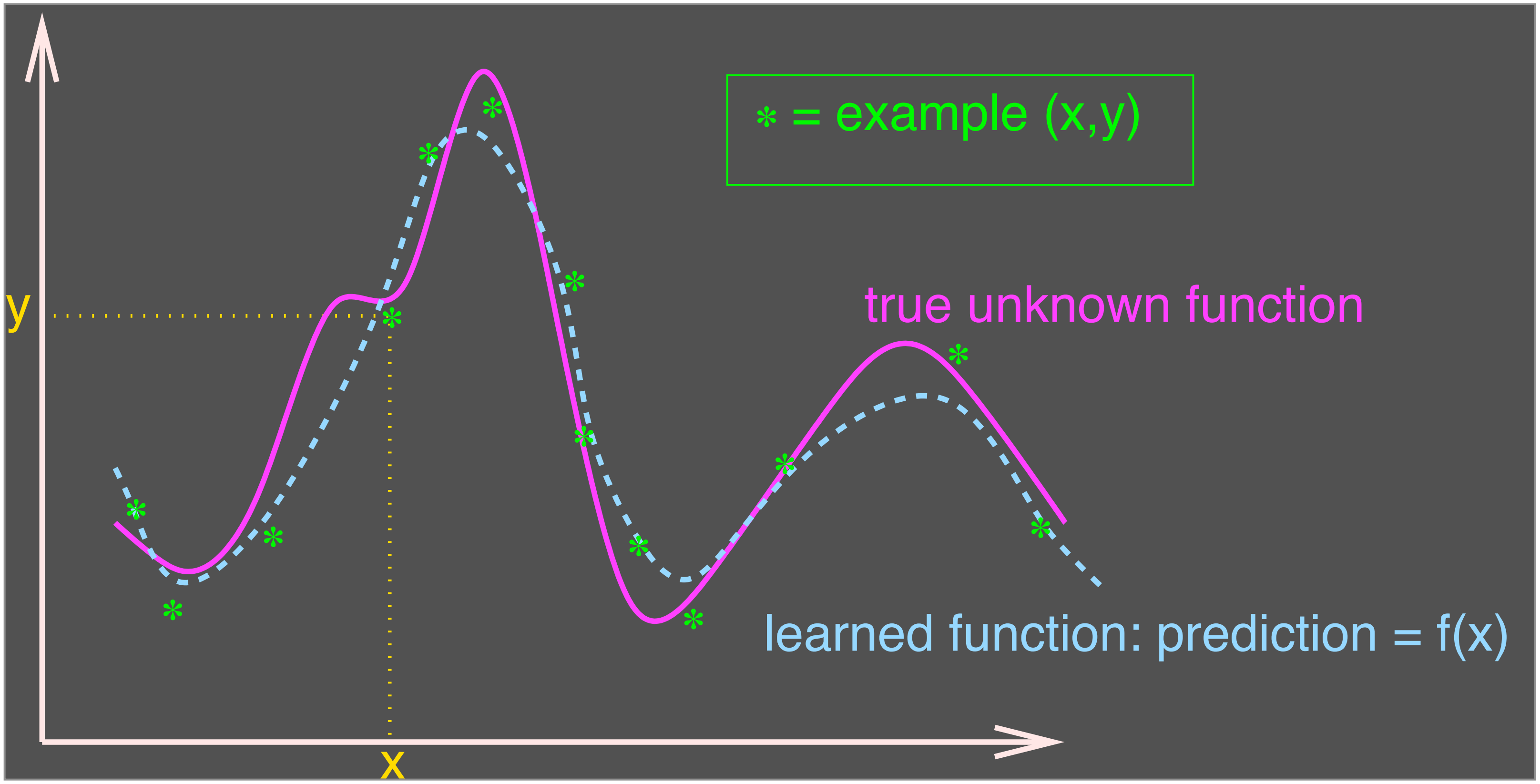
- Computers that can
  - see and hear
  - understand natural language
  - understand human behavior
- Better understanding of human & animal intelligence
- Personnal assistants, self-driving cars...

Technical Goals Hierarchy

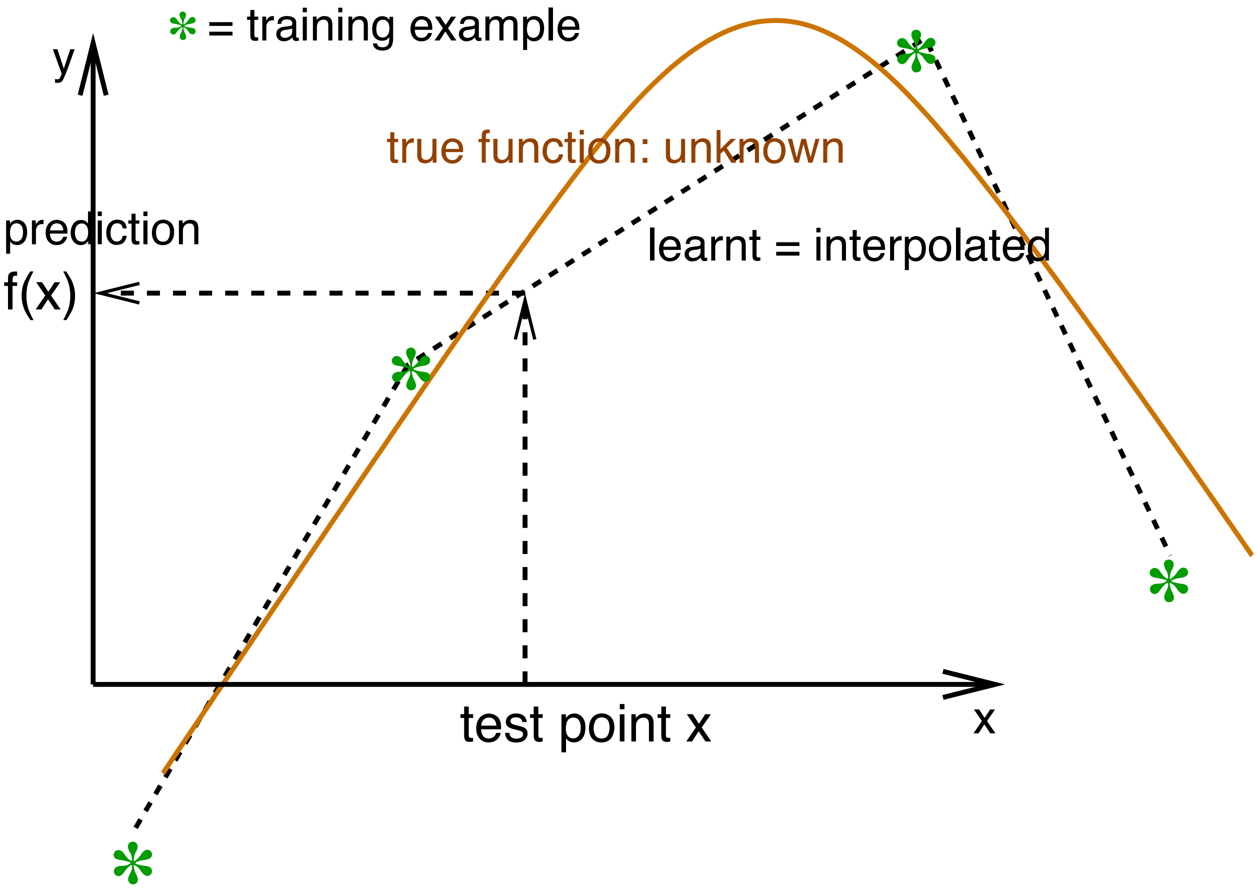
To reach AI:

- Needs **knowledge**
- Needs **learning**
  (involves priors + ***optimization***/*search* ***+ efficient computation***)
- 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)

Easy Learning



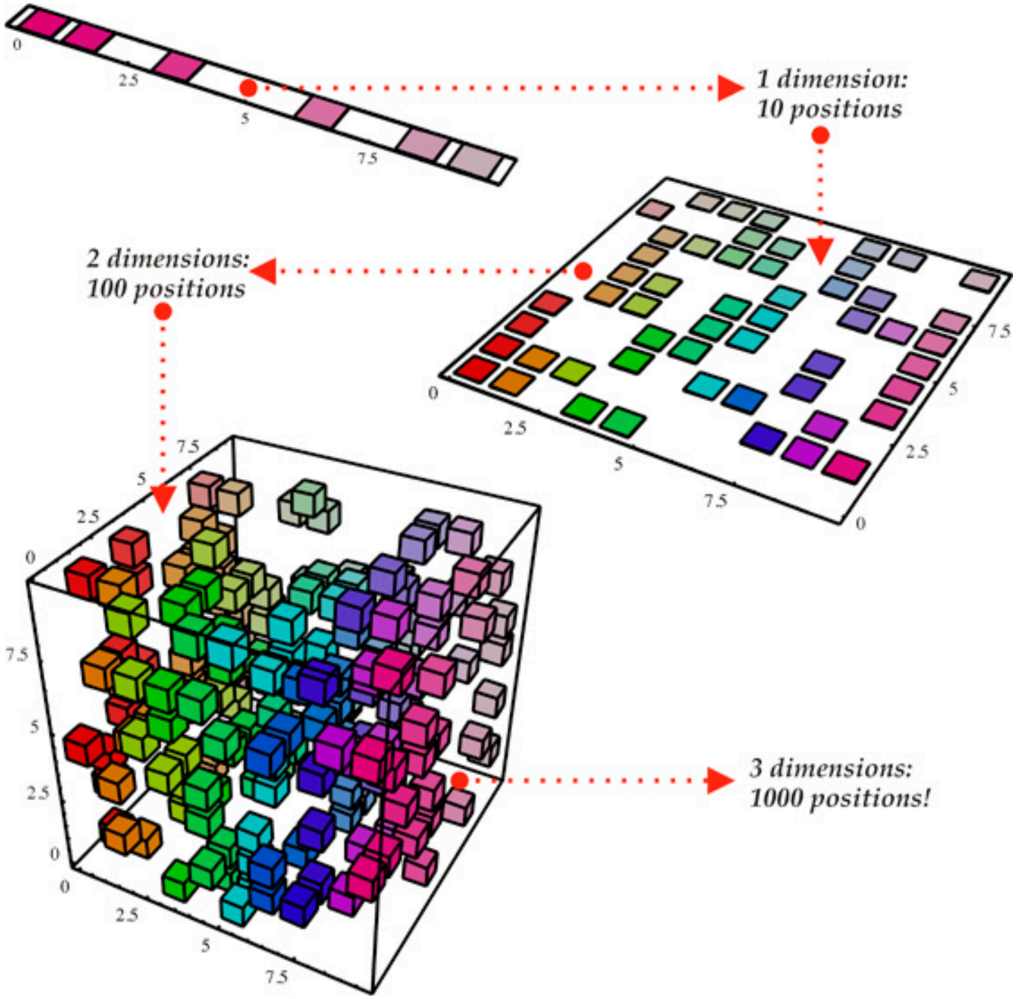
Local Smoothness Prior: Locally Capture the Variations



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

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

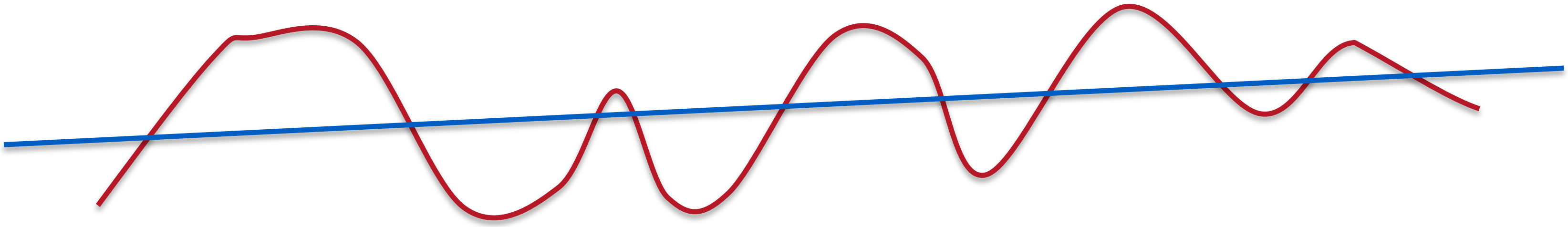
Classical solution: hope for a smooth enough target function, or make it smooth by handcrafting good features / kernel



Not Dimensionality so much as Number of Variations

(Bengio, Dellalleau & Le Roux 2007)

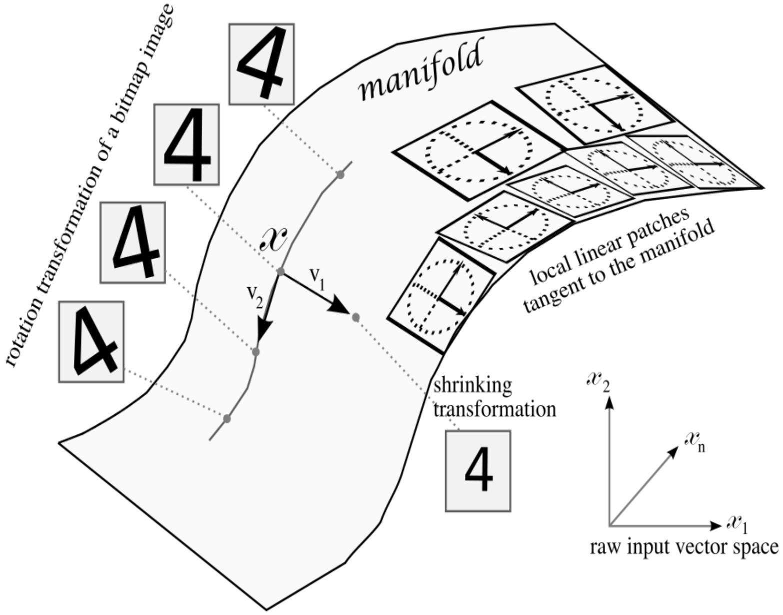
- Theorem: Gaussian kernel machines need at least *k* examples to learn a function that has *2k* zero-crossings along some line

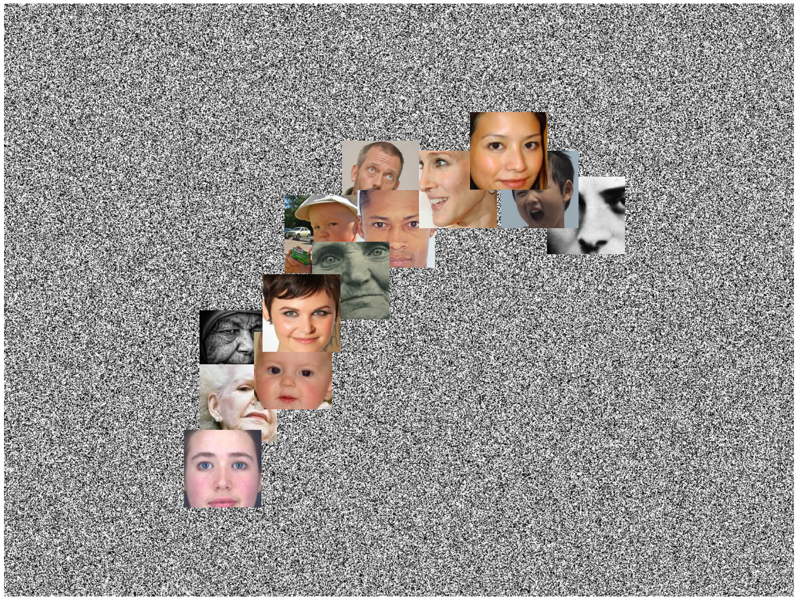


- Theorem: For a Gaussian kernel machine to learn some maximally varying functions over *d* inputs requires O(2^d) examples

For AI Tasks: Manifold structure

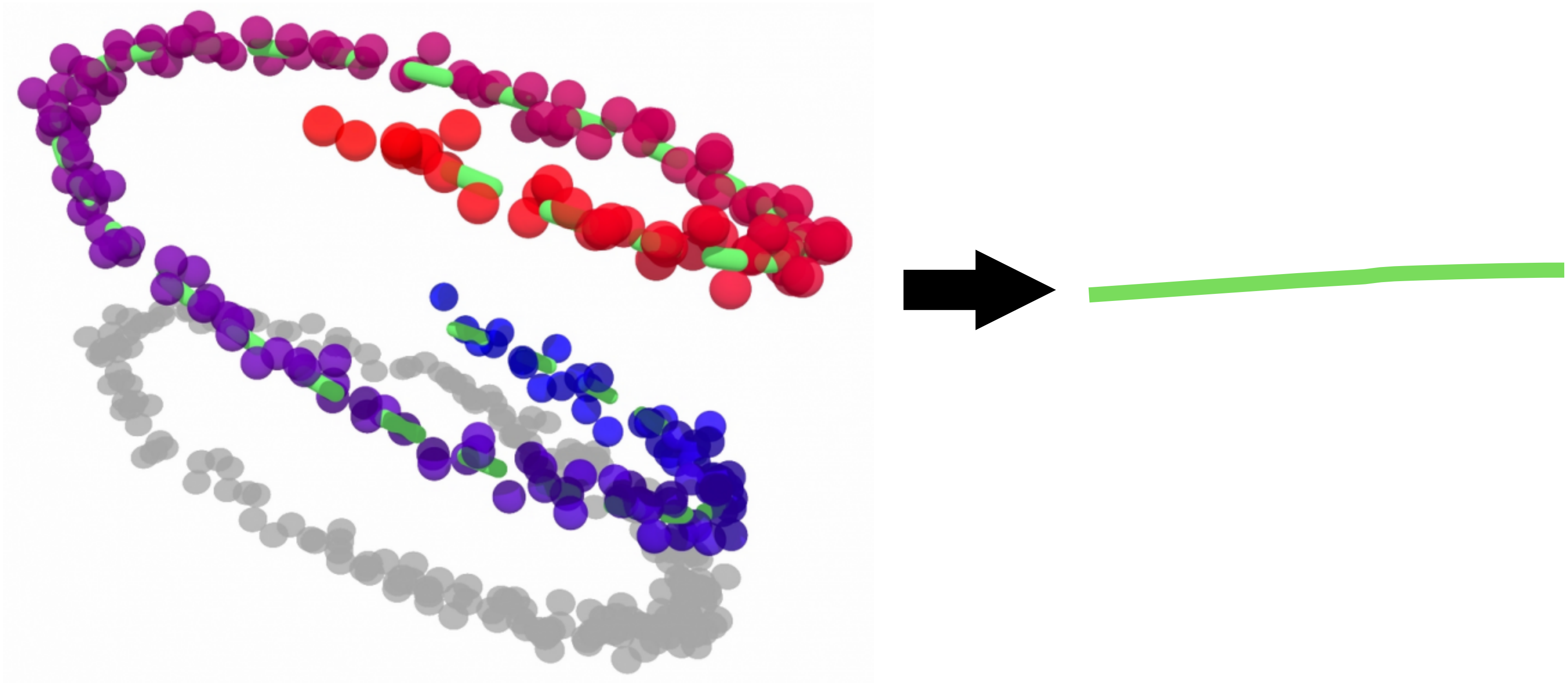
- examples **concentrate** near a lower dimensional “manifold
- **Evidence: most input configurations are unlikely**



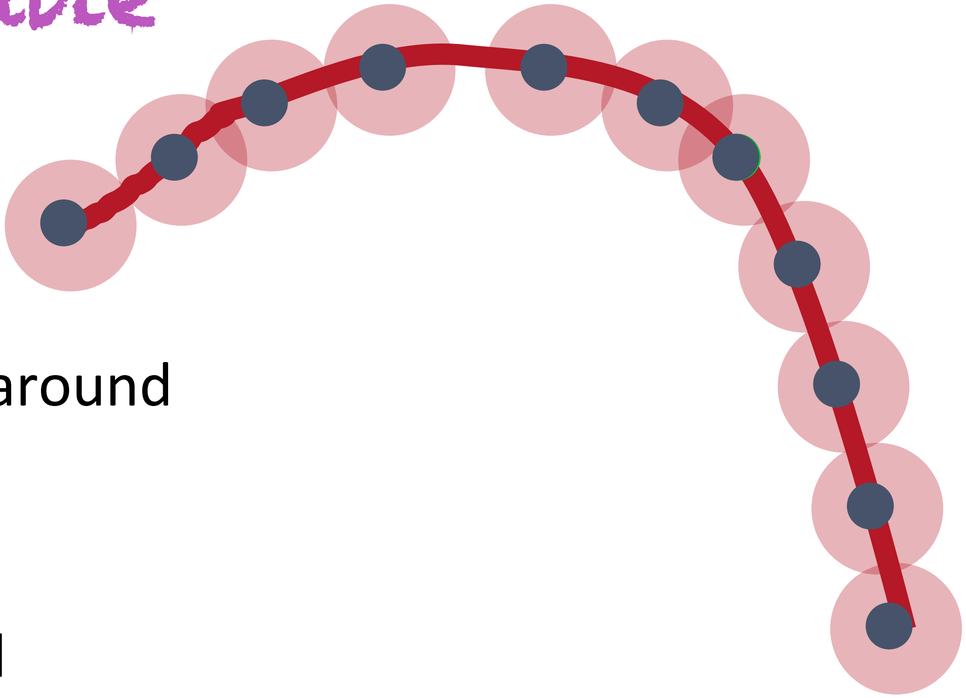


Geometrical view on machine learning

- Generalization: guessing **where** *probability* mass concentrates
- Challenge: the curse of dimensionality (exponentially many configurations of the variables to consider)
- Representation Learning: mapping to a new space, unfolding



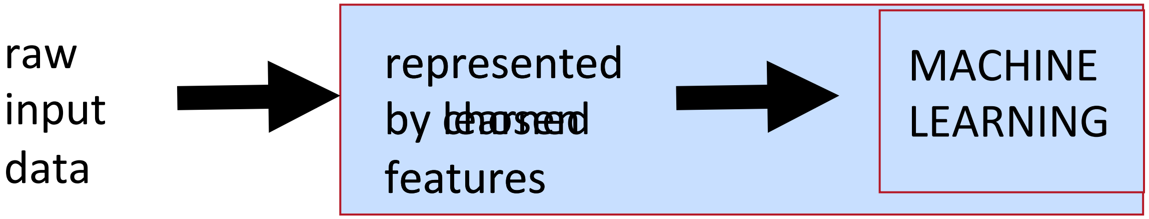
Putting Probability Mass where Structure is Plausible



- Empirical distribution: mass at training examples
- Smoothness: spread mass around
- Insufficient
- Guess some 'structure' and generalize accordingly
- Equivalent to guessing a good representation in which distance is meaningful and relationships are simple, linear

Representation Learning

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



raw input data

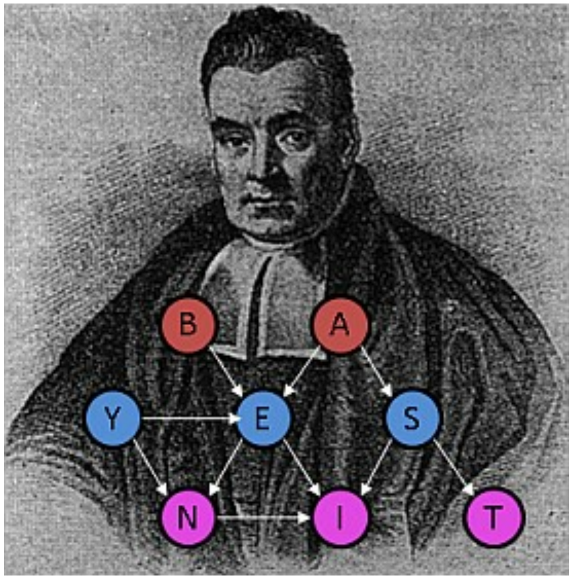
represented by learned chosen features

MACHINE LEARNING

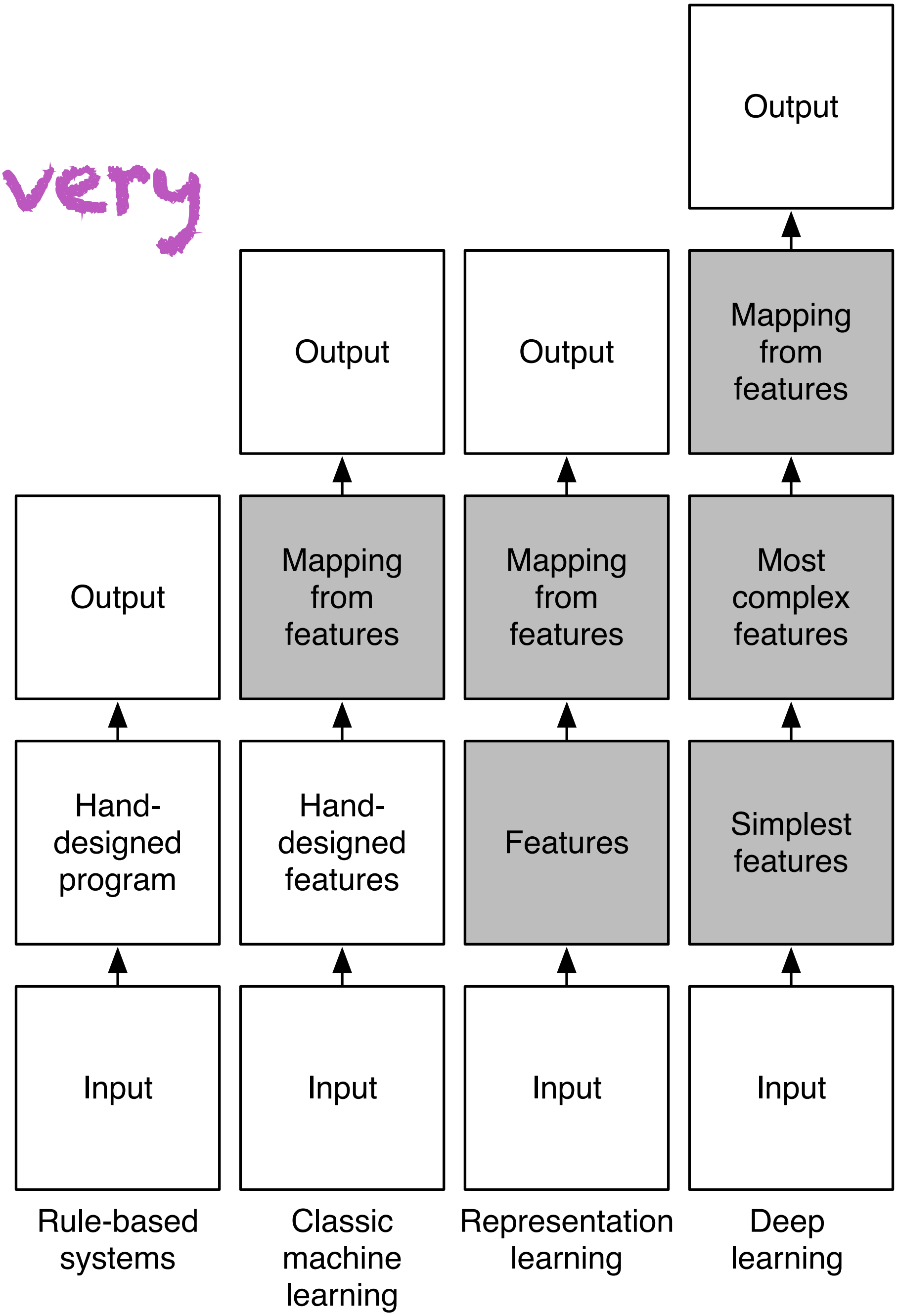
- Handcrafting features vs learning them

- Good representation?
- guesses

  the features / factors / causes



Automating Feature Discovery



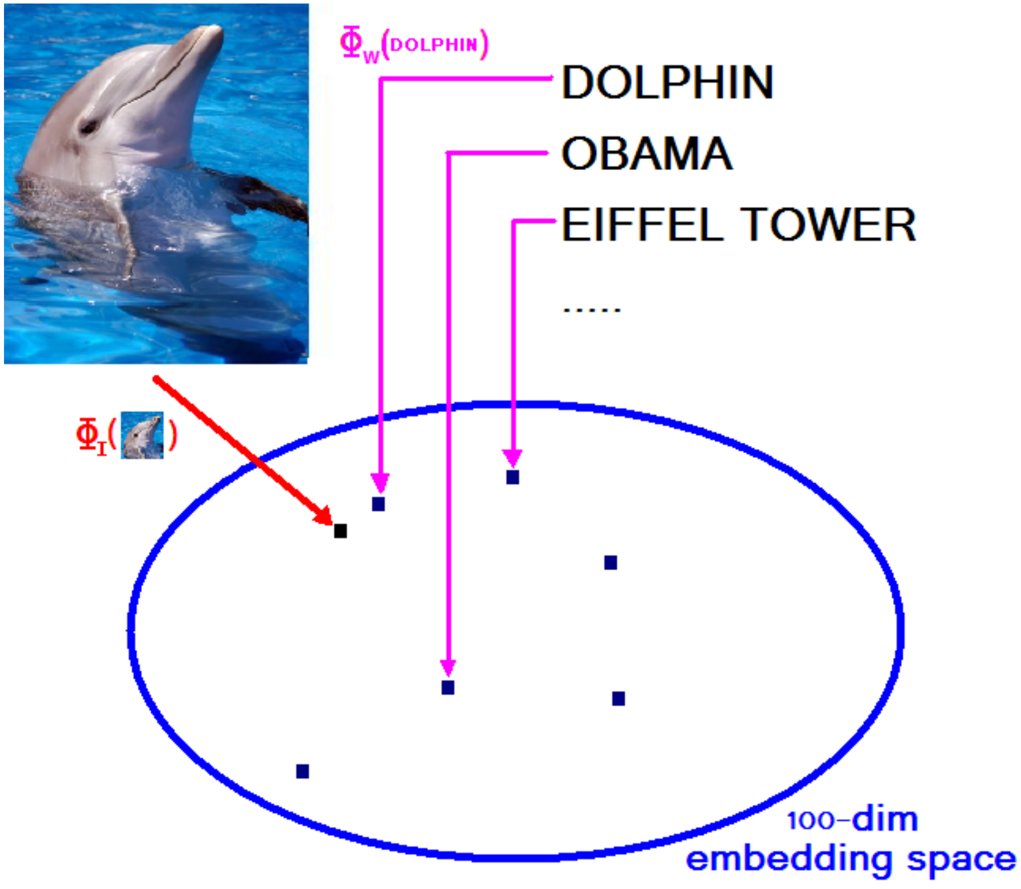
Google Image Search:

Different object types represented in the same space

Google:

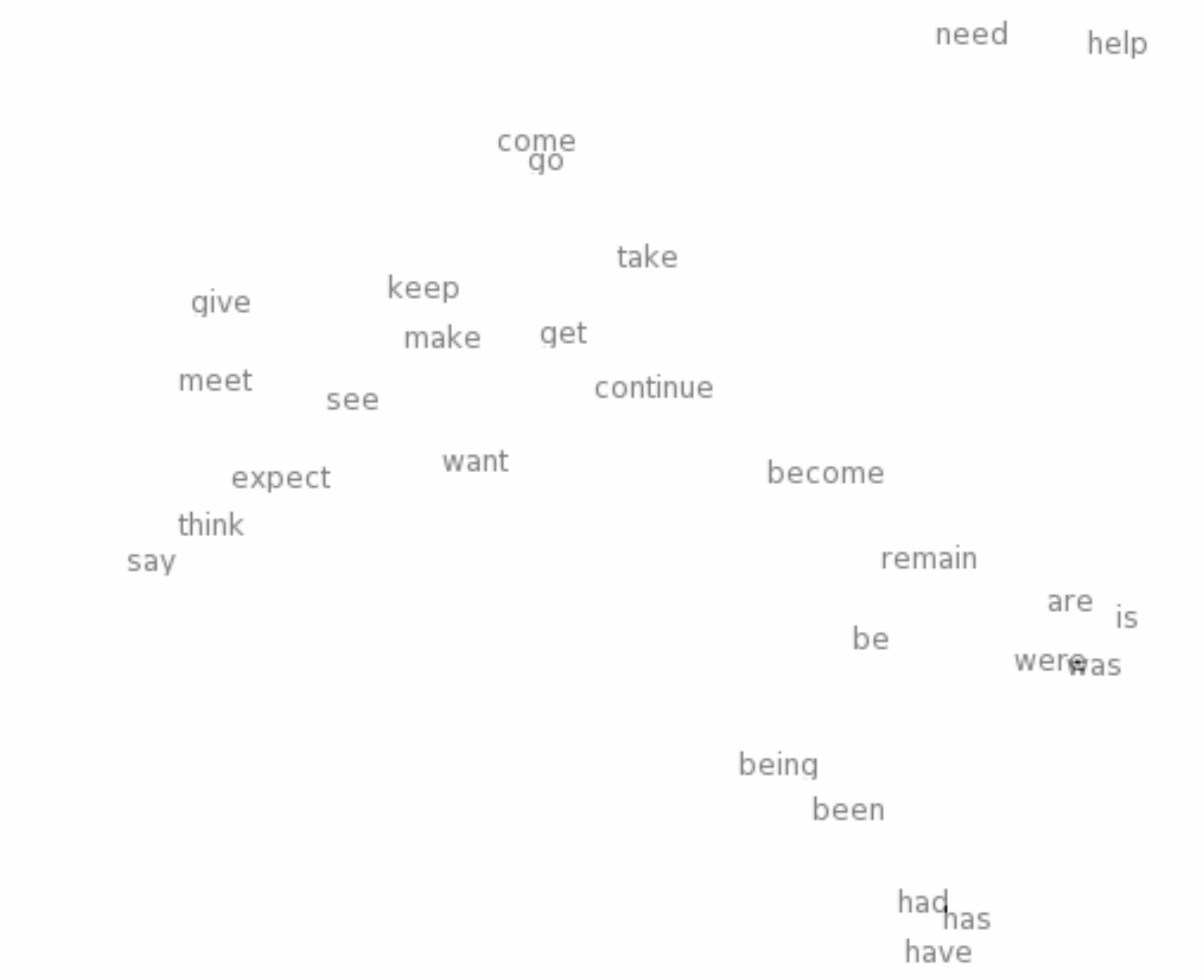
S. Bengio, J. Weston & N. Usunier

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



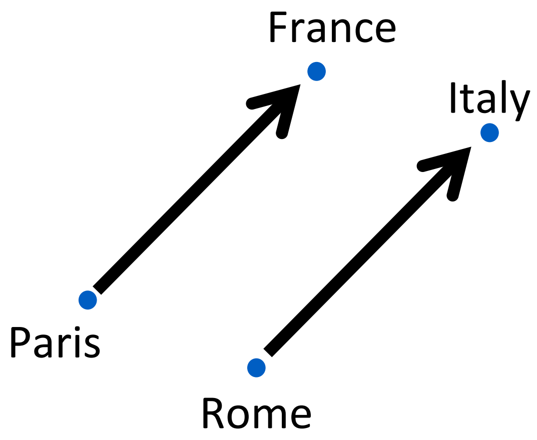
Learn $\Phi_I(\cdot)$ *and* $\Phi_W(\cdot)$ *to optimize precision@k.*

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



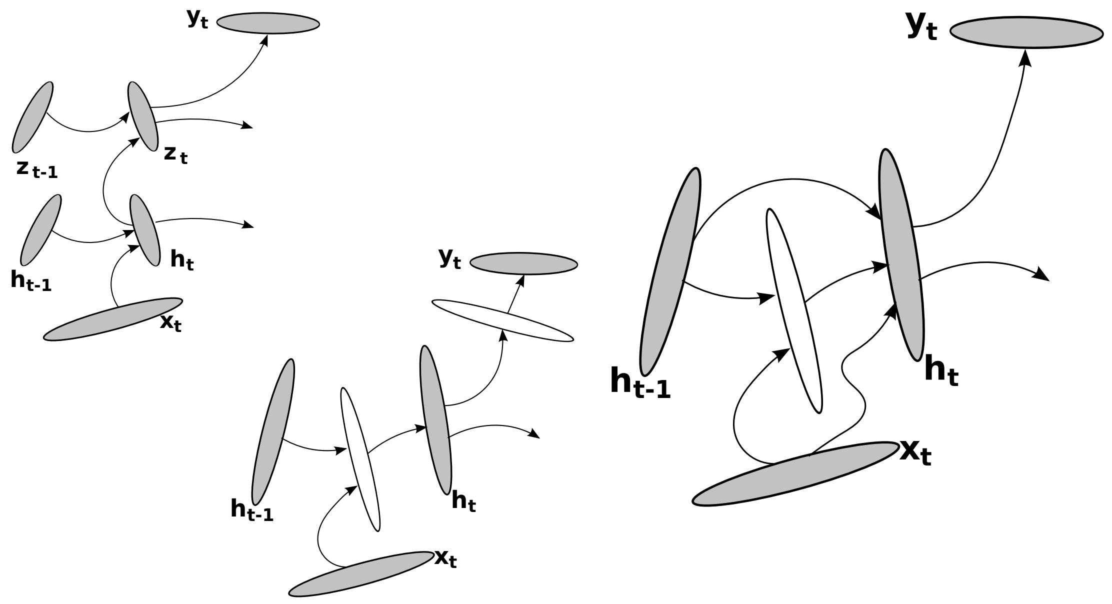
Breakthroughs in Machine Translation

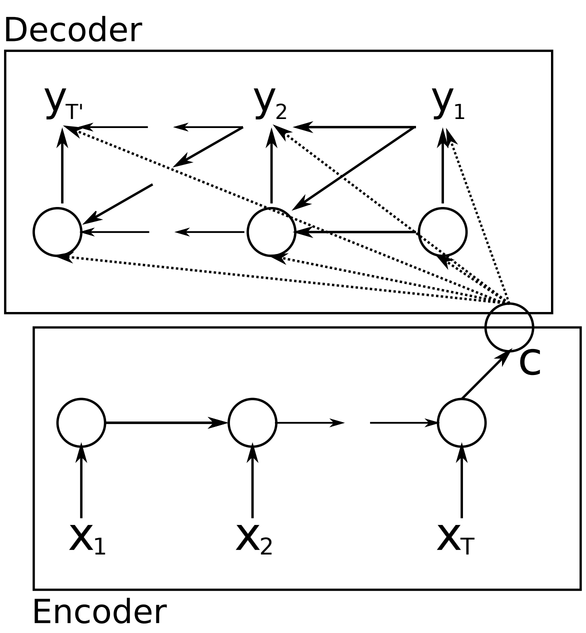
- (Cho et al, EMNLP 2014) Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation

Almost 2 BLEU points improvement for English-French

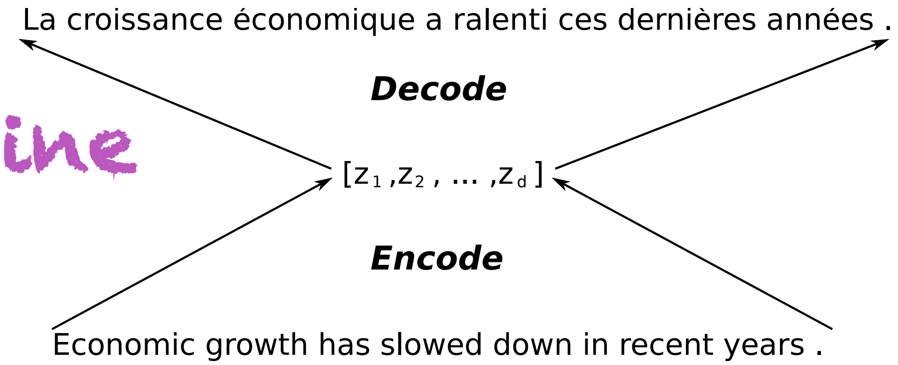
- (Devlin et al, ACL 2014) Fast and Robust Neural Network Joint Models for Statistical Machine Translation

Best paper award, 6 BLEU points improvement for Arabic-English





Encoder-Decoder Framework for Machine Translation


La croissance économique a ralenti ces dernières années .

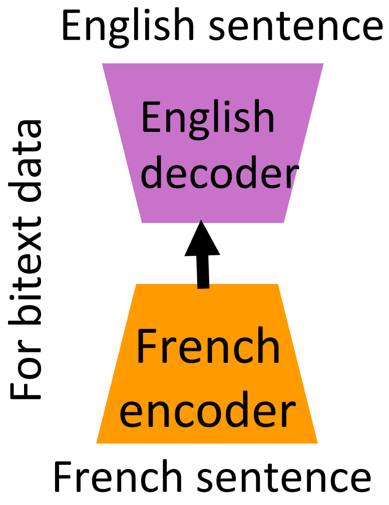
Decode

$[z_1, z_2, \ldots, z_d]$

Encode

Economic growth has slowed down in recent years .

- One encoder and one decoder per language
- Universal intermediate representation
- Encode(French) → Decode(English) = translation model
- Encode(English) → Decode(English) = language model
- Parametrization grows linearly with # languages, not quadratic

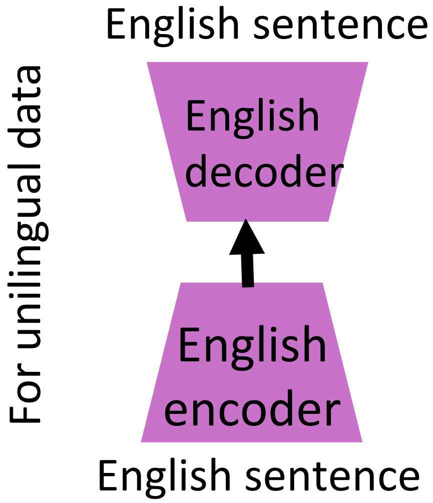

English sentence

English decoder

French encoder

French sentence

For bitext data


English sentence

English decoder

English encoder

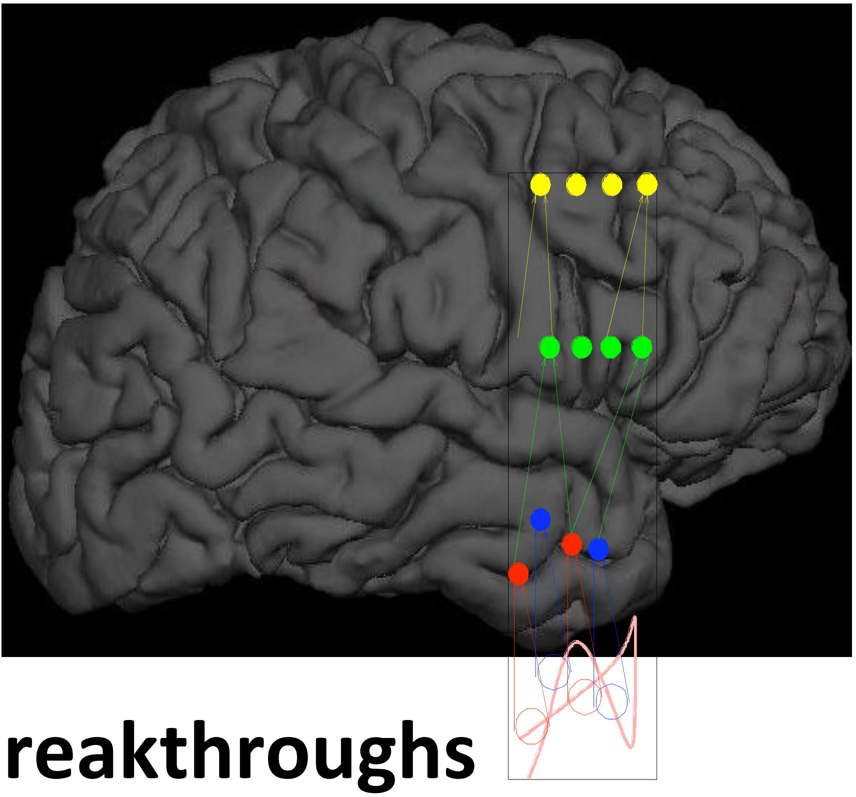
English sentence

For unilingual data

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

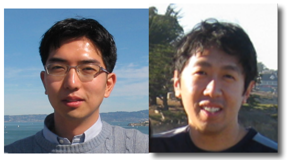


Biologically inspired learning

- Brain has a deep architecture
- Cortex seems to have a generic learning algorithm
- **Humans first learn simpler concepts and compose them**

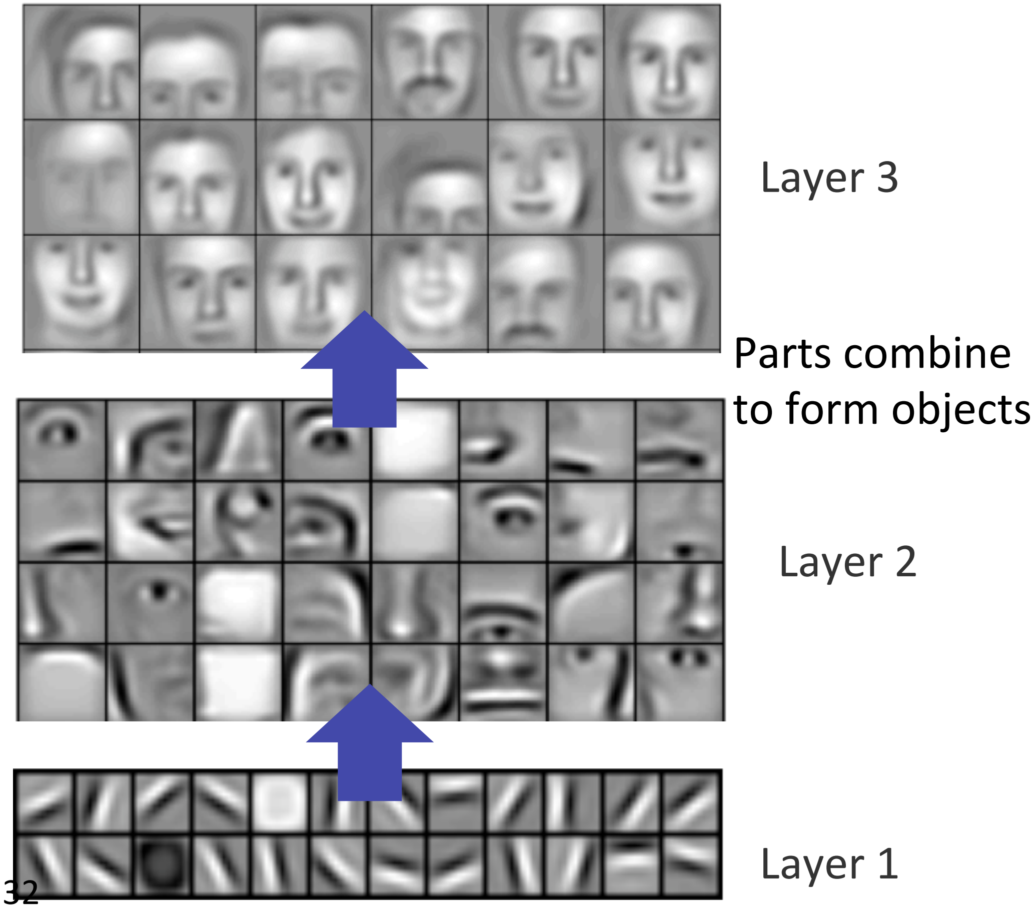
It works! Speech + vision breakthroughs

Learning multiple levels of representation



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

Successive model layers learn deeper intermediate representations

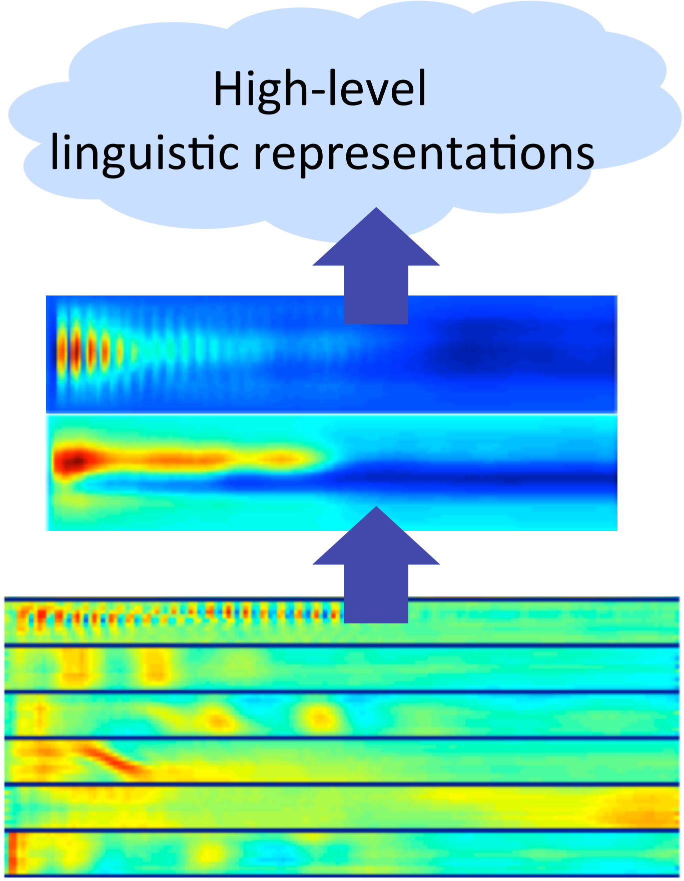


Layer 3

Parts combine to form objects

Layer 2

Layer 1



High-level linguistic representations

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

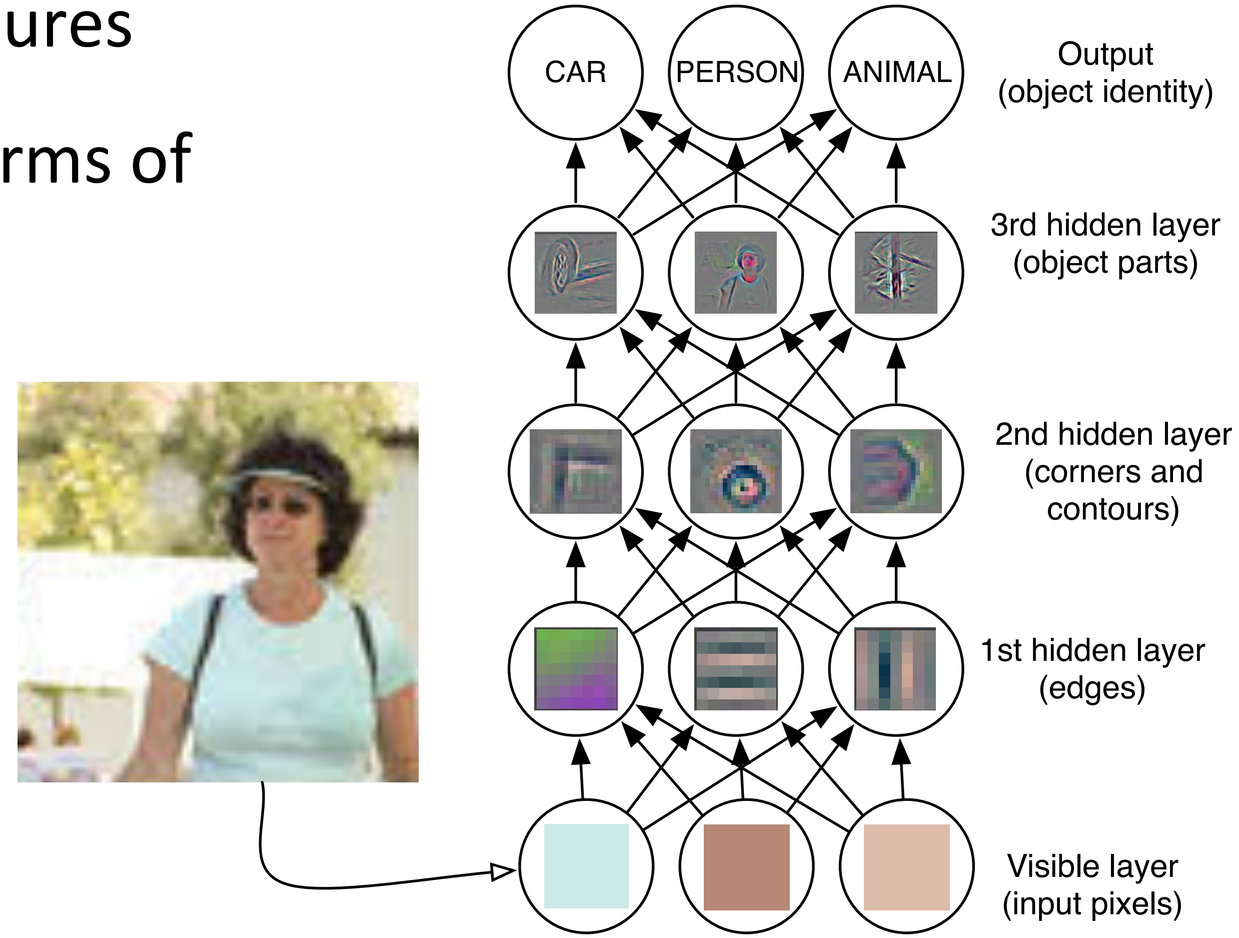
Composing Features on Features

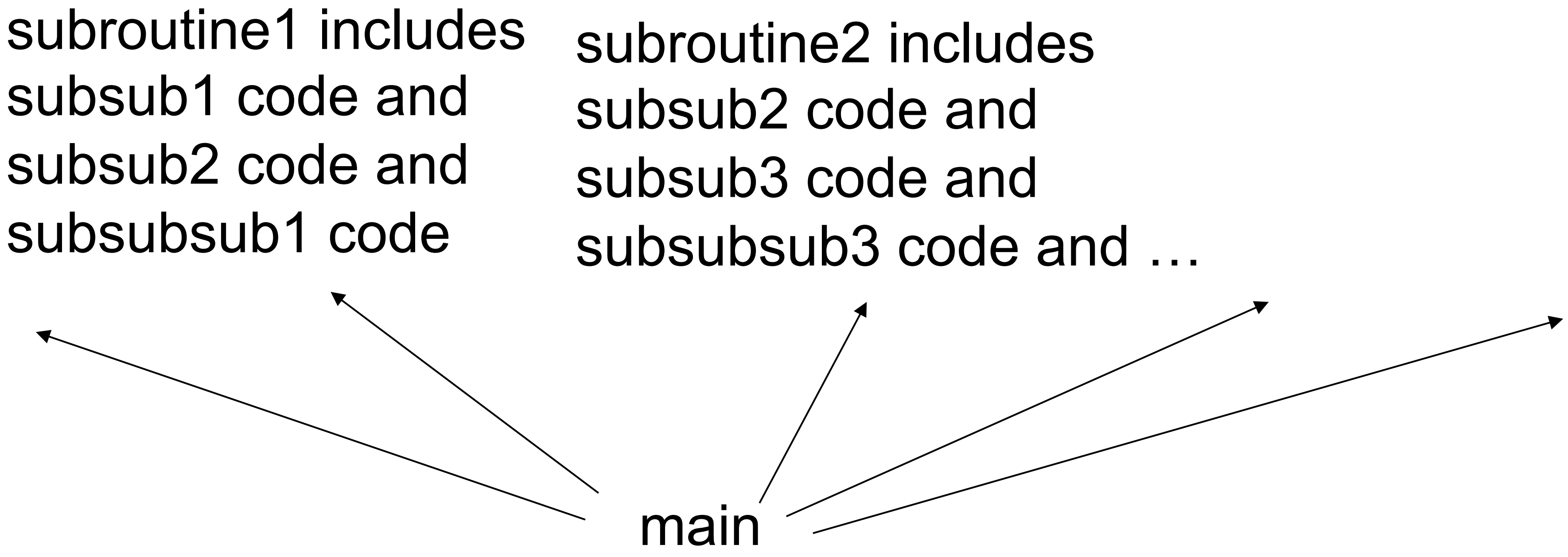
Higher-level features

are defined in terms of

lower-level

features



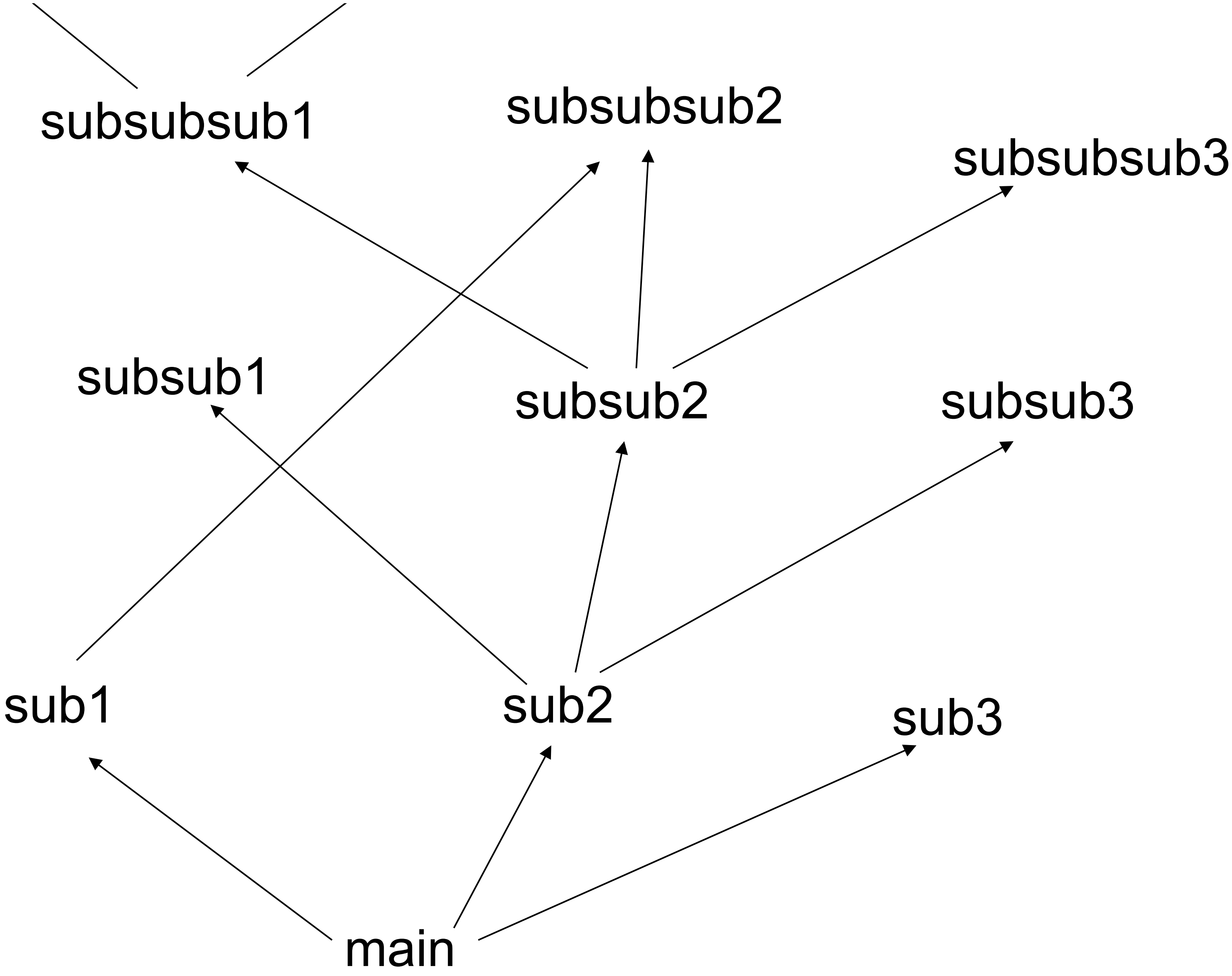


subroutine1 includes
subsub1 code and
subsub2 code and
subsubsub1 code

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

main

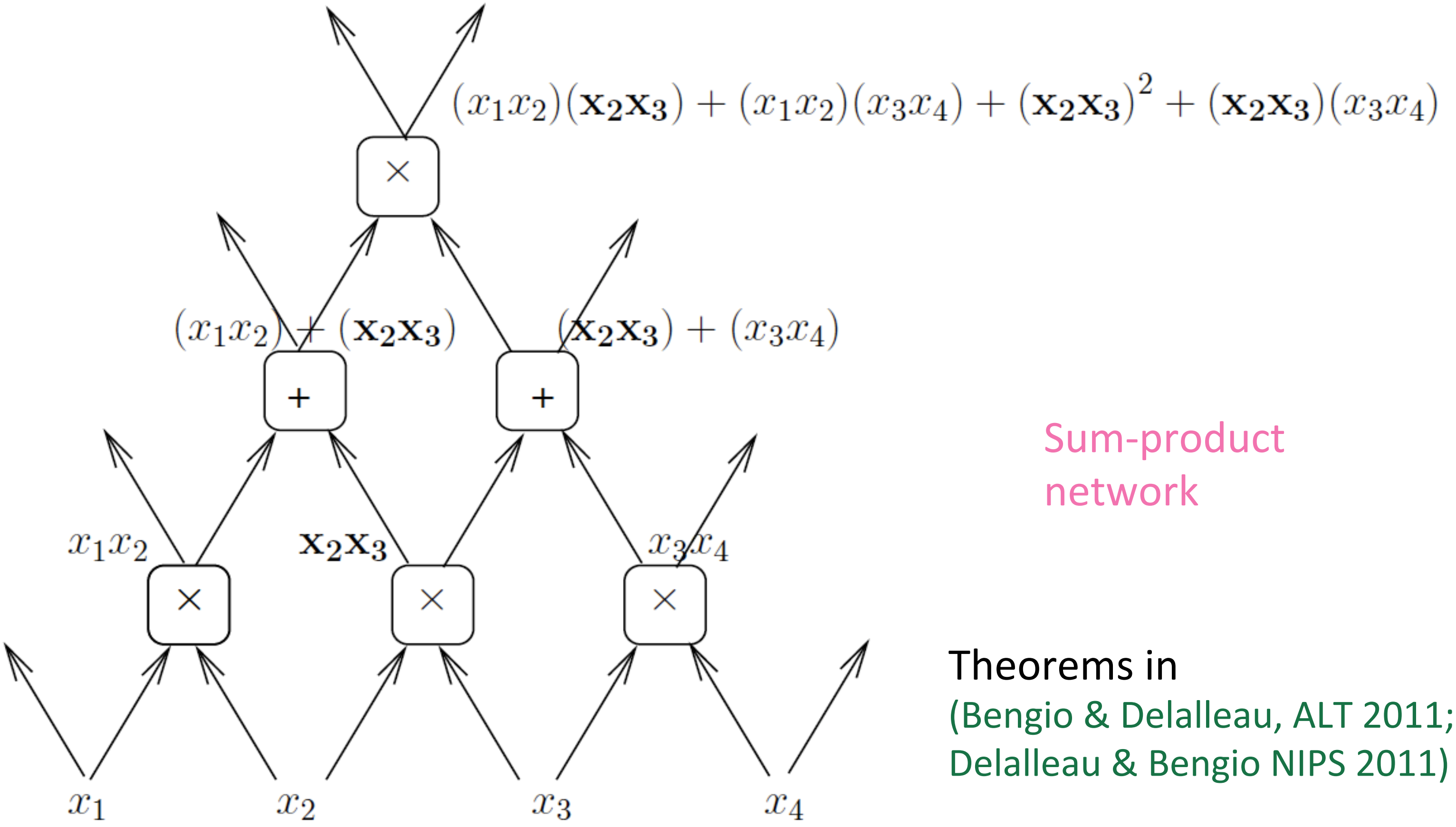
“Shallow” computer program



"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:

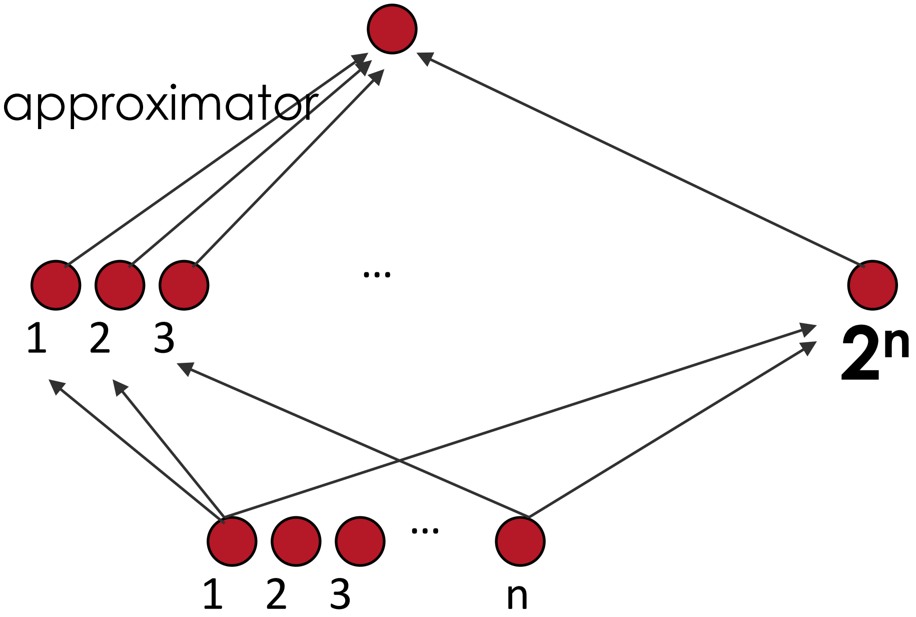
2 layers of
- Logic gates
- Formal neurons
- RBF units

= 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)

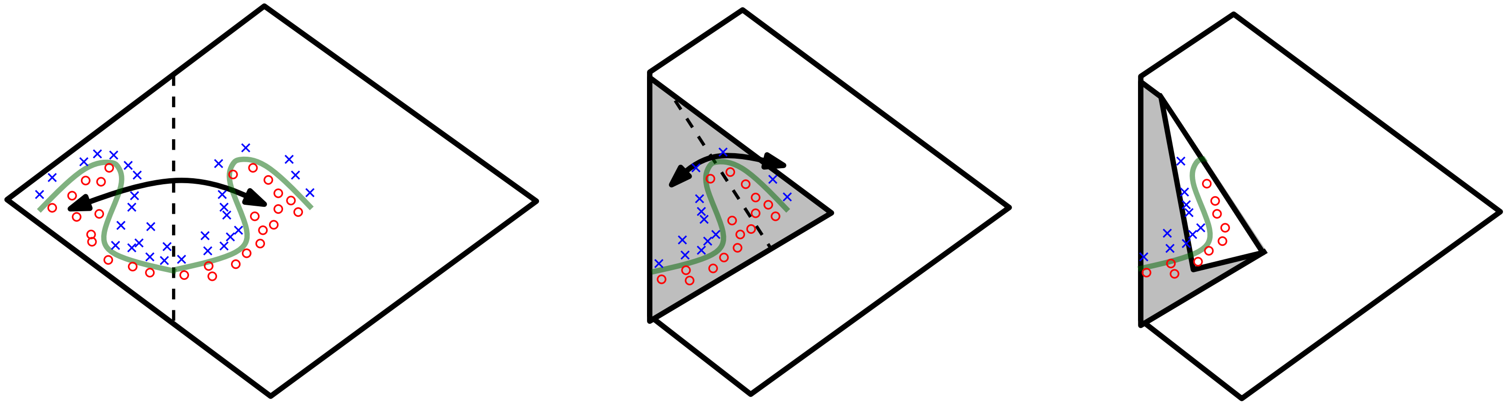
Some functions compactly represented with k layers may require exponential size with 2 layers



New theoretical result: Expressiveness of deep nets with piecewise-linear activation fns

(Pascanu, Montufar, Cho & Bengio; ICLR 2014)

Deeper nets with rectifier/maxout units are exponentially more expressive than shallow ones (1 hidden layer) because they can split the input space in many more (not-independent) linear regions, with constraints, e.g., with abs units, each unit creates mirror responses, folding the input space:



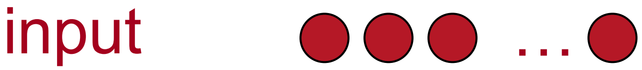
Major Breakthrough in 2006

- 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

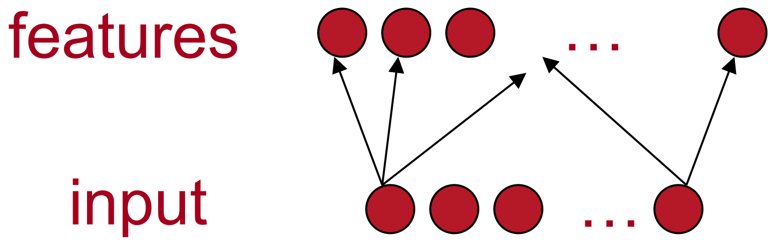
(Bengio & LeCun 2007), Scaling Learning Algorithms towards AI

Layer-wise Unsupervised Learning

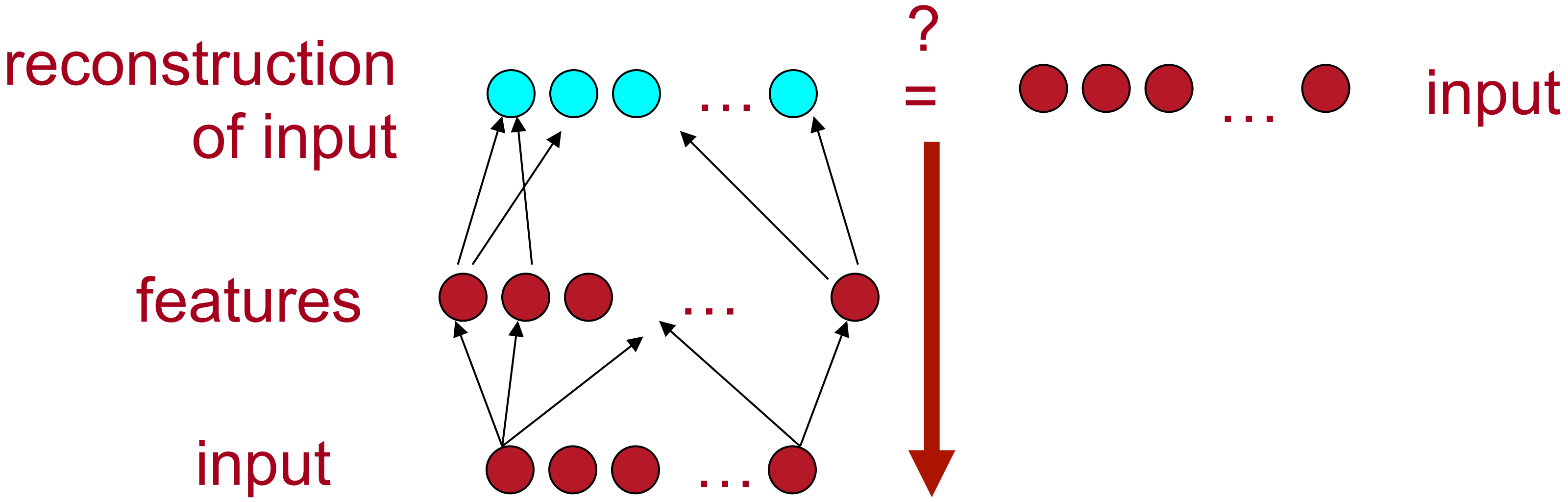


input

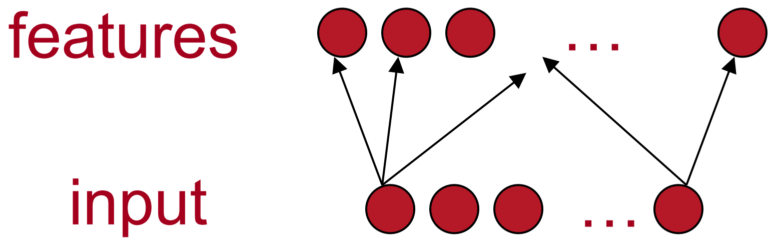
Layer-Wise Unsupervised Pre-training



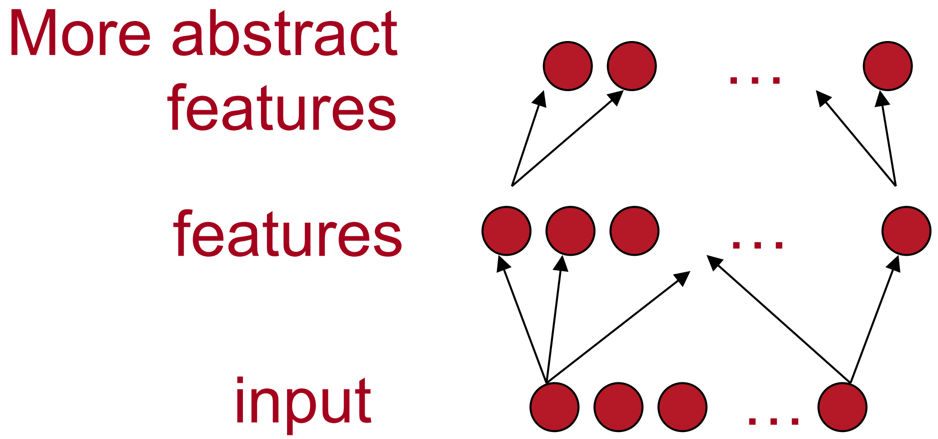
Layer-Wise Unsupervised Pre-training



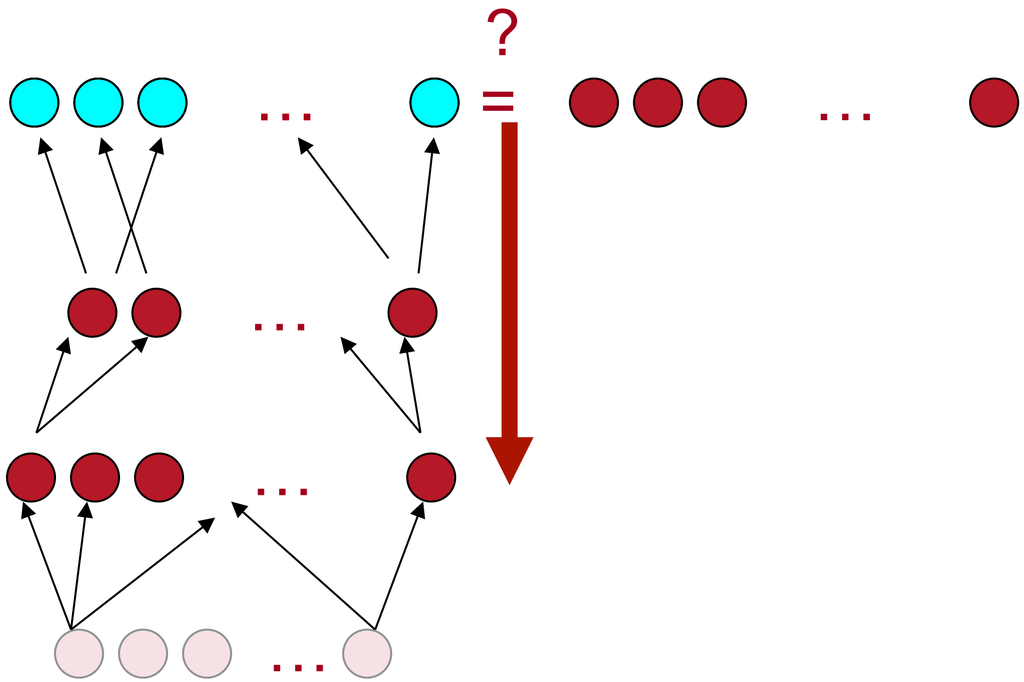
Layer-Wise Unsupervised Pre-training



Layer-Wise Unsupervised Pre-training



Layer-wise Unsupervised Learning



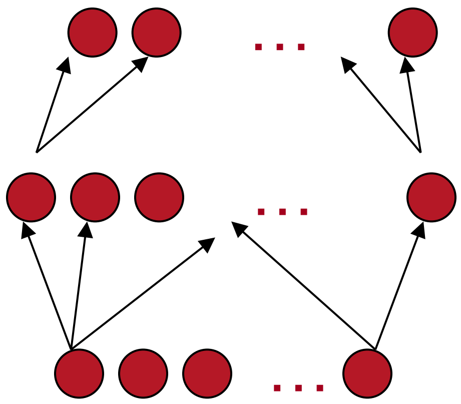
reconstruction
of features

More abstract
features

features

input

Layer-Wise Unsupervised Pre-training

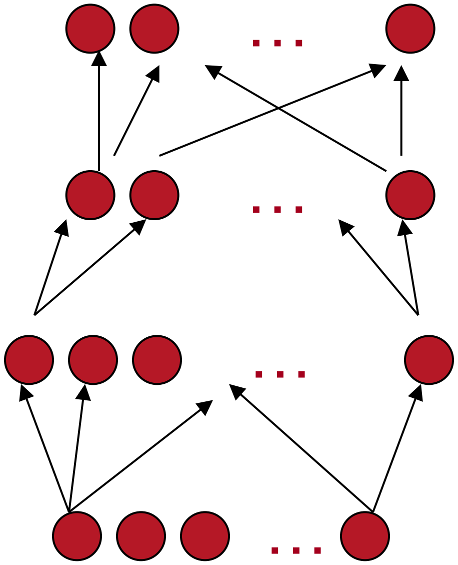


More abstract features

features

input

Layer-wise Unsupervised Learning



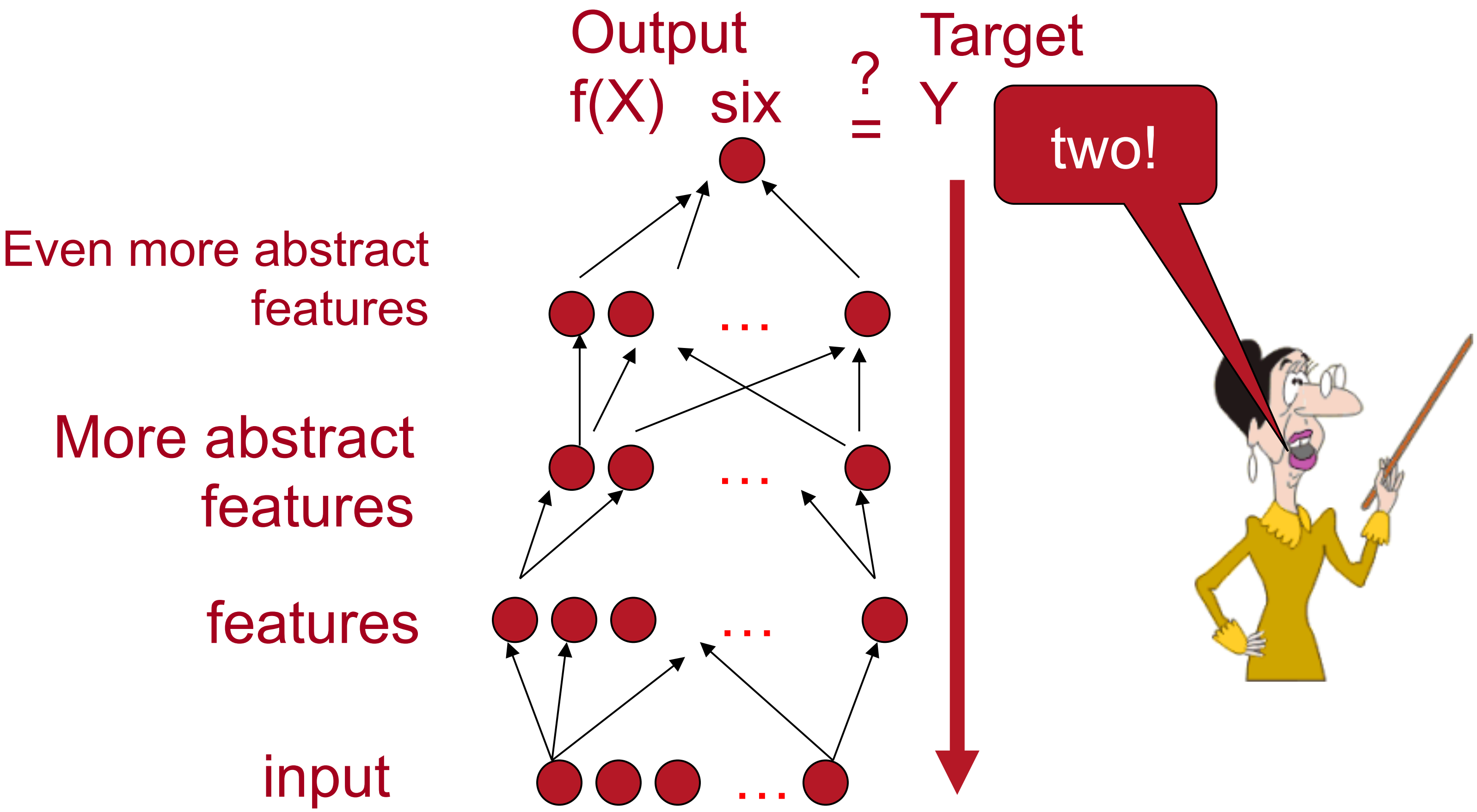
Even more abstract features

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)

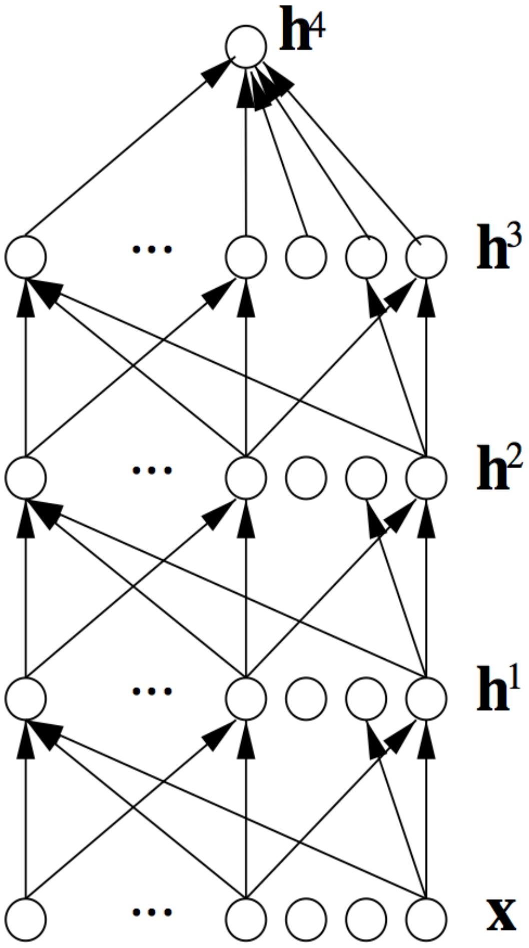
Deep Supervised Neural Nets

- 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.



Machine Learning 101

- Family of functions f_θ
- Tunable parameters θ
- Examples *Z* ~ unknown data generating distribution *P(Z)*
- Loss *L* maps *Z* and f_θ to a scalar
- Regularizer *R* (typically on depends on θ but possibly also on *Z*)
- Training criterion:

$$C(\theta) = \text{average}_{Z\sim\text{dataset}} L(f_\theta, Z) + R(\theta, Z)$$

- Approximate minimization algorithm to search for good θ
- Supervised learning:
  - *Z=(X,Y)* and $L = L(f_\theta(X), Y)$

Log-Likelihood for Neural Nets

- Estimating a conditional probability $P(Y|X)$
- Parametrize it by $P(Y|X) = P(Y|\omega = f_\theta(X))$
- Loss = $-\log P(Y|X)$
- E.g. Gaussian *Y*, $\omega = (\mu, \sigma)$

  typically only μ is the network output, depends on *X*

  Equivalent to MSE criterion:

  Loss = $-\log P(Y|X) = \log\sigma + ||f_\theta(X) - Y||^2/\sigma^2$
- E.g. Multinoulli *Y* for classification,

  $$\omega_i = P(Y = i|x) = f_{\theta,i}(X) = \text{softmax}_i(a(X))$$

  Loss = $-\log\omega_Y = -\log f_{\theta,Y}(X)$

Multiple Output Variables

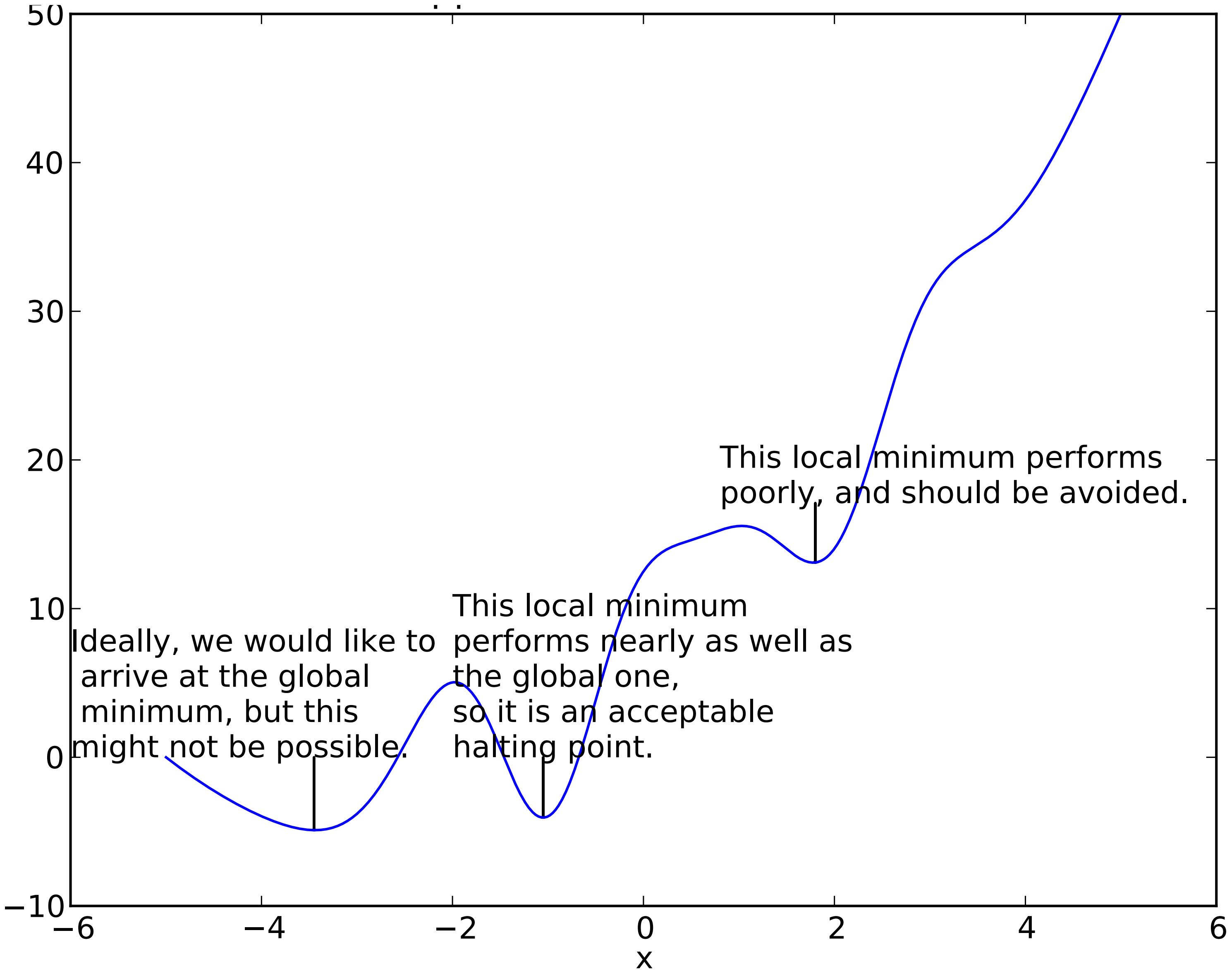
- If they are conditionally independent (given X), the individual prediction losses add up:

$$-\log P(Y|X) = -\log P(Y_1, \ldots Y_k|X) = -\log \prod_i P(Y_i|X) = -\sum_i \log P(Y_i|X)$$

- Likelihood if some Y_i's are missing: just ignore those losses

- If not conditionally independent, need to capture the conditional joint distribution $P(Y_1, \ldots Y_k|X)$
  - Example: output = image, sentence, tree, etc.
  - Similar to unsupervised learning problem of capturing joint
  - Exact likelihood is often similarly intractable

Approximate Minimization



Gradient-Based Optimization & Gradient Descent

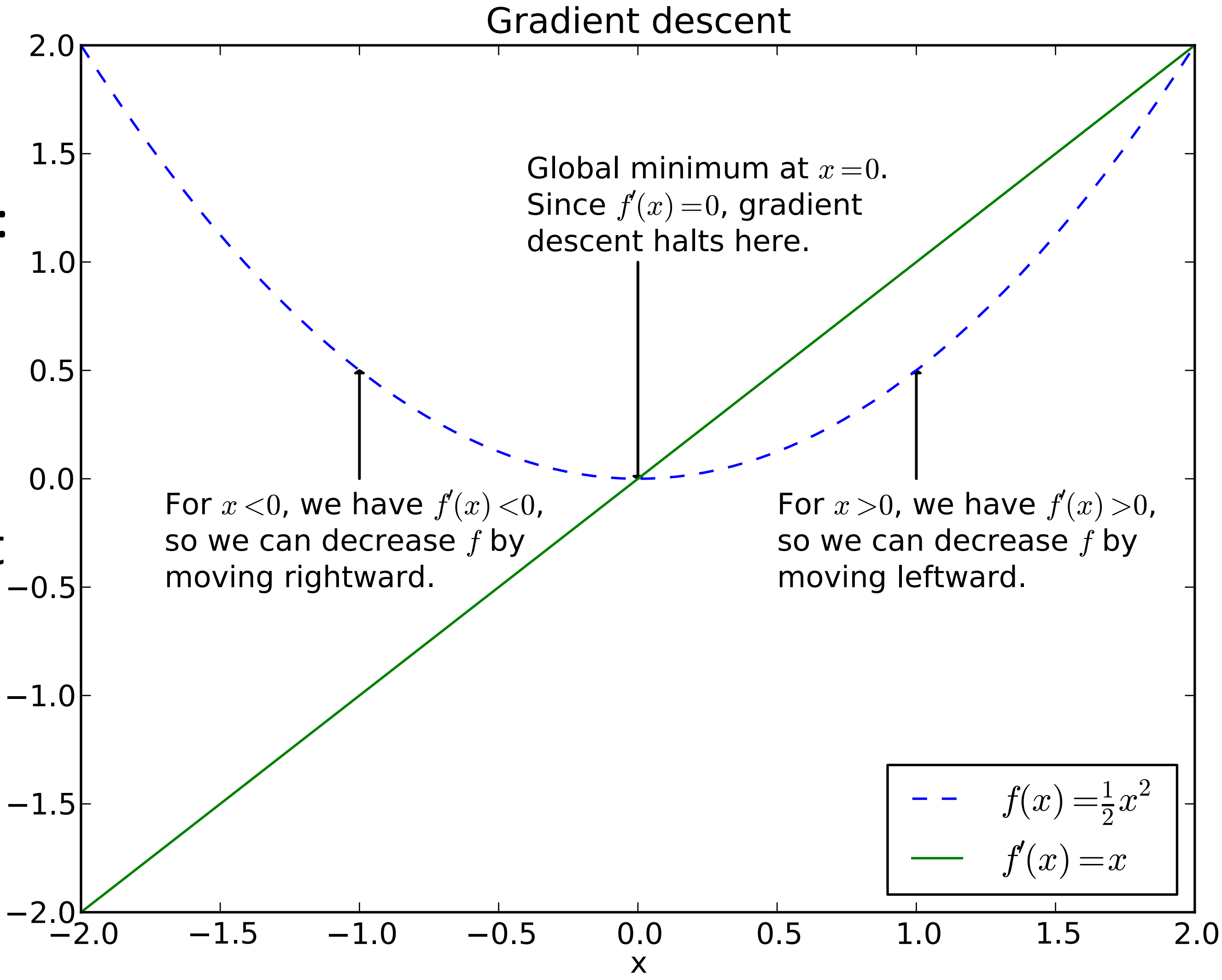
Gradient descent parameter update:

New param =

Old param –

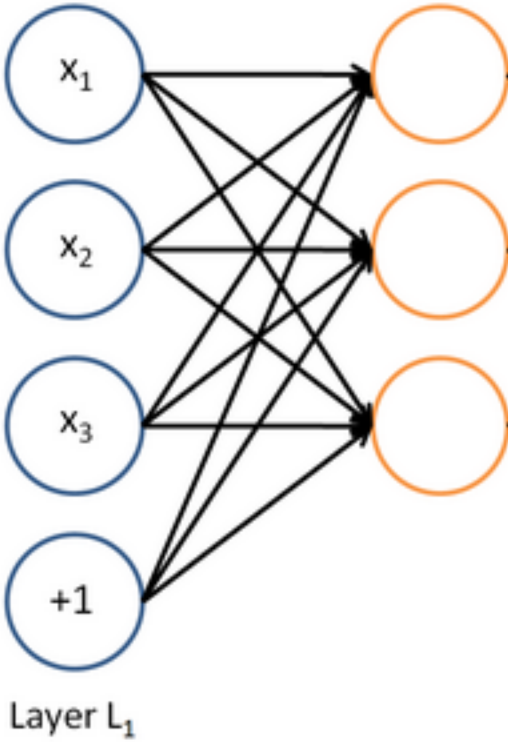
stepsize * gradient

$$x \leftarrow x - \epsilon f'(x)$$



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

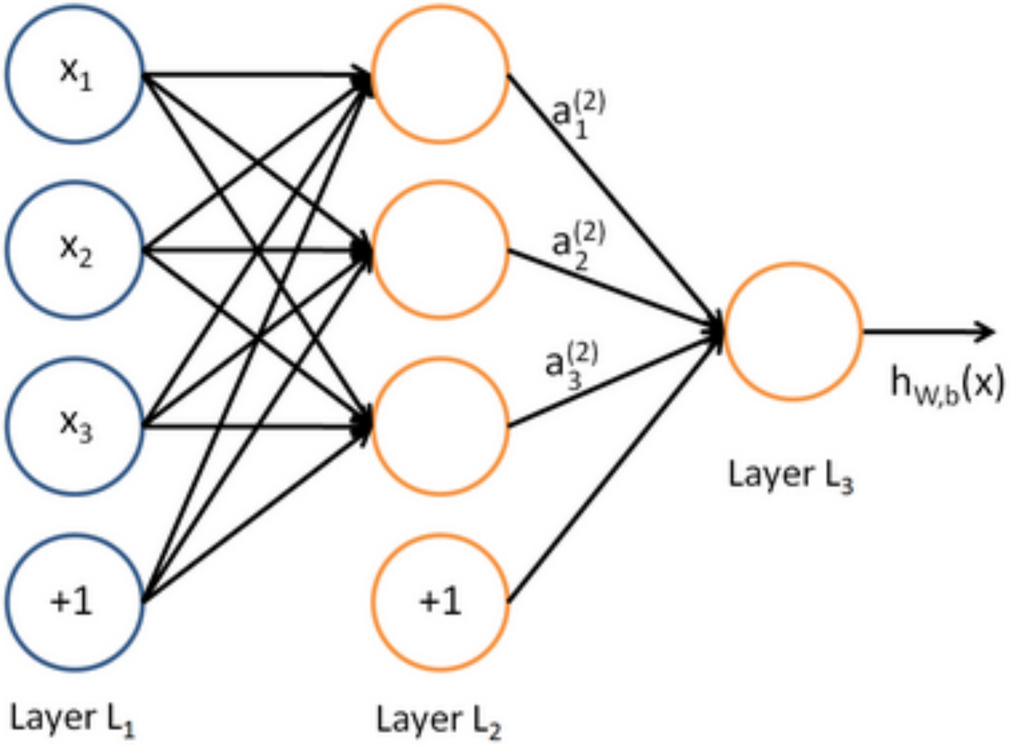
If we feed a vector of inputs through a bunch of logistic regression functions, then we get a vector of outputs



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

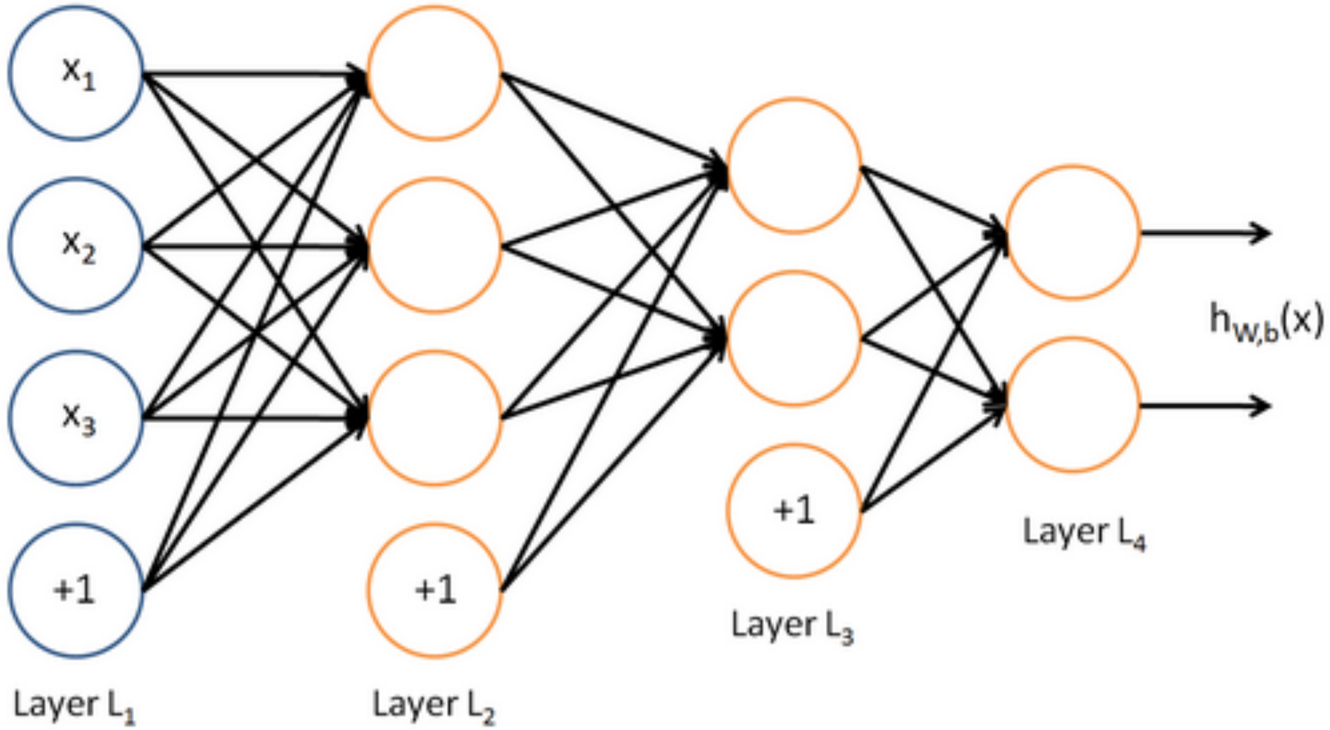
… which we can feed into another logistic regression function



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

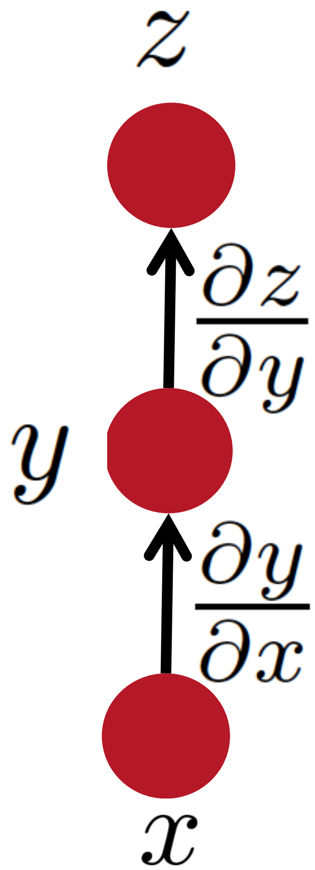
- Compute gradient of example-wise loss wrt parameters

- 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

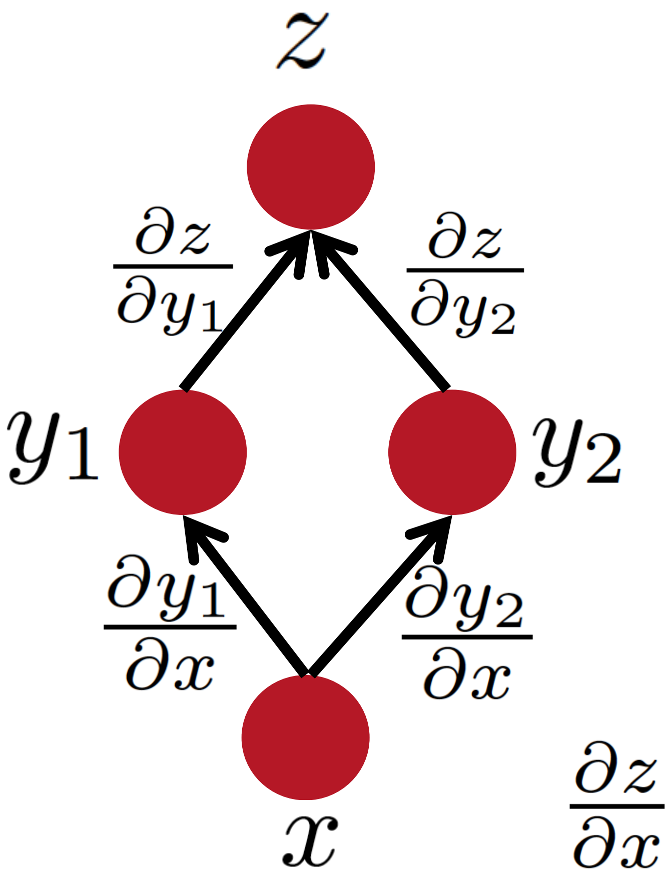
$$\Delta z = \frac{\partial z}{\partial y} \Delta y$$

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

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

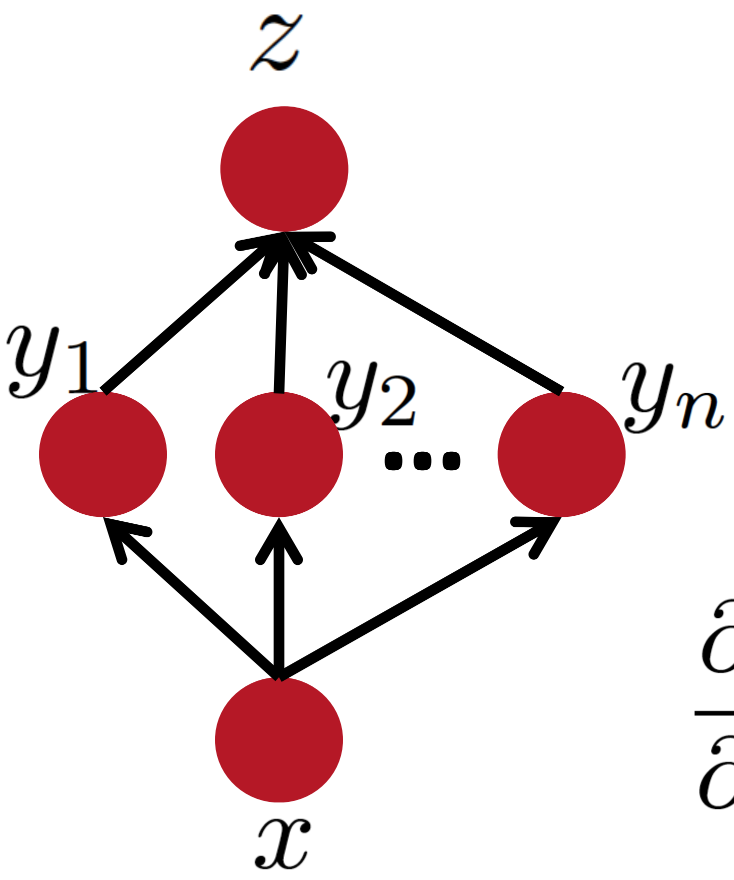
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}$$

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

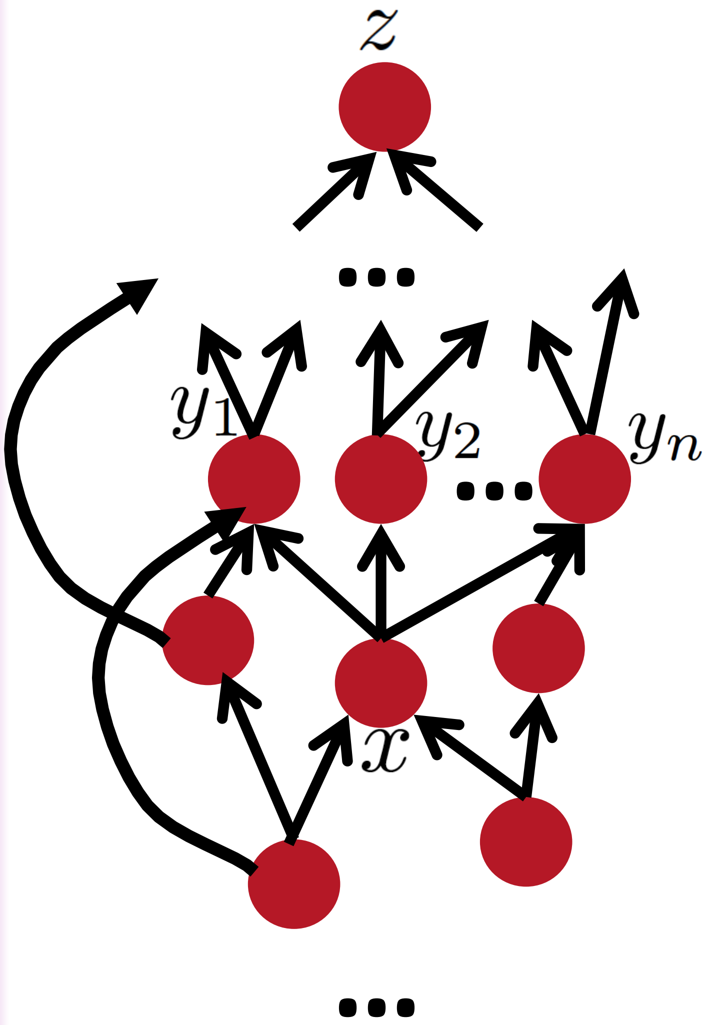
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial x}$$

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

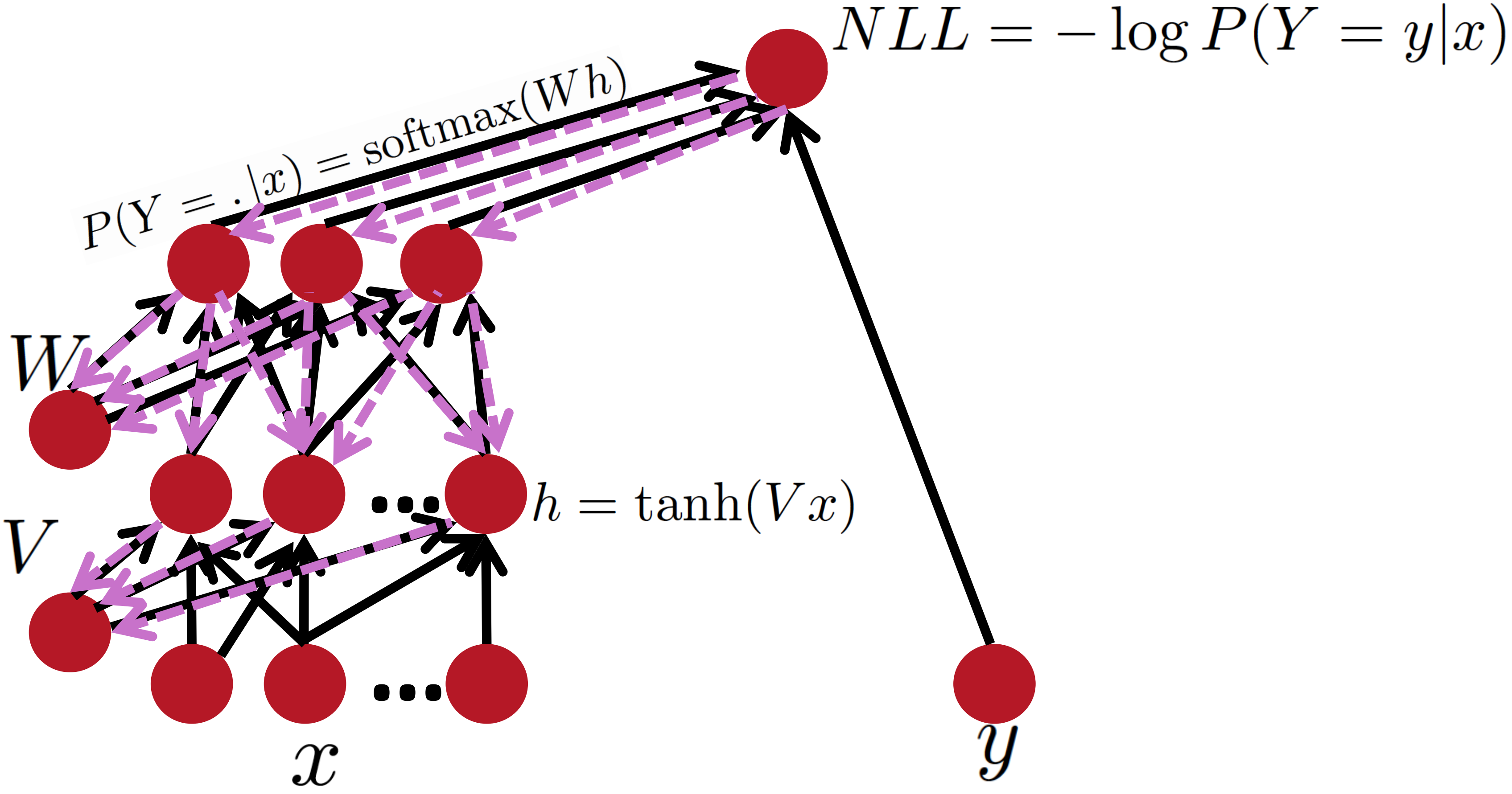
node = computation result

arc = computation dependency

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

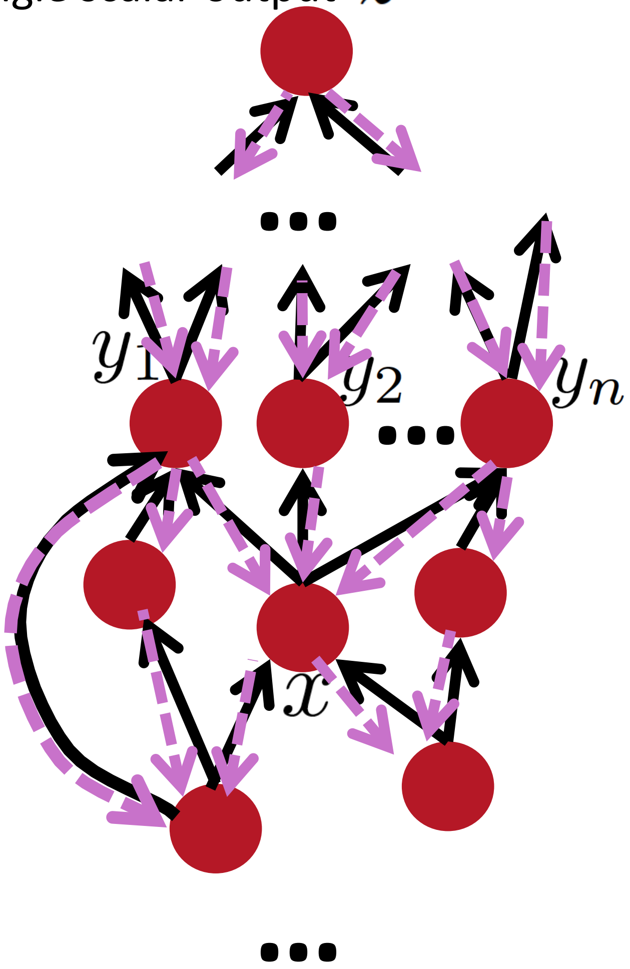
$$\frac{\partial z}{\partial x} = \sum_{i=1}^{n} \frac{\partial z}{\partial y_i} \frac{\partial y_i}{\partial x}$$

Back-Prop in Multi-Layer Net



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:

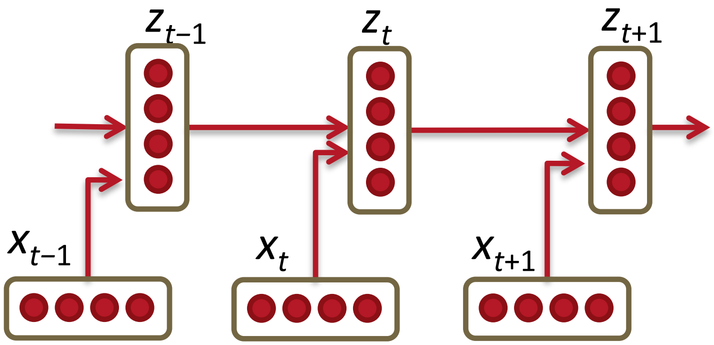
     Compute gradient wrt each node using gradient wrt successors

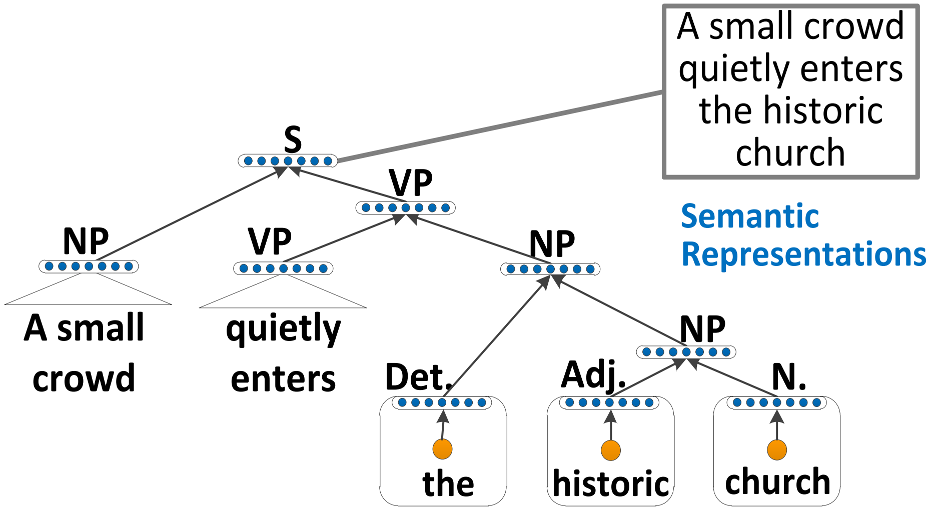
$\{y_1,\ y_2,\ \ldots\ y_n\}$ = successors of x

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

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



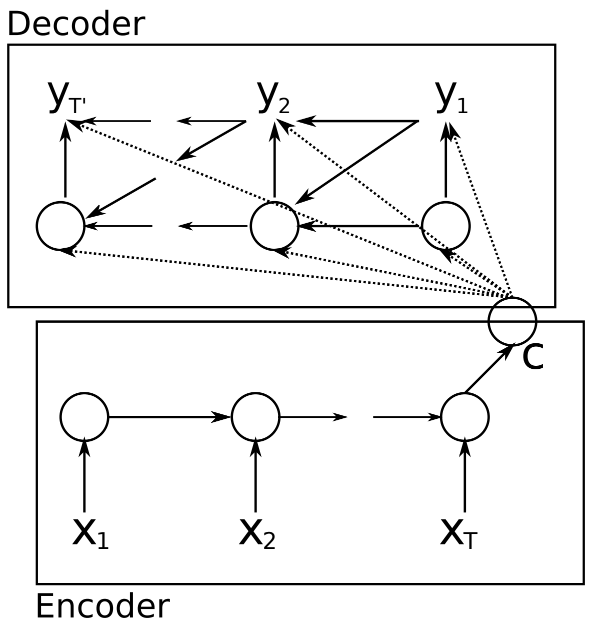


Generative RNNs for Machine Translation

- Decoder = ‘generative’ RNN: context C → distribution over word sequences
  - $P(Y_1..Y_{T'} \mid C) = \prod P(Y_t \mid H_t, C)$

  where hidden state H_t summarizes past seq.

  $H_t = f(H_{t-1}, Y_{t-1}, C) = F(Y_{t-1}, \ldots Y_1, C)$



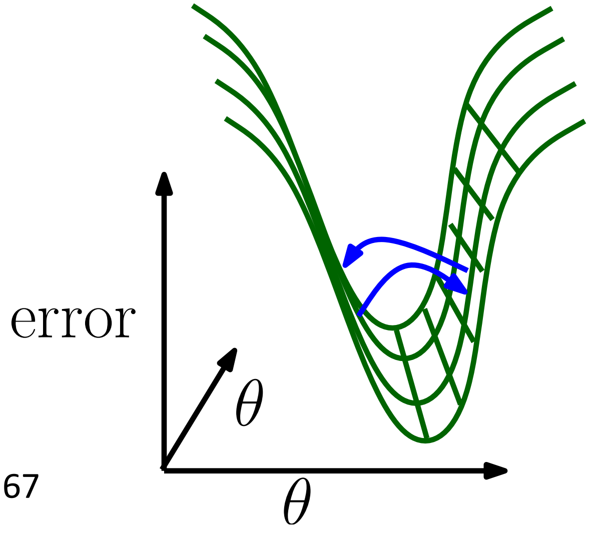
- Directed graphical$_t$ model: ancestral sampling from Y_1 to $Y_{T'}$.

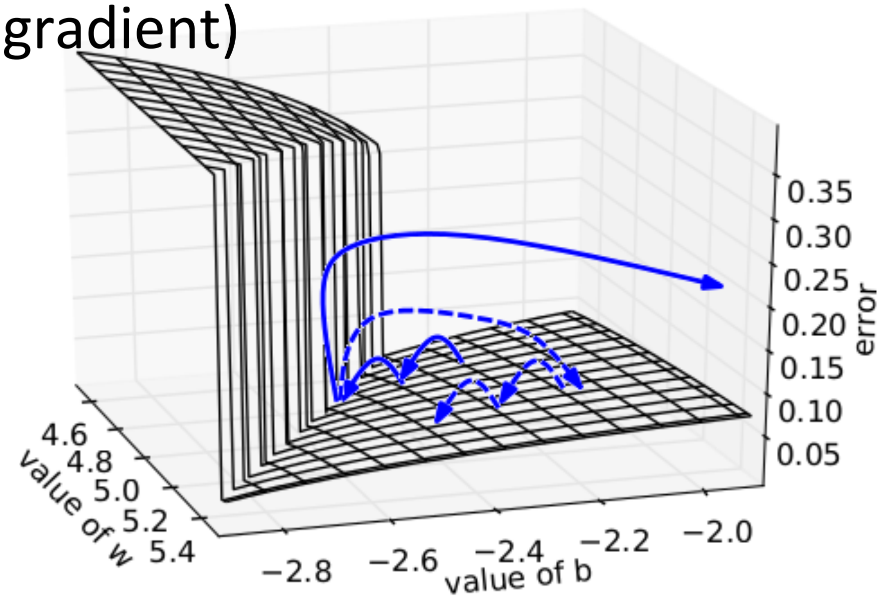
- Output sequence can be of different length $T' \neq T$ not necessarily aligned with input sequence

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)



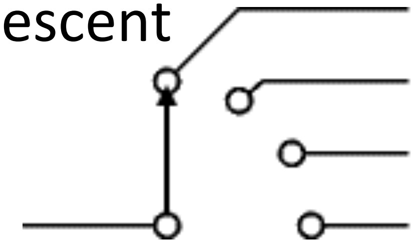


Gradient Norm Clipping

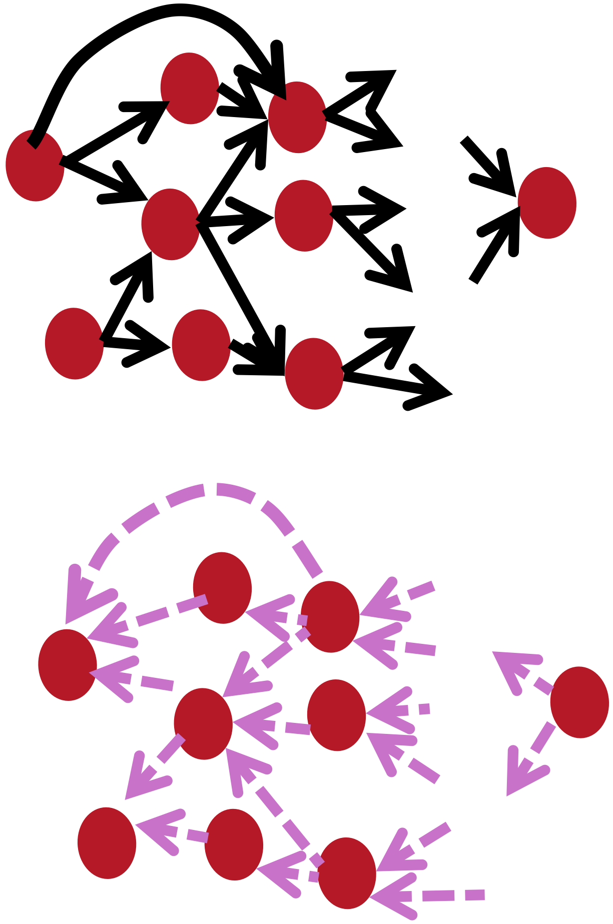
```
$\hat{\mathbf{g}} \leftarrow \frac{\partial error}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ then
    $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$
end if
```

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

Is there any hope to generalize non-locally?

Yes! Need good priors!

Depth prior: Abstraction

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

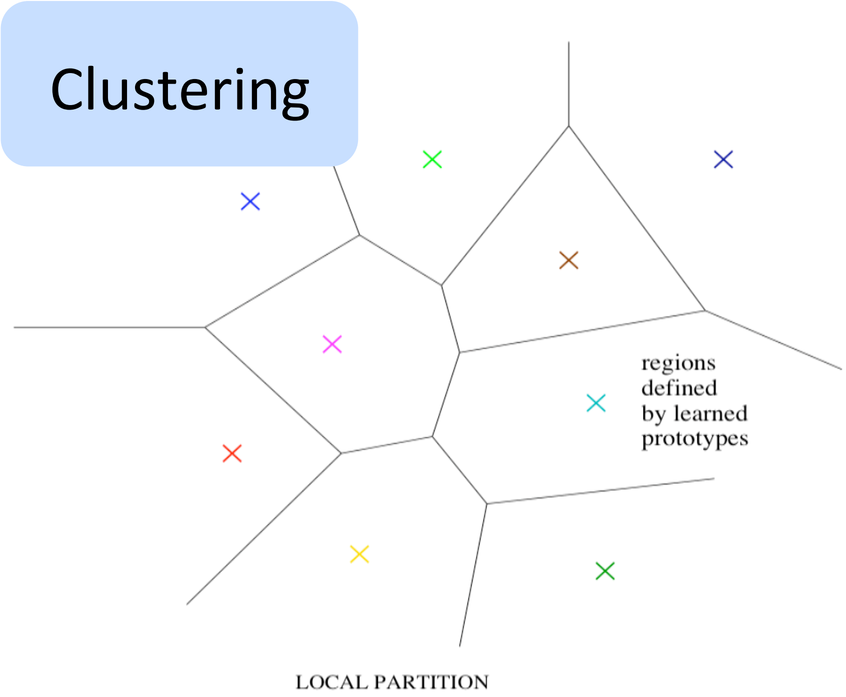
Exploiting compositionality gives an exponential gain in representational power

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**

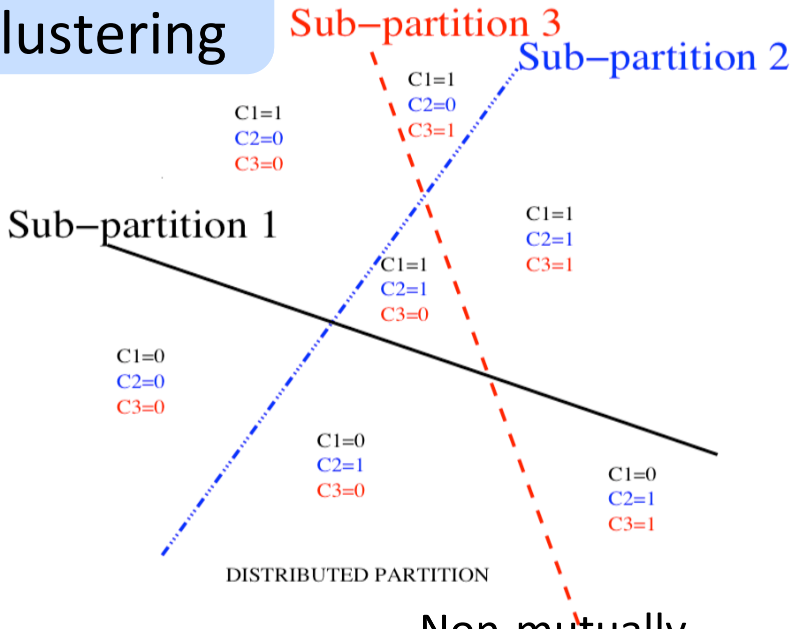
→ No non-trivial generalization to regions without examples

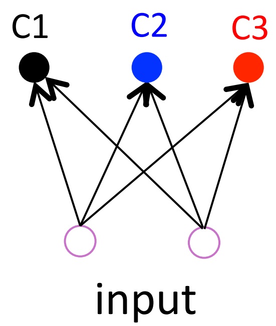
The need for distributed representations

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**

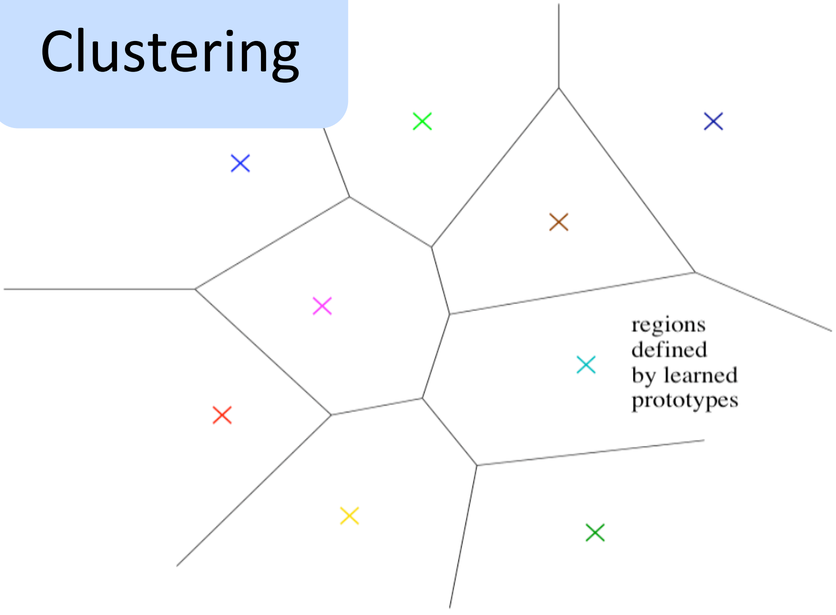




Non-mutually exclusive features/ attributes create a combinatorially large set of distinguiable configurations

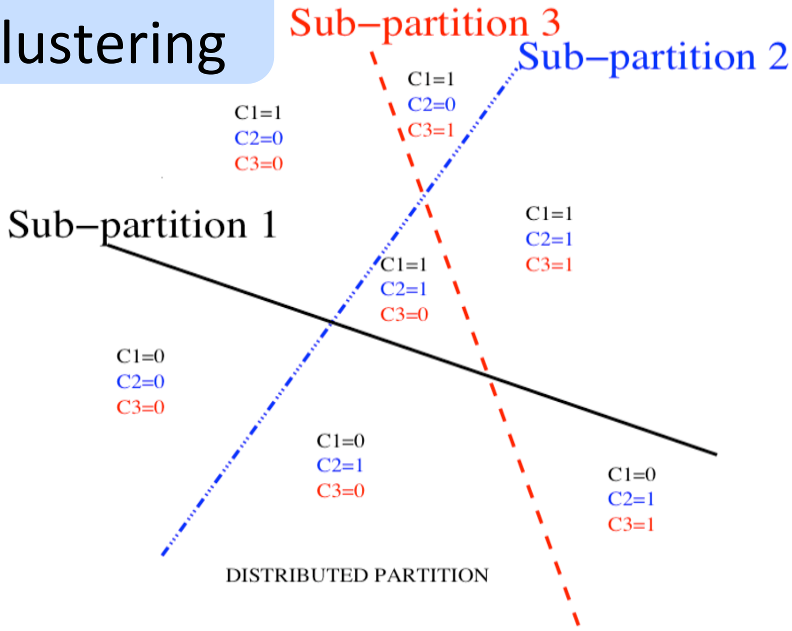
The need for distributed representations

Clustering



LOCAL PARTITION

Multi-Clustering



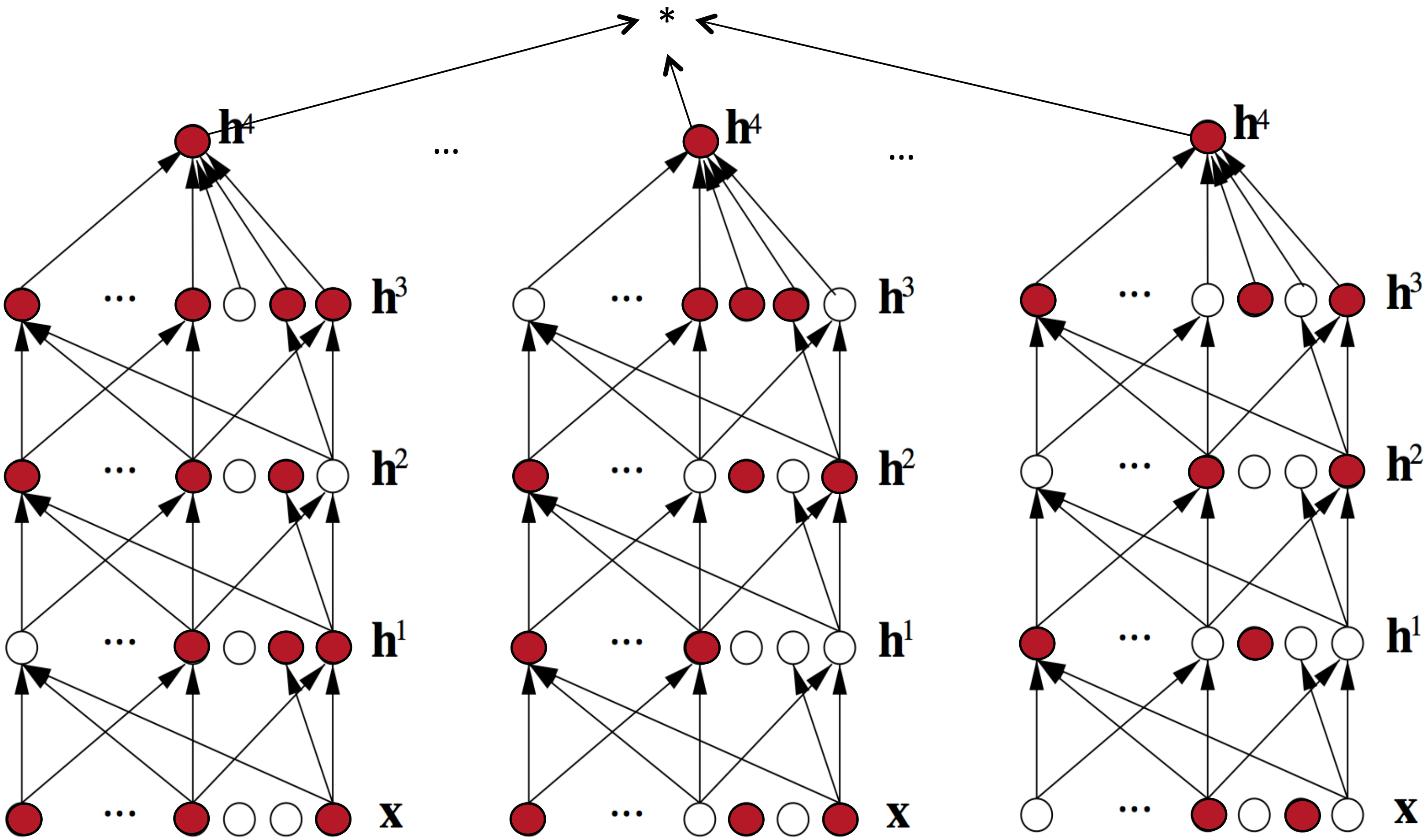
DISTRIBUTED PARTITION

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

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

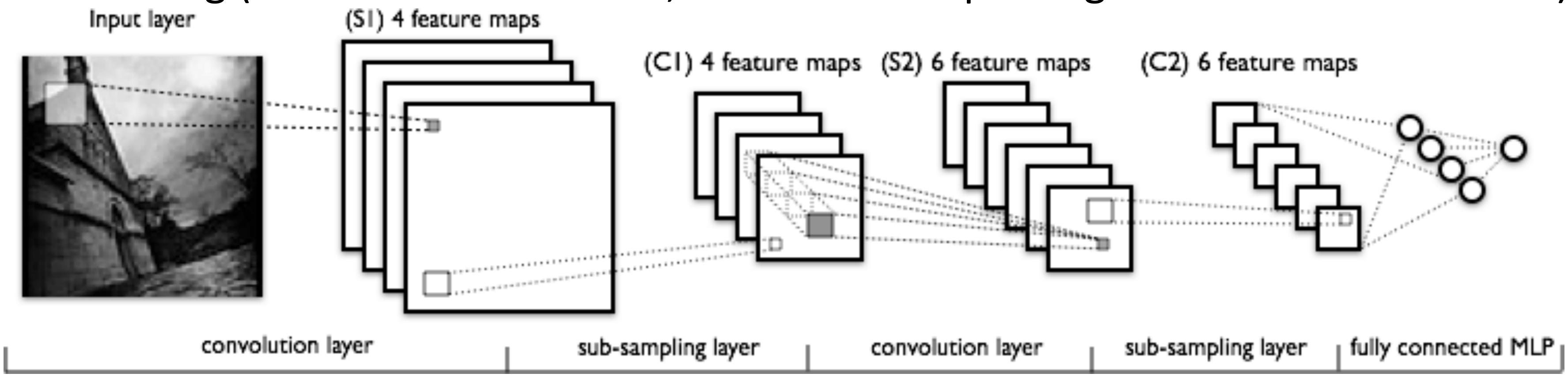
- **Dropouts** trick: during training multiply neuron output by random bit (p=0.5), during test by 0.5
- Used in deep supervised networks
- Similar to denoising auto-encoder, but corrupting every layer
- Works better with some non-linearities (rectifiers, maxout) (Goodfellow et al. ICML 2013)
- Equivalent 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)

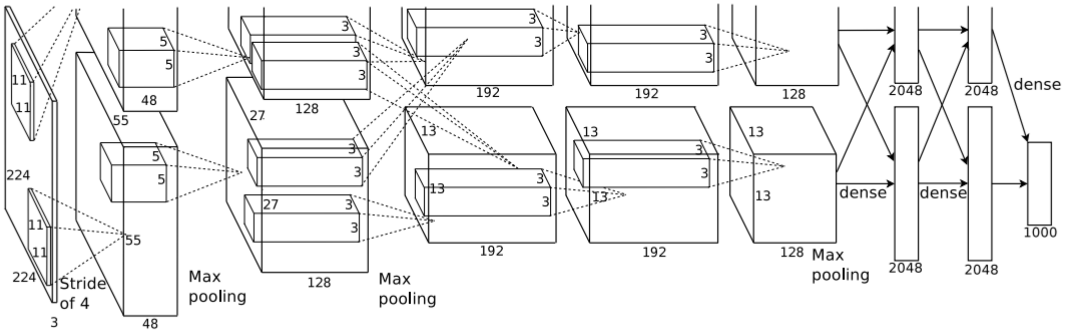
Dropout Regularizer: Super-Efficient Bagging

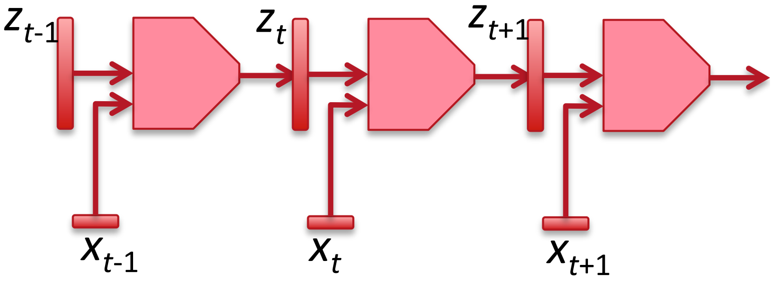


Temporal & Spatial Inputs: Convolutional & Recurrent Nets

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





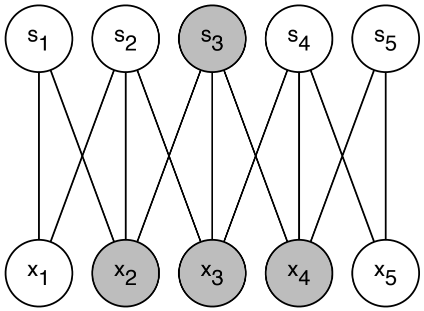


Recurrent nets (RNNs) can summarize information from the past

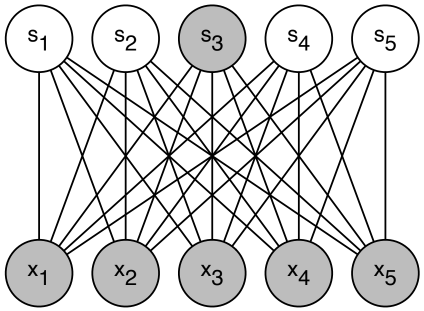
Bidirectional RNNs also summarize information from the future

Convolution = sparse connectivity + parameter sharing

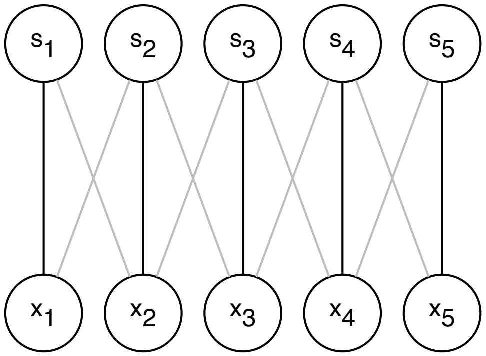
$$s[t] = (x * w)(t) = \sum_{a=-\infty}^{\infty} x[a]w[t-a]$$



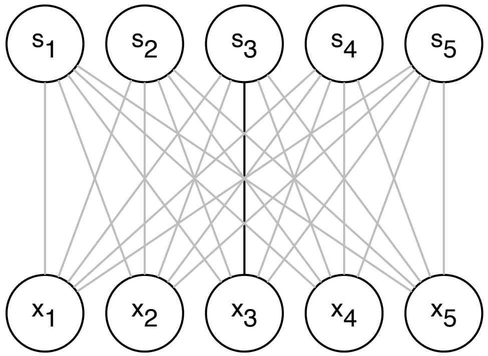
sparse



dense



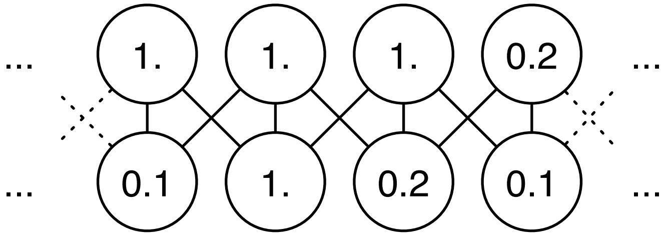
shared



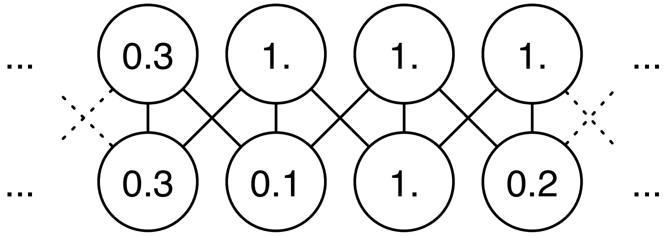
not shared

Pooling Layers

- Aggregate to achieve local invariance

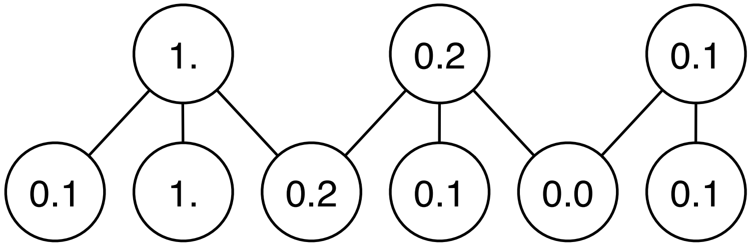


Max-pooling

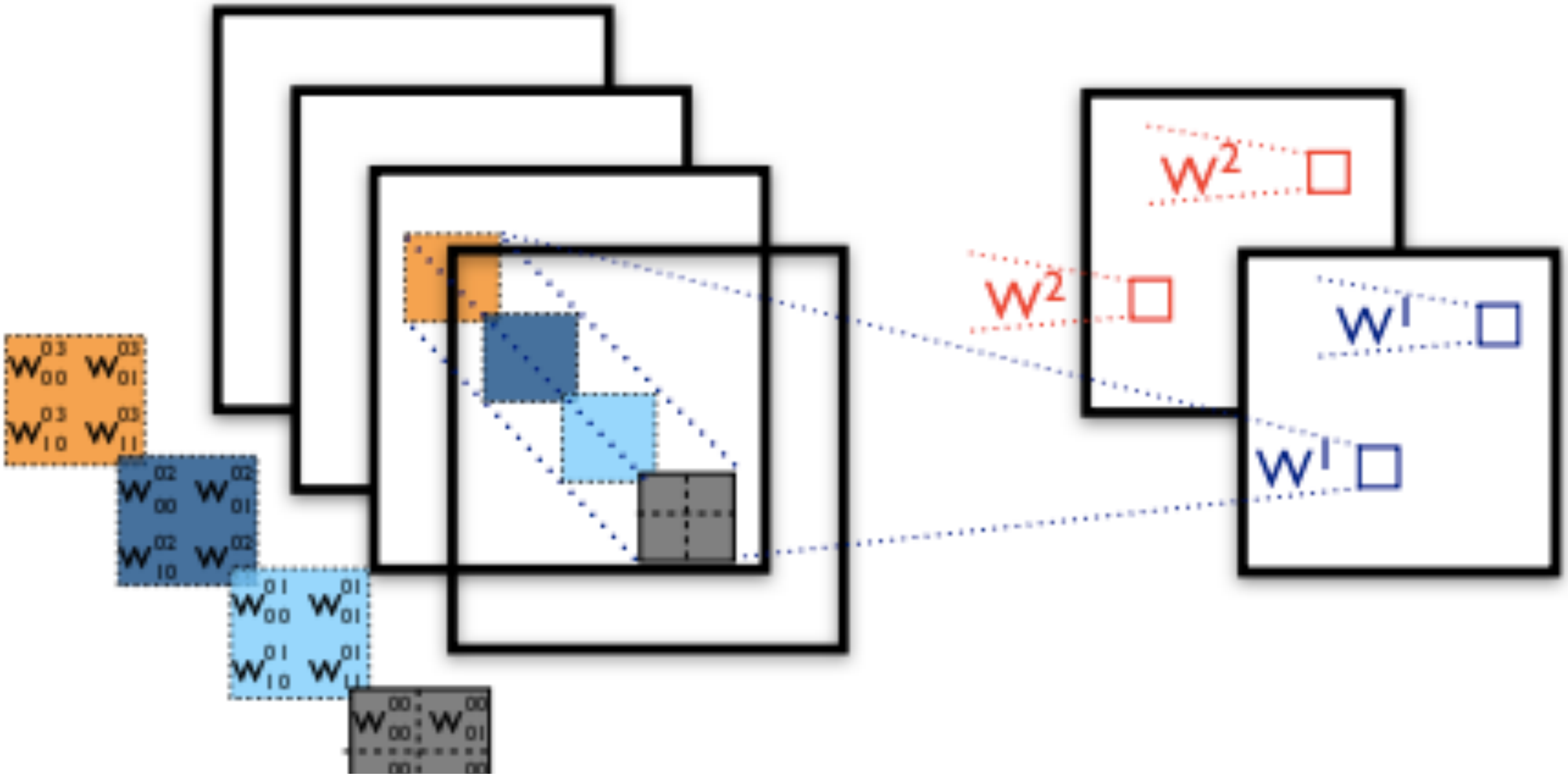


Effect of translation

- Subsampling to reduce temporal/spatial scale and computation

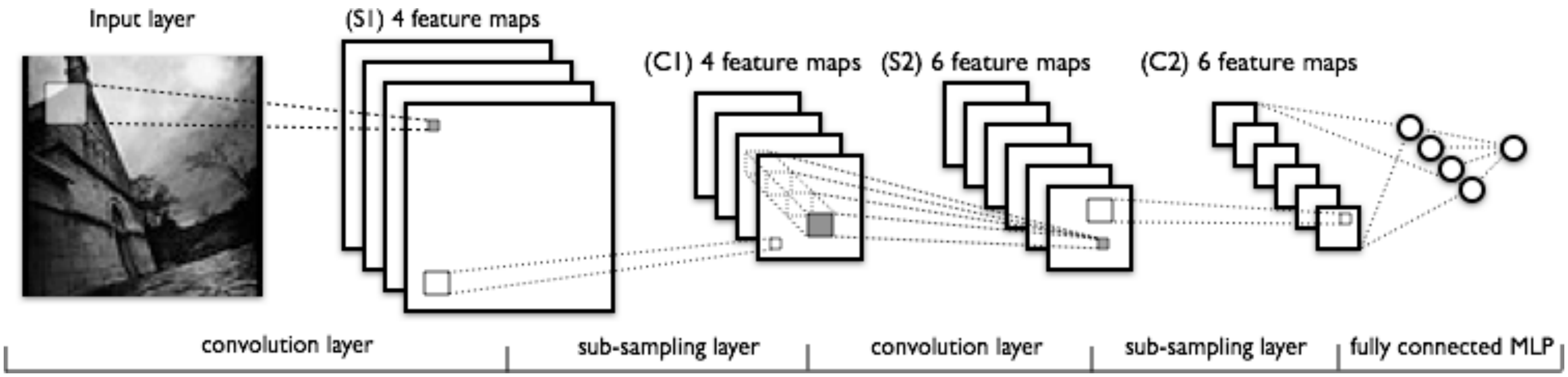


Multiple Convolutions: Feature Maps


W^{03}_{00} W^{03}_{01}
W^{03}_{10} W^{03}_{11}
W^{02}_{00} W^{02}_{01}
W^{02}_{10} W^{02}_{11}
W^{01}_{00} W^{01}_{01}
W^{01}_{10} W^{01}_{11}
W^{00}_{00} W^{00}_{01}
W^2
W^2
W^1
W^1

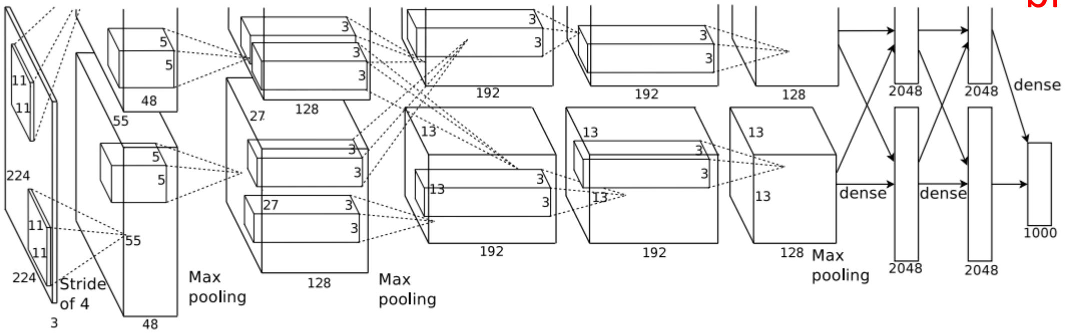
Alternating convolutions & pooling

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

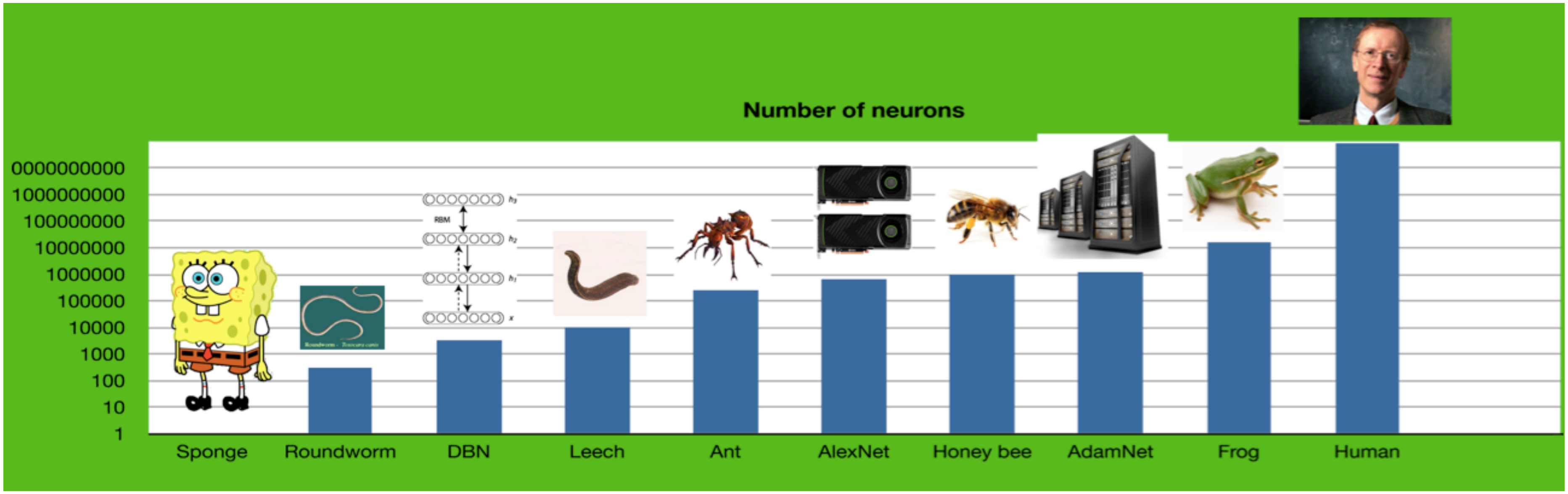


- Increasing number of features, decreasing spatial resolution
- Top layers are fully connected

Krizhevsky, Sutskever & Hinton 2012
breakthrough in object recognition

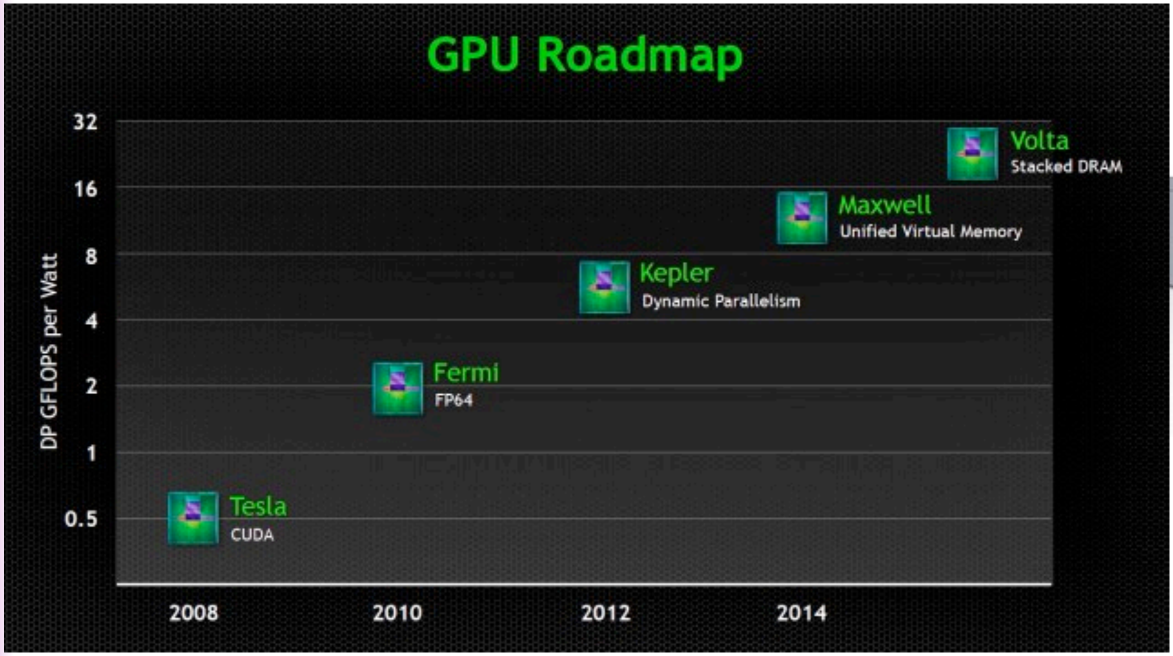


Scaling up computation: we still have a long way to go in raw computational power



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



GPUs

- 10-fold to 100-fold speedup makes them unavoidable
- Now 2688 cores (Titan) in parallel
- Organized in SIMD blocks
- 2 challenges to take advantage of hardware:
  - Parallelization
  - Cache usage (memory access is expensive)

Challenge: Computational Scaling

- Recent breakthroughs in speech, object recognition and NLP hinged on faster computing, GPUs, and large datasets

- In speech, vision and NLP applications we tend to find that

BIGGER IS BETTER

Because deep learning is

EASY TO REGULARIZE while

it is MORE DIFFICULT TO AVOID UNDERFITTING

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)

Distributed Training

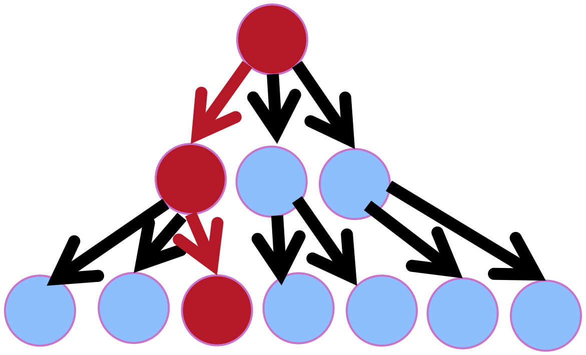
- Minibatches
- Large minibatches + 2nd order & natural gradient methods
- Asynchronous SGD (Bengio et al 2003, Le et al ICML 2012, Dean et al NIPS 2012)
  - Bottleneck: sharing weights/updates among nodes, to avoid node-models to move too far from each other
- Ideas forward:
  - Low-resolution sharing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

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

- We need to improve (reduce) the ratio of

  NUMBER OF COMPUTATIONS / NUMBER OF PARAMETERS

- Extreme success story (but poor generalization): decision trees
  - Deep nets: O(N) computations for O(N) parameters
  - Decision trees: O(N) computations for O(2^N) parameters



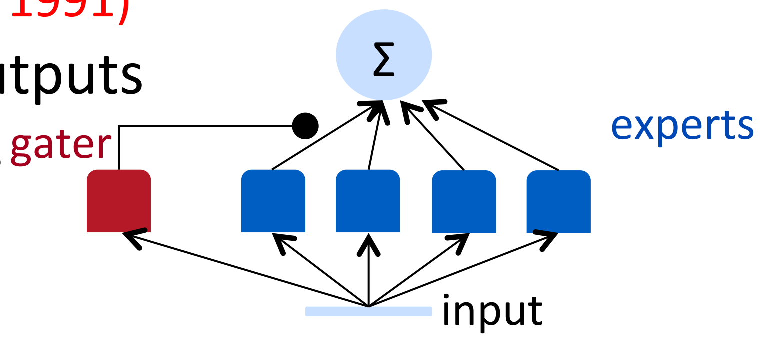
Conditional Computation: only visit a small fraction of parameters / example

- Regular **mixture of experts** (Jacobs et al 1991)

  Output = weighted sum of experts outputs

  Gater partitions input space, chooses which expert to listen in each region.

  Gater softmax output = weights

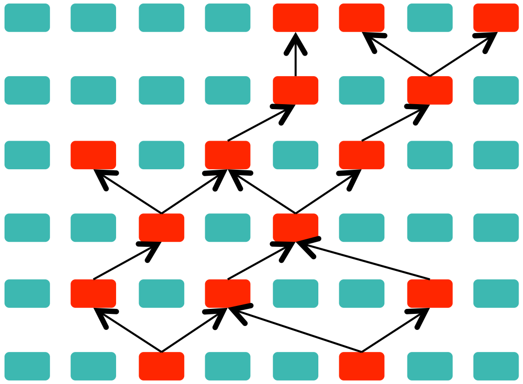


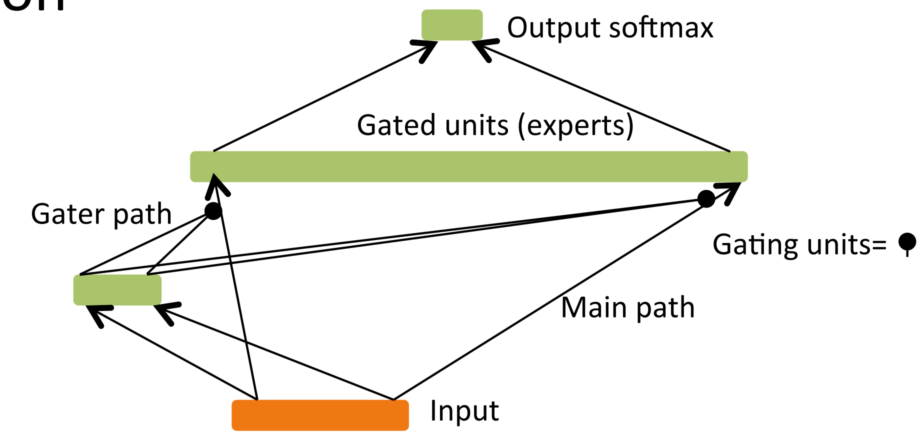
  - No computational benefit, but easier optimization (each expert specializes in its gater-assigned region)

- **Hard mixtures of experts** (Collobert, Bengio & Bengio 2002)
  - Gater takes a hard decision
  - No benefit a training time (need to run all experts to tell gater which one it should have chosen)
  - O(K) speedup at test time if K experts

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

- Conditional computation for deep nets: sparse distributed gaters selecting combinatorial subsets of a deep net
- Challenges:
  - Credit assignment for hard decisions
  - Gated architectures exploration
- (Bengio, Leonard, Courville 2013): *Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation*





Output softmax

Gated units (experts)

Gater path

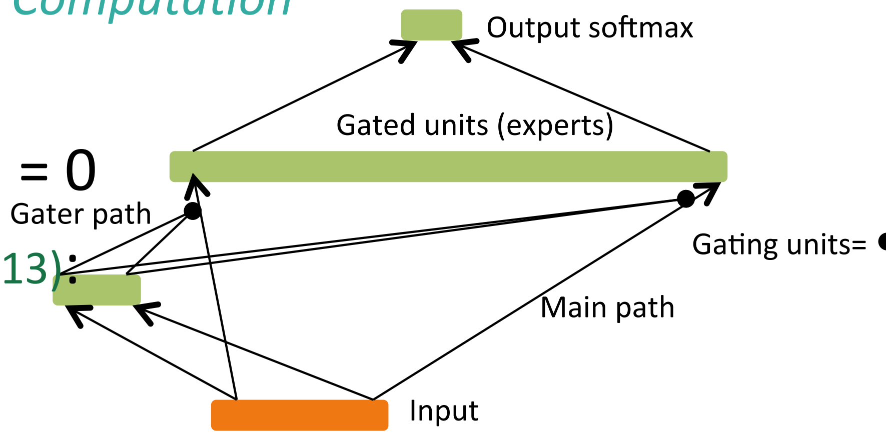
Gating units= ●

Main path

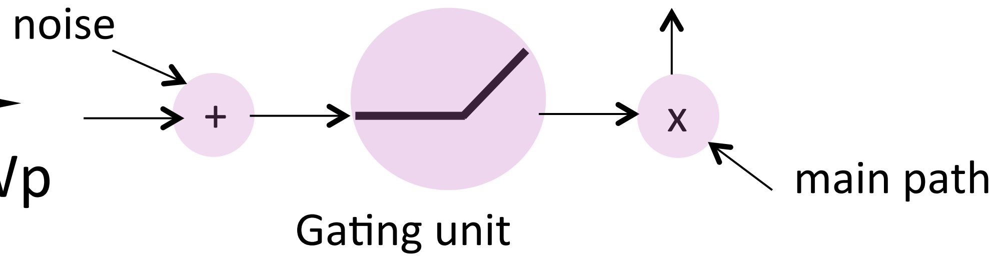
Input

Credit Assignment for Discrete Actions

(Bengio, Leonard, Courville 2013): *Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation*



- Gating units take a hard decision
- Gradient through discrete function = 0
- Solutions in (Bengio, Leonard, Courville 2013):
  - Heuristic back-prop (straight through estimator), also (Gregor et al ICML 2014).
  - Noisy rectifier: ➡
  - Smooth times Stochastic b√p with b ~ Bin(√p)
  - REINFORCE with variance reduction baseline, i.e., RL, i.e. correlate with loss, no back-prop for gaters



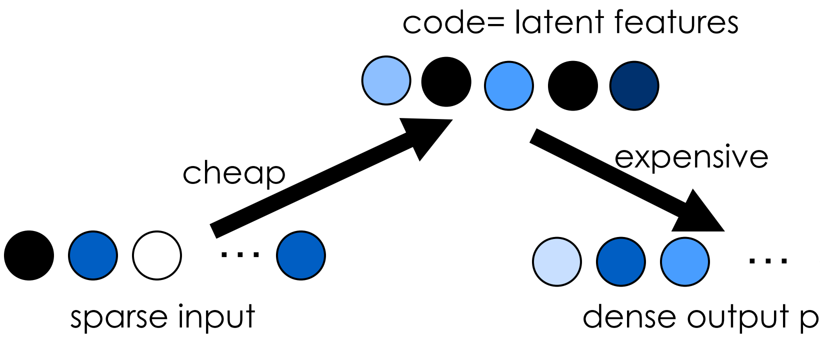
- Another option: train a stochastic credit-assignment machine by Reweighted Wake-Sleep (Bornschein & Bengio 2014)

Conditional Computation on the Output Layer

- When computing the loss *L(f(x),y)*, we can exploit the knowledge of y to make the computation of the loss NOT HAVE TO COMPUTE WITH ALL THE PARAMETERS involved in f(x).
- Example 1: - *log P(y|x)* can be decomposed in a tree structure over the classes *y*, into super-(super-)categories
- Example 2: a sampling approximation of *L(f(x),y)* can be computed that is much cheaper

Handling Large Output Spaces

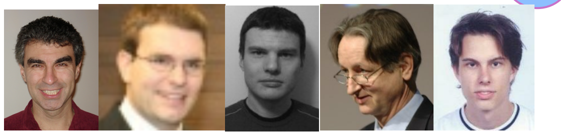
- Auto-encoders and RBMs reconstruct the input, which is sparse and high-dimensional; Language models have a huge output space (1 unit per word).

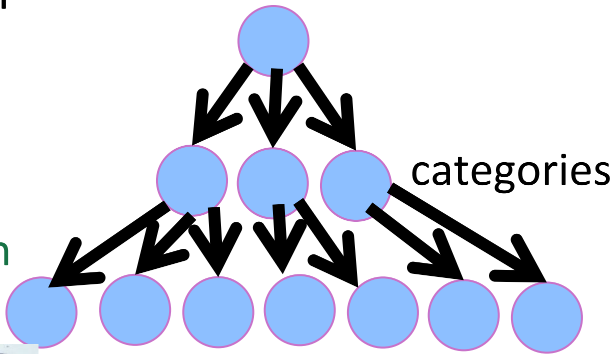


Alternatives to likelihood not requiring the compute the normalization constant, e.g. NCE (Mnih&Kavukcuoglu NIPS 2013)

- (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)





Exploiting Sparsity

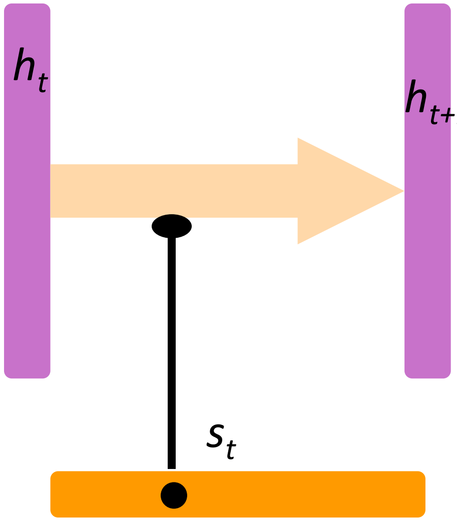
- If *x* is sparse, computing *Wx* only needs to touch the columns associated with non-zero x_i
- Unfortunately, it is more difficult to exploit sparsity on GPUs, especially when the pattern of sparsity is not the same between examples of the same mini-batch (due to lack of caching)
- Implementations using the sparse matrix multiplications with CUDA can be 100x slower than their dense counterparts (on a per-multiply-add basis) while their CPU equivalents can be 10x slower than their dense counterparts.

Sparse inputs + 3-way connections

- Much larger payoff can be obtained when using 3-way connections if the input is very sparse e.g. one-hot code for characters *(Sutskever et al ICML 2011)*

- E.g. Each input symbol s_t selects a different recurrent weight matrix of an RNN

$$h_{t+1} = tanh(b + W_{s_t}\, h_t)$$



Exponentially Exploding the #Parameters for fixed Computation

(Cho & Bengio 2014)

To drastically increase the ratio of parameters to computation, binarize the pattern of activations of a layer to select up to 2^k weight matrices (n x m) for computing the next layer.

Gater: *k* of the *n* units

Memory = parameters: $O(2^k\ n\ m)$

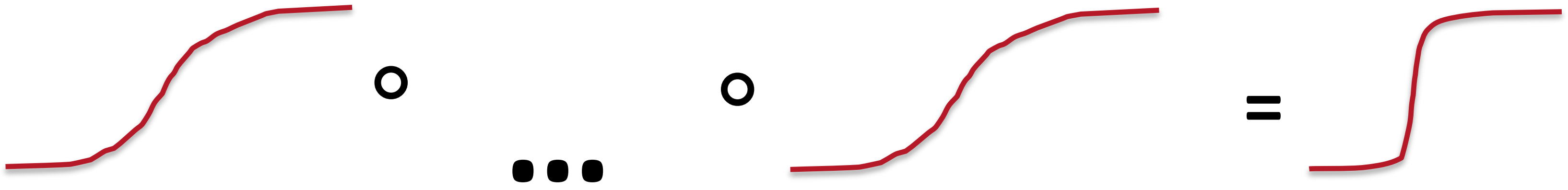
Computation: O($n\ m$)

Many variants are possible (may use different *k* bits for each hidden unit, may add prefix-indexed matrices, etc.)

Scaling up numerical optimization

Issues with Back-Prop

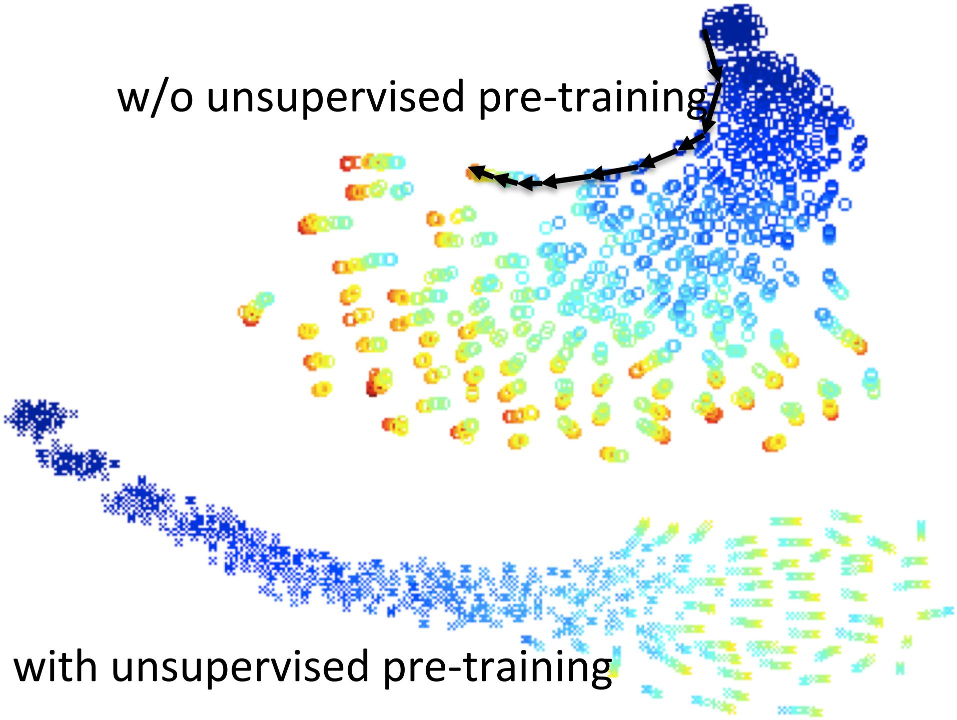
- Over very deep nets or recurrent nets with many steps, non-linearities compose and yield sharp non-linearity → gradients vanish or explode
- Training deeper nets: harder optimization
- In the extreme of non-linearity: discrete functions, can't use back-prop
- Not biologically plausible



Effect of Initial Conditions in Deep Nets

- *(Erhan et al 2009, JMLR)*
- Supervised deep net with vs w/o unsupervised pre-training → very different minima

Neural net trajectories in function space, visualized by t-SNE



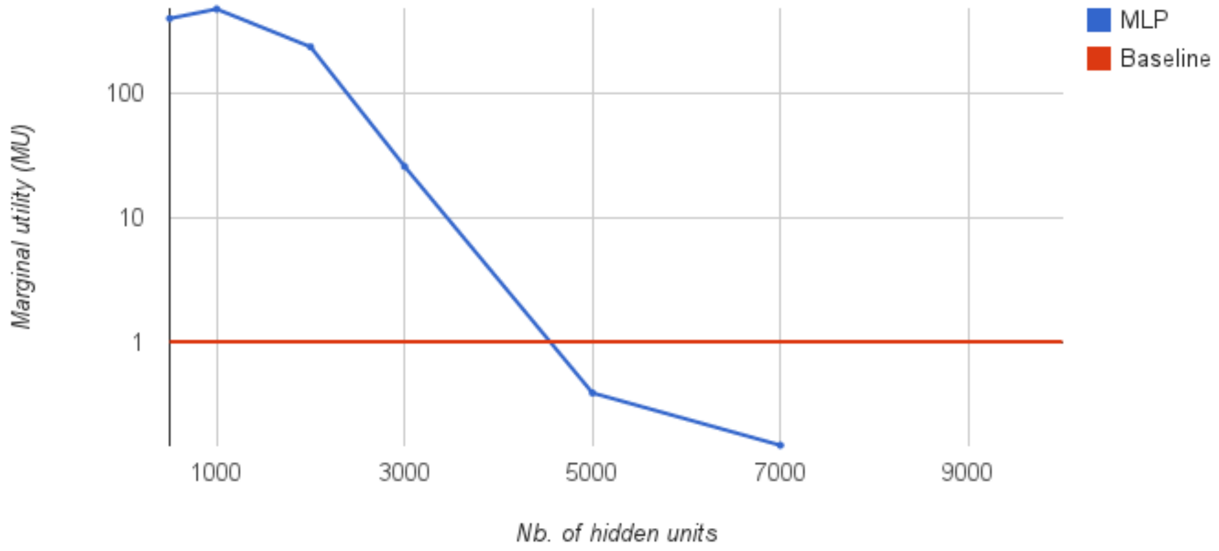
w/o unsupervised pre-training

with unsupervised pre-training

No two training trajectories end up in the same place → huge number of effective local minima

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?

Good News: Piece-wise Linear Activation Functions = Easier Training

- The effects are stronger for deeper nets
- Sigmoid non-linearity worse than Tanh
- Rectifier (max{*0,x*}) better than Tanh
- Maxout ($\max_i (b_i + W_i x)$) better than Rectifier (ReLU)
- Why?
  - Symmetry breaking
  - Less interactions between parameters
  - See *(Glorot et al AISTATS 2010 & 2011, Goodfellow et al ICML'2013)*

- Problem: unstable for very deep nets, impossible for RNNs

Sparse Representations

- Just add a sparsifying penalty on learned representation (prefer 0s in the representation)

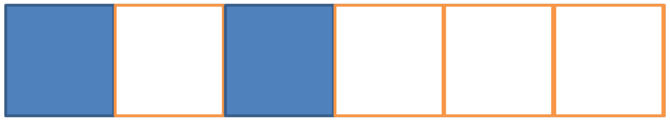
- Information disentangling (compare to dense compression)

- More likely to be linearly separable (high-dimensional space)

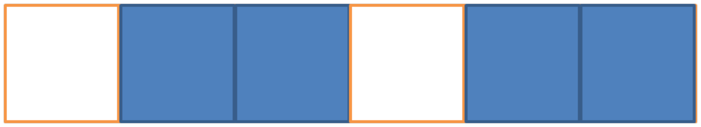
- Locally low-dimensional representation = local chart
- Hi-dim. sparse = efficient variable size representation

  = data structure

Few bits of information



Many bits of information



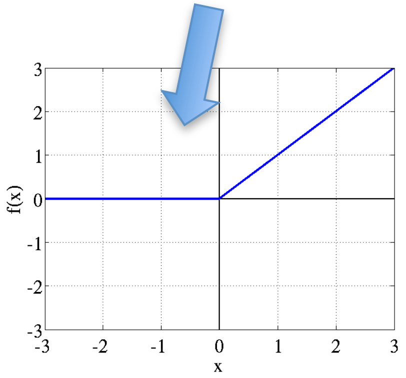
Prior: only few concepts and attributes relevant per example

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

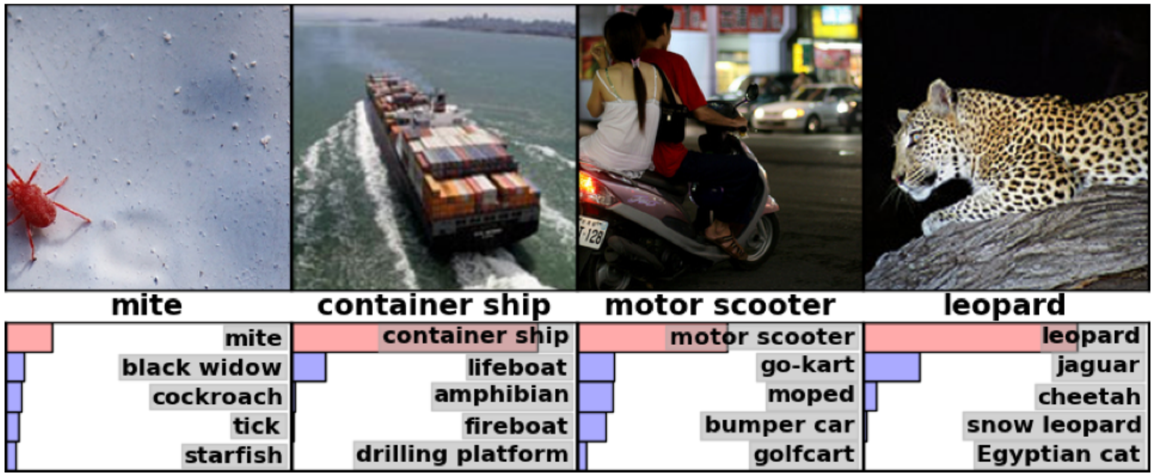


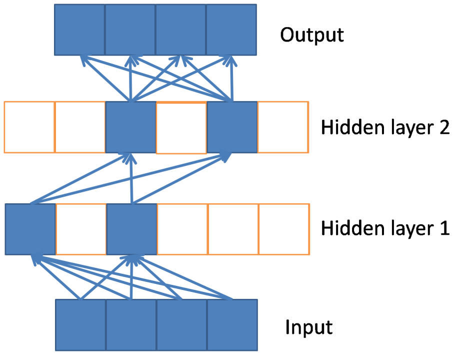
Rectifier

f(x)=max(0,x)

Machine learning motivations

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



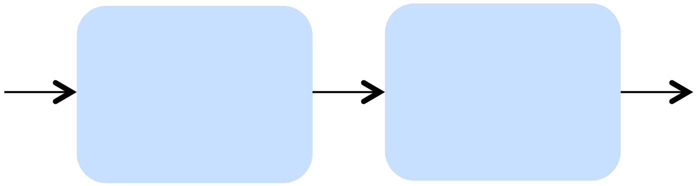


Outstanding results by Krizhevsky et al 2012 killing the state-of-the-art on ImageNet 1000:

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

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



Effective Local Minima

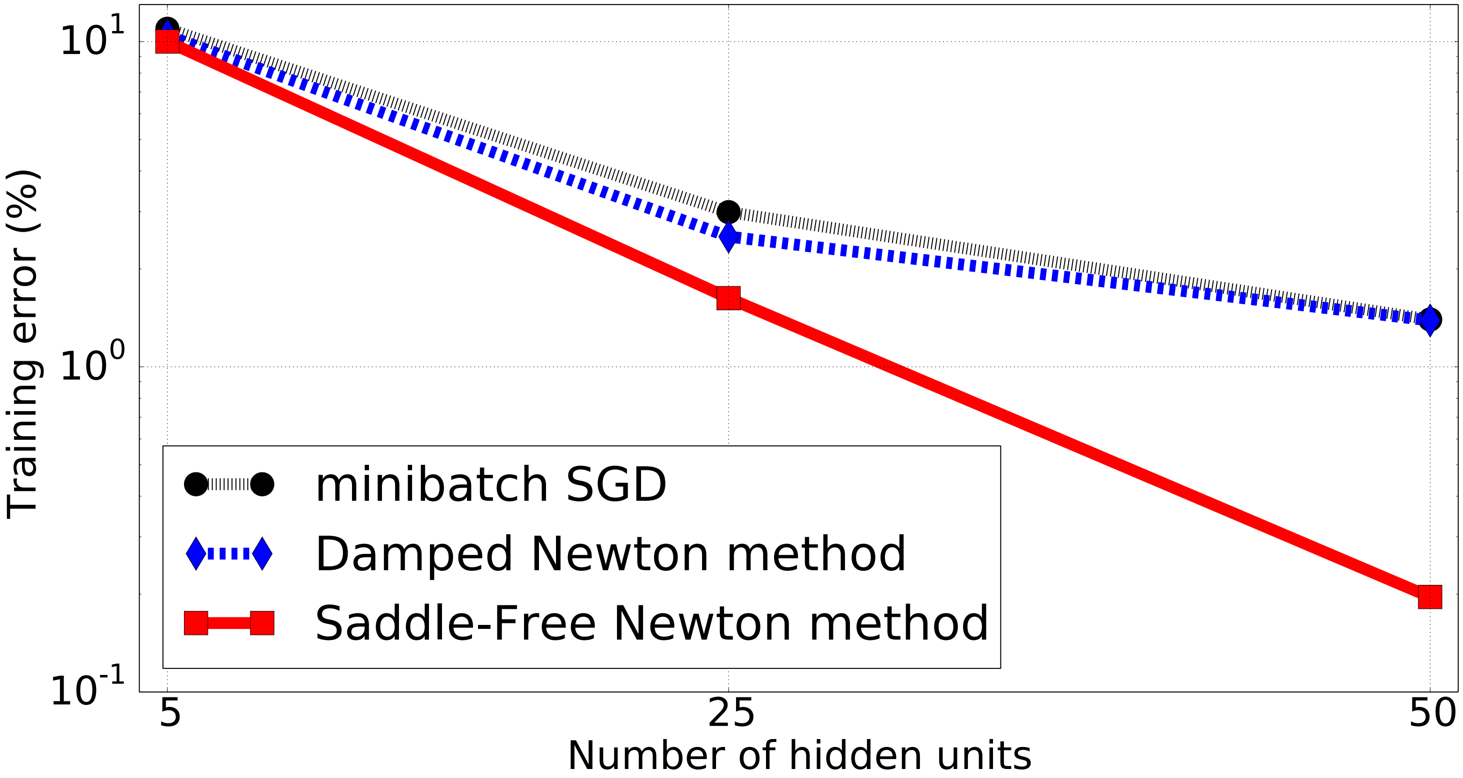
- It is not clear that **actual** local minima are a real issue in training deep nets
  - But initial conditions can sometimes matter a lot!
  - see evidence suggesting instead that saddle points create plateaus that act as obstacles:

    Pascanu et al, On the saddle point problem for non-convex optimization, arXiv 2014

- An optimizer like the one in brains may get stuck → **effective local minima**

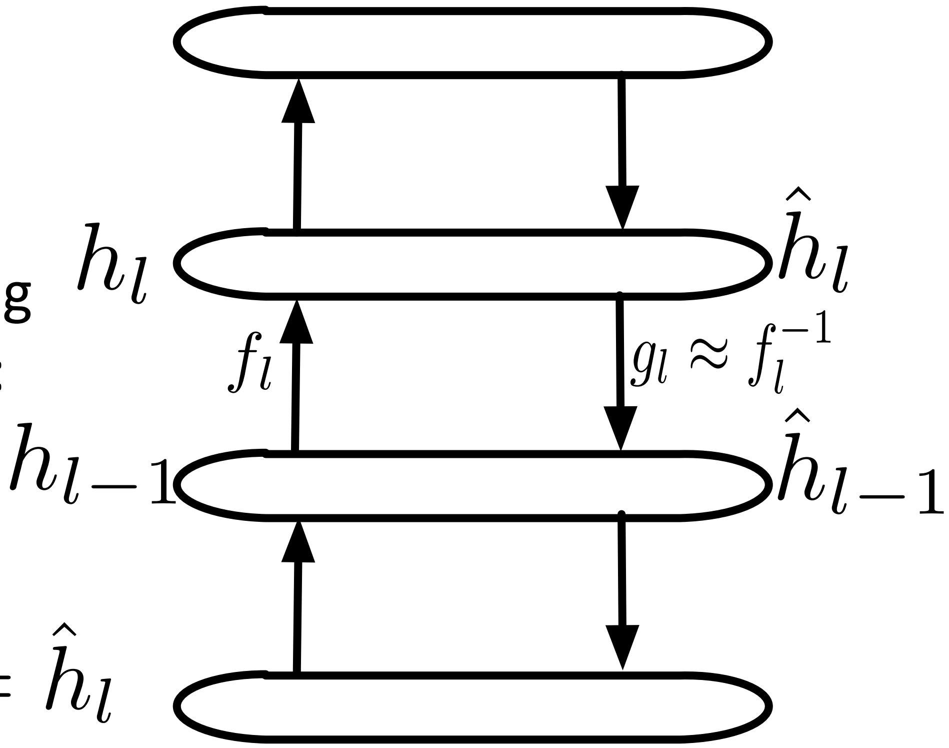
Saddle-Free Optimization
(Pascanu, Dauphin, Ganguli, Bengio 2014)

- Replace eigenvalues λ of Hessian by |λ|



Target Prop
(Bengio 2014, arXiv 1407.7906)

- Instead of propagating the effect of an infinitesimal change, propagate a target that would be
  - Near the original value
  - Yielding to a lower loss
- Can be obtained by maintaining each layer as an auto-encoder:



good target $\hat{h}_{l-1}$ s.t.

$$f_l(\hat{h}_{l-1}) = f_l(g_l(\hat{h}_l)) = \hat{h}_l$$

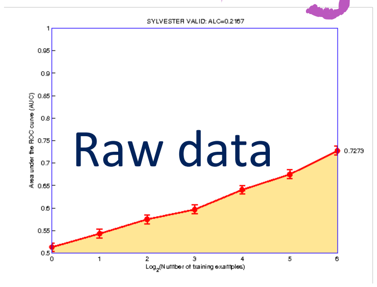
Can potentially deal with highly non-linear or even discrete functions

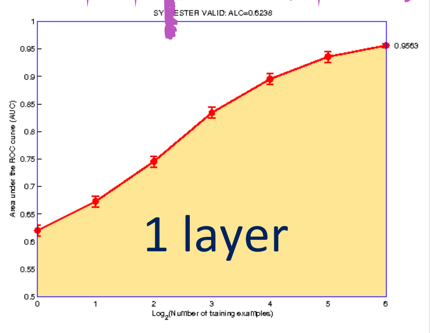
Scaling up Unsupervised & Structured Output Learning

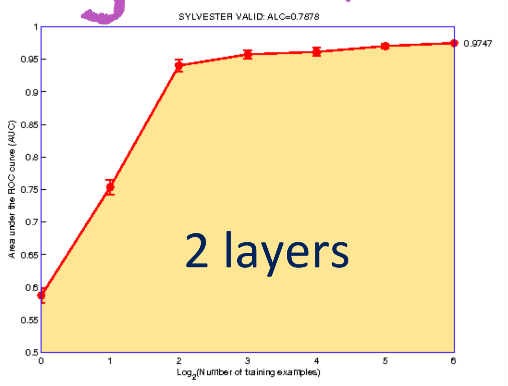
Why Unsupervised Learning?

- 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

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

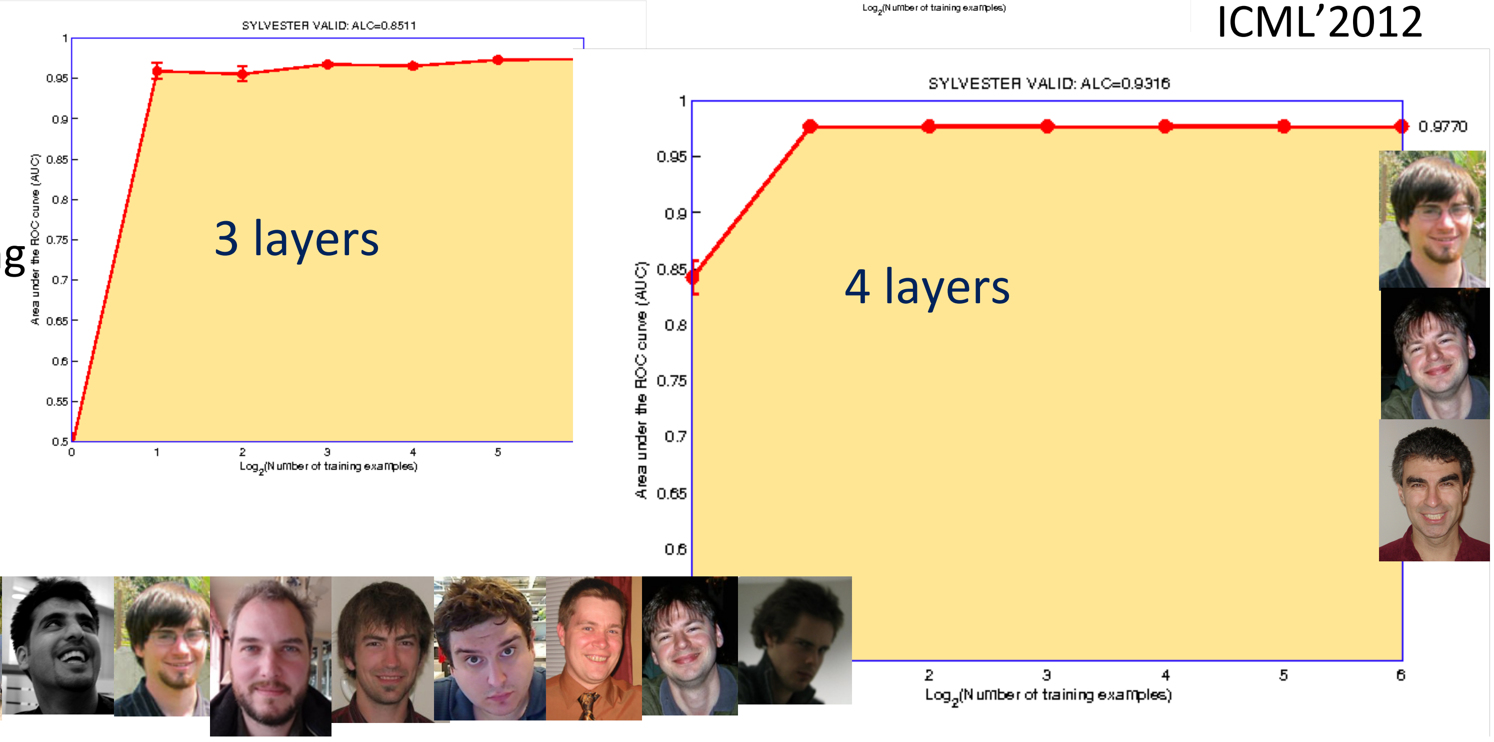


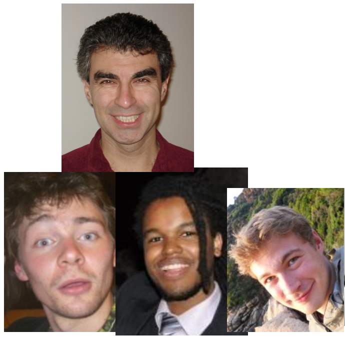




NIPS'2011 Transfer Learning Challenge Paper: ICML'2012

ICML'2011 workshop on Unsup. & Transfer Learning



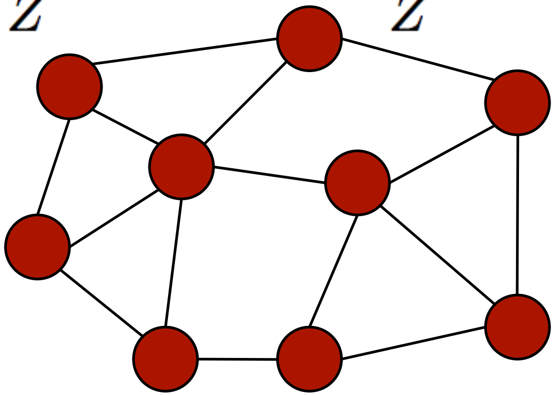


Boltzmann Machines & MRFs

- 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:

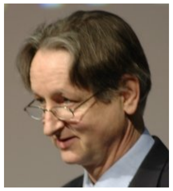


Undirected graphical models

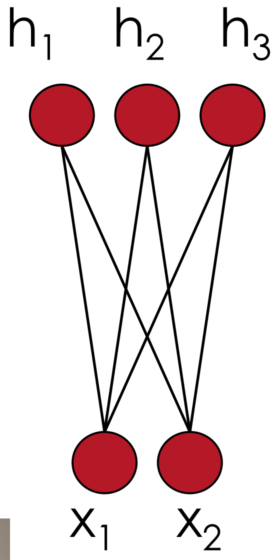
$$P(x) = \frac{1}{Z} e^{\sum_i w_i f_i(x)}$$

Soft constraint / probabilistic statement

- More interesting with latent variables!

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 $P(h_i \mid x)$

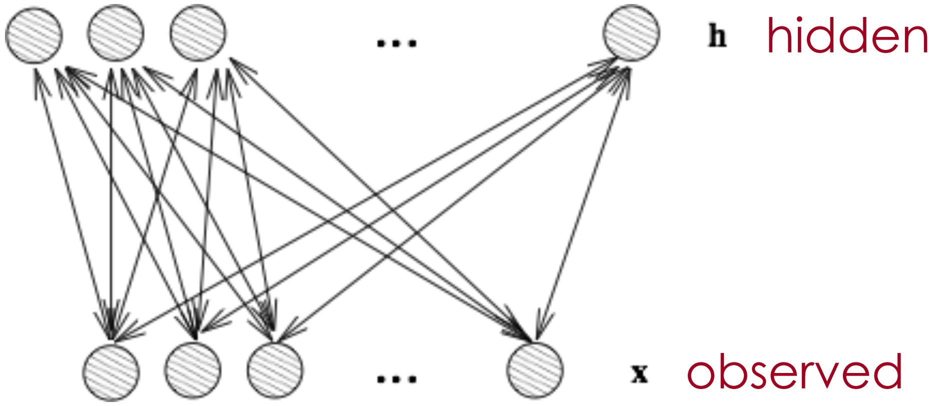
- See Bengio (2009) detailed monograph/review: *"Learning Deep Architectures for AI"*.
- See Hinton (2010) *"A practical guide to training Restricted Boltzmann Machines"*

Restricted Boltzmann Machine (RBM)

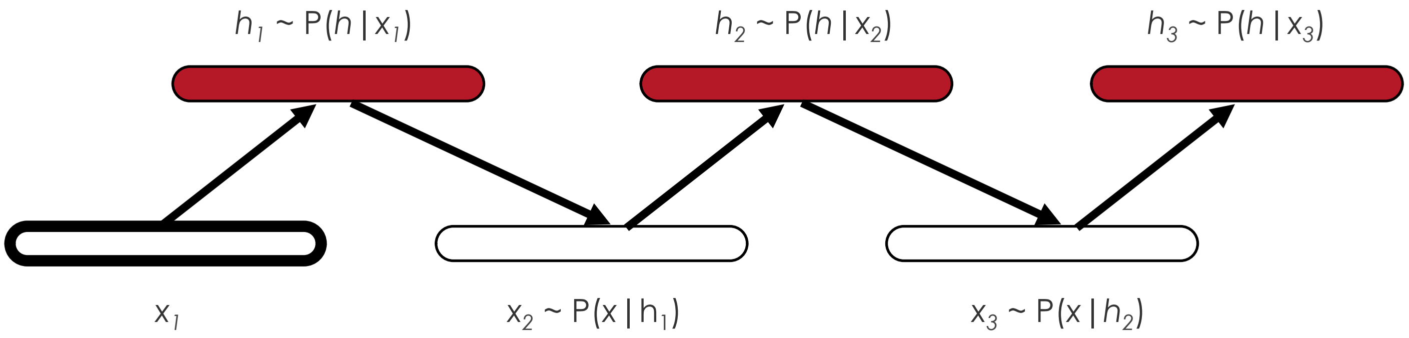
$$P(x,h) = \frac{1}{Z} e^{b^T h + c^T x + h^T W x} = \frac{1}{Z} e^{\sum_i b_i h_i + \sum_j c_j x_j + \sum_{i,j} h_i W_{ij} x_j}$$

- A popular building block for deep architectures

- **Bipartite** undirected graphical model



Block Gibbs Sampling in RBMs



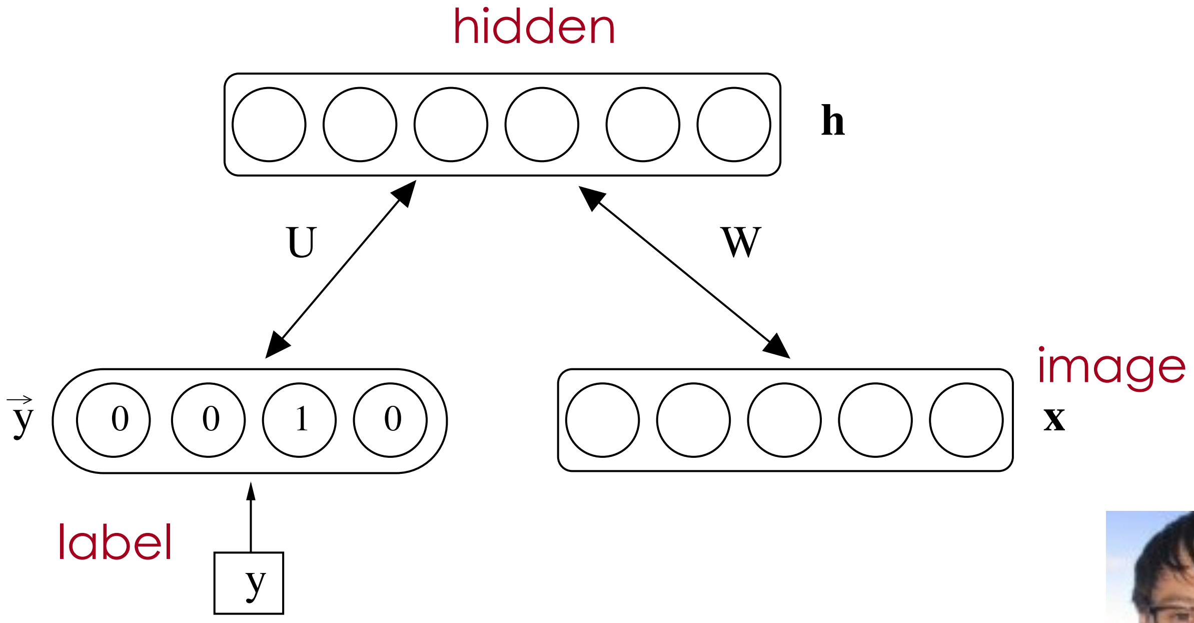
P(h|x) and P(x|h) factorize

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

- Easy inference
- Efficient **block Gibbs** sampling x→h→x→h…

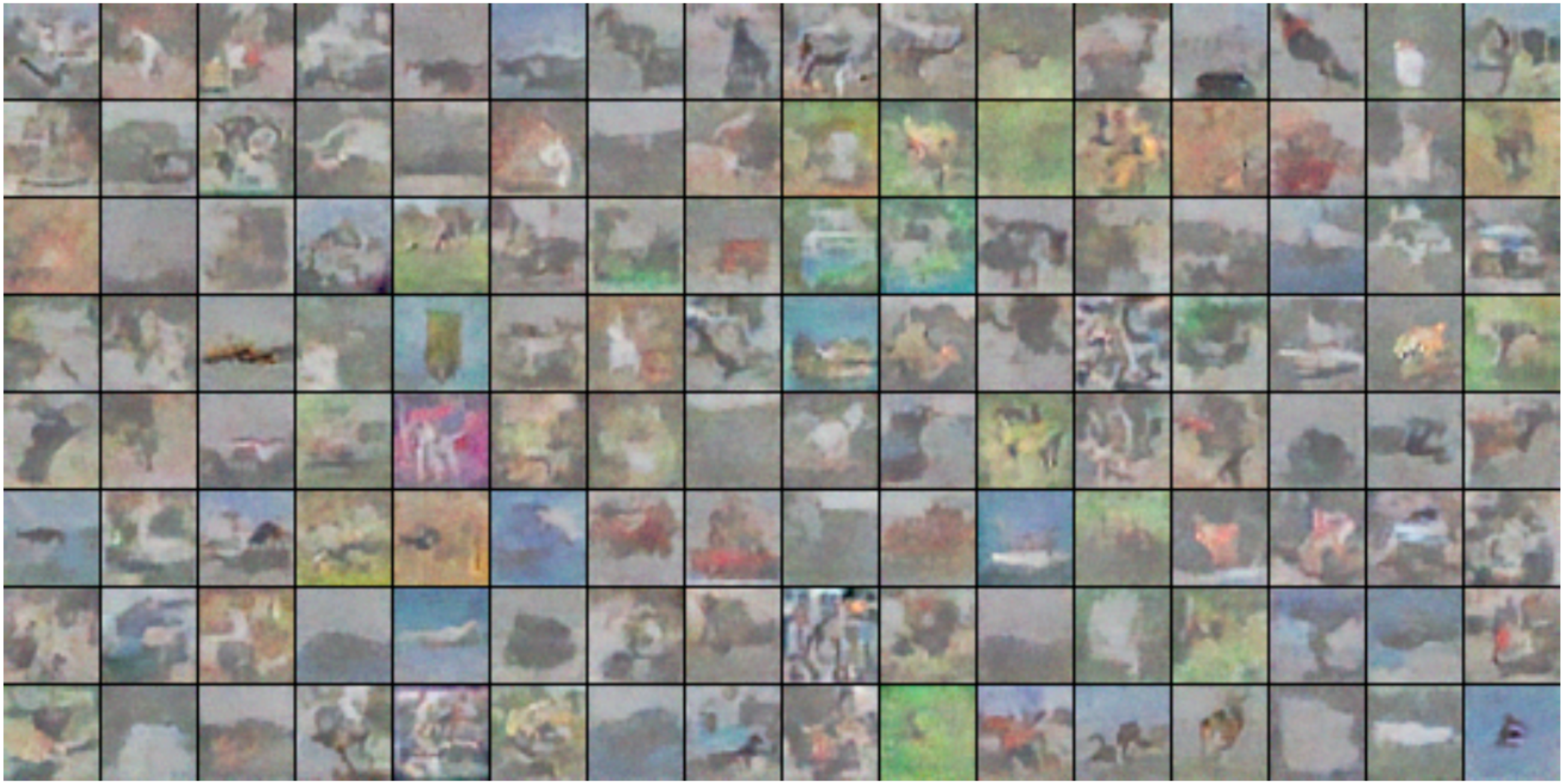
$$P(x,h) = \frac{1}{Z} e^{b^T h + c^T x + h^T W x}$$

RBM with (image, label) visible units



(Larochelle & Bengio 2008)

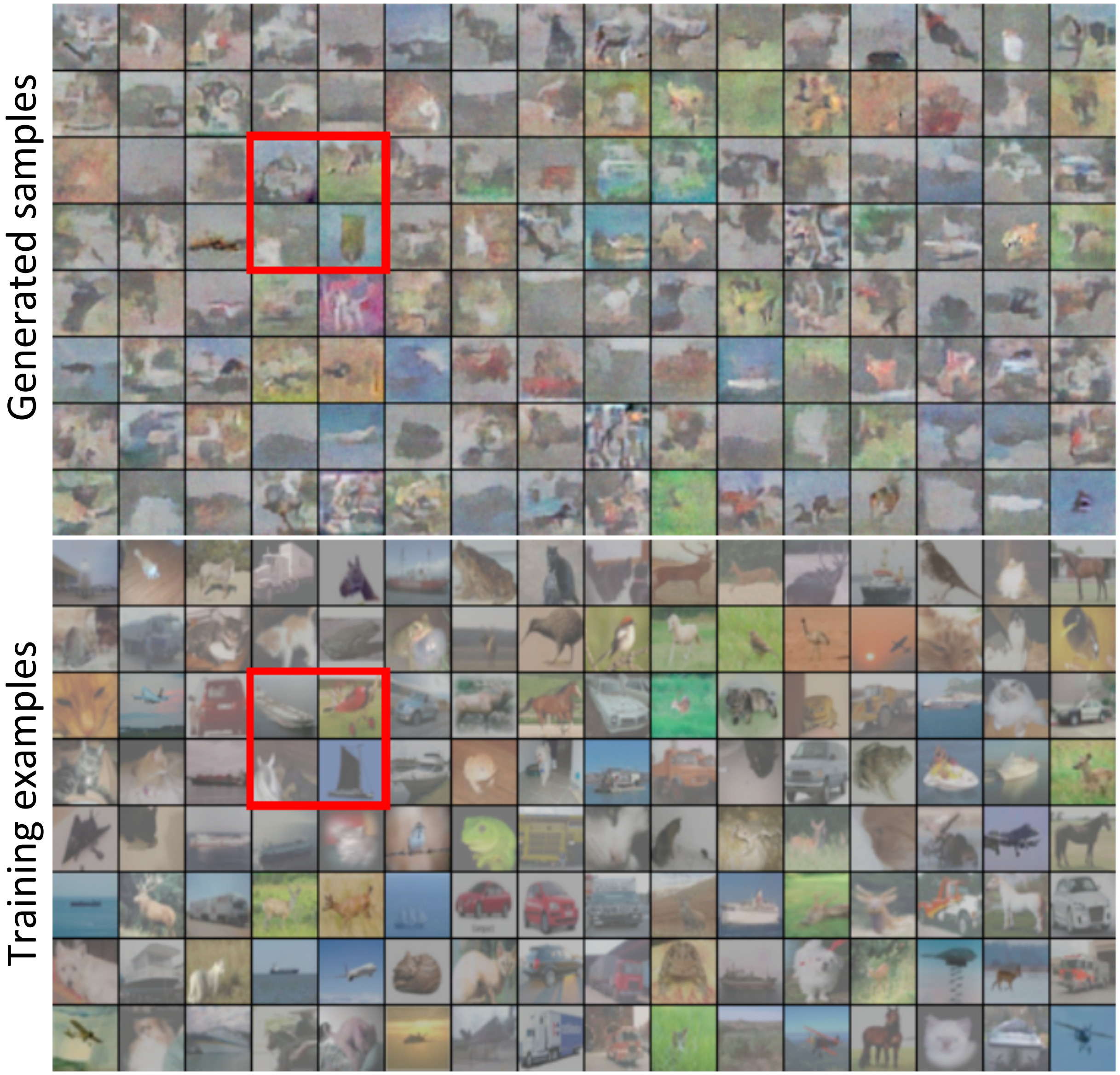
Convolutionally Trained Spike & Slab RBMs Samples



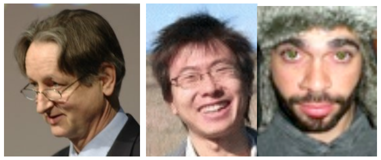
ssRBM is not Cheating

Samples from μ-ssRBM:

Nearest examples in CIFAR:
(least square dist.)

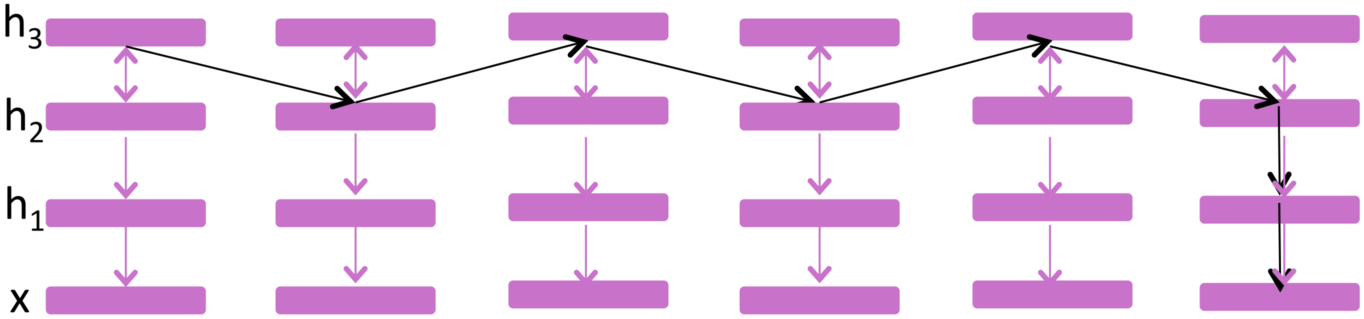


Stack of RBMs → Deep Belief Net



(Hinton et al 2006)

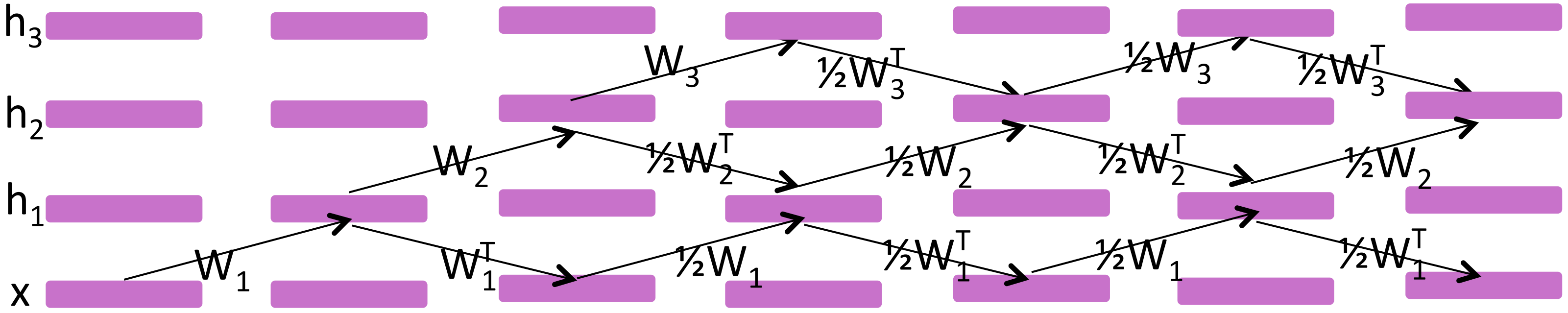
- Stack lower levels RBMs' P(x|h) along with top-level RBM
- P(x, h_1, h_2, h_3) = P(h_2, h_3) P(h_1|h_2) P(x | h_1)
- Sample: Gibbs on top RBM, propagate down



Stack of RBMs
→ Deep Boltzmann Machine

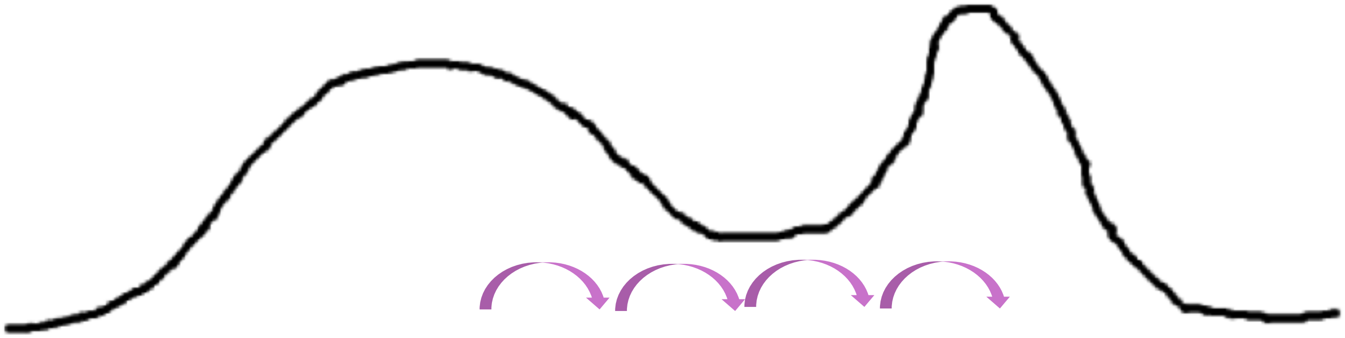
(Salakhutdinov & Hinton AISTATS 2009)

- 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

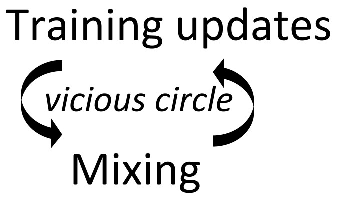


Obstacle: Vicious Circle Between Learning and MCMC Sampling

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



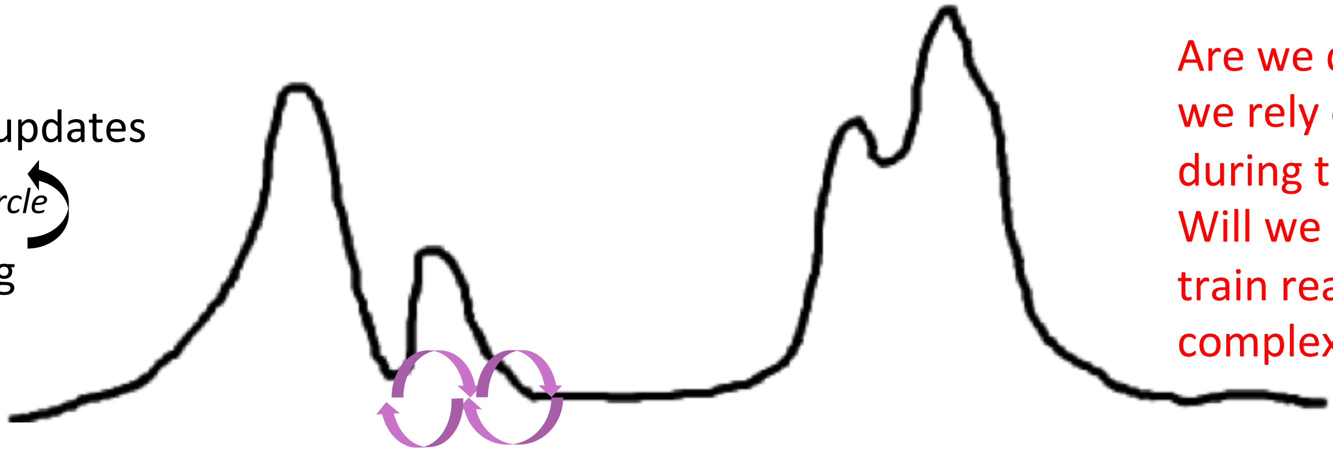
- Later on, hard to cross empty voids between modes



Training updates

vicious circle

Mixing

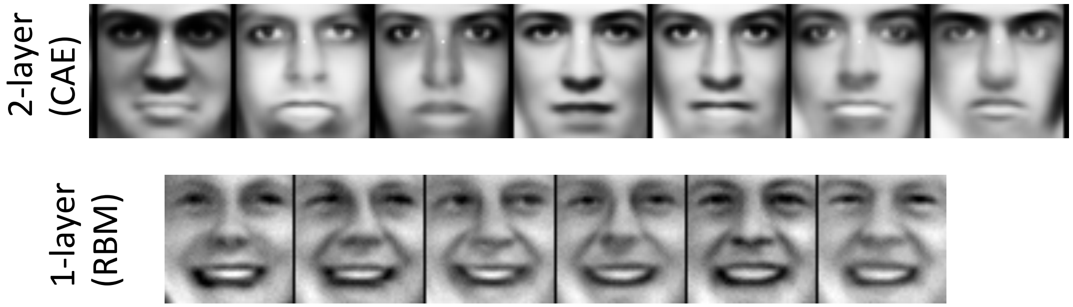


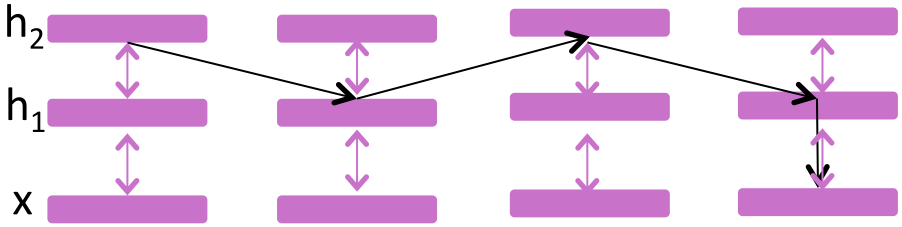
Are we doomed if we rely on MCMC during training? Will we be able to train really large & complex models?

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



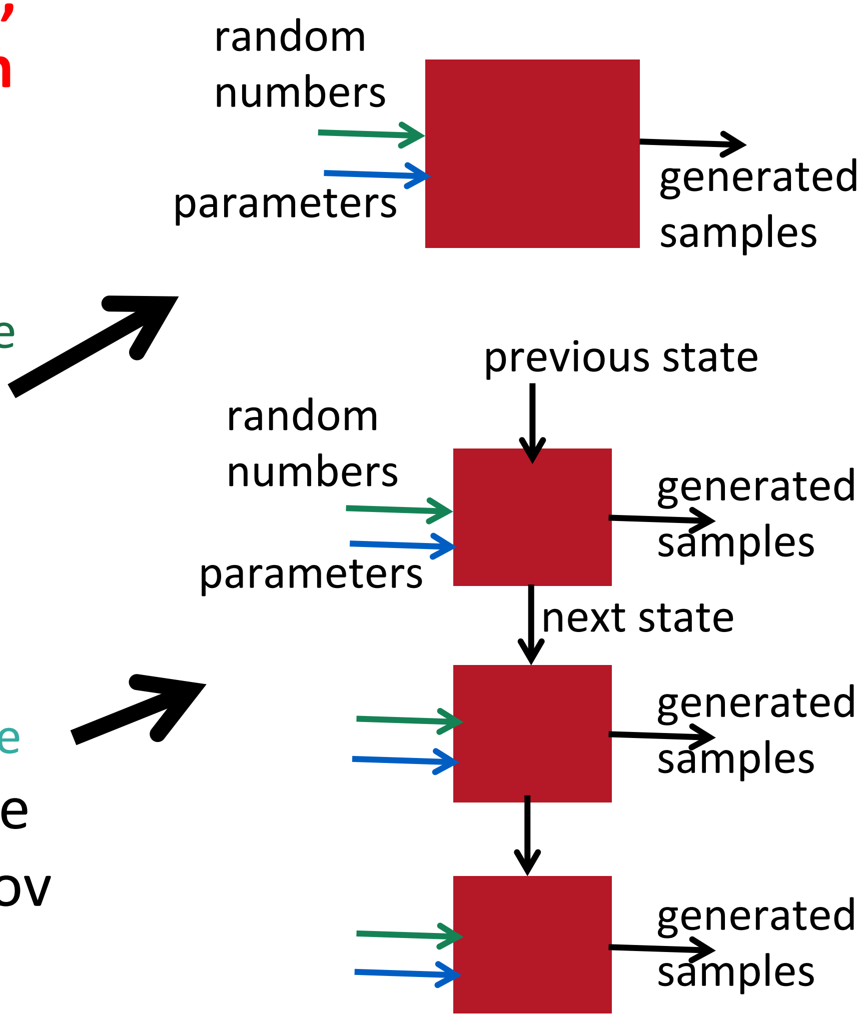


Bypassing Normalization Constants with Generative Black Boxes

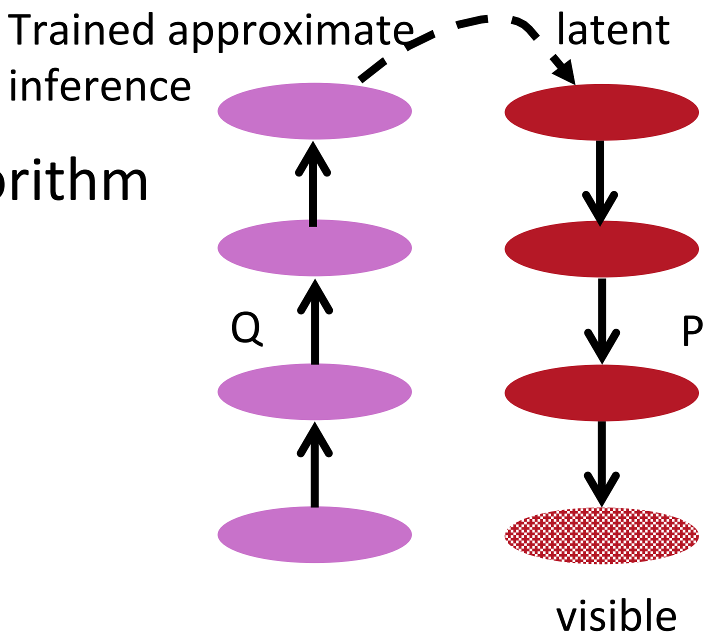
- **Instead of parametrizing p(x), parametrize a machine which generates samples**

- (Goodfellow et al, 2014, Generative adversarial nets) for the case of ancestral sampling in a deep generative net

- (Bengio et al, ICML 2014, Generative Stochastic Networks), learning the transition operator of a Markov chain that generates the data



Ancestral Sampling with Learned Approximate Inference

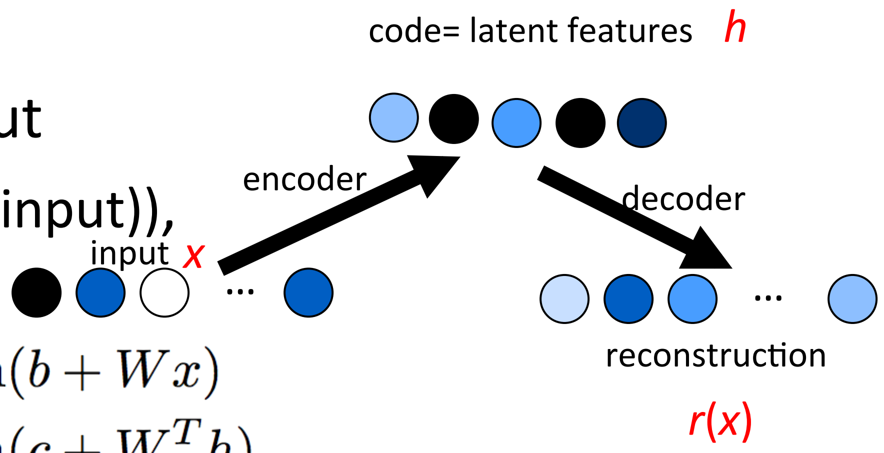


- Helmholtz machine & Wake-Sleep algorithm
  - (Dayan, Hinton, Neal, Zemel 1995)
- Variational Auto-Encoders
  - (Kingma & Welling 2013, ICLR 2014)
  - (Gregor et al ICML 2014)
  - (Rezende et al ICML 2014)
  - (Mnih & Gregor ICML 2014)
- Reweighted Wake-Sleep
  - (Bornschein & Bengio 2014)
- Target Propagation
  - (Bengio 2014)

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

(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)

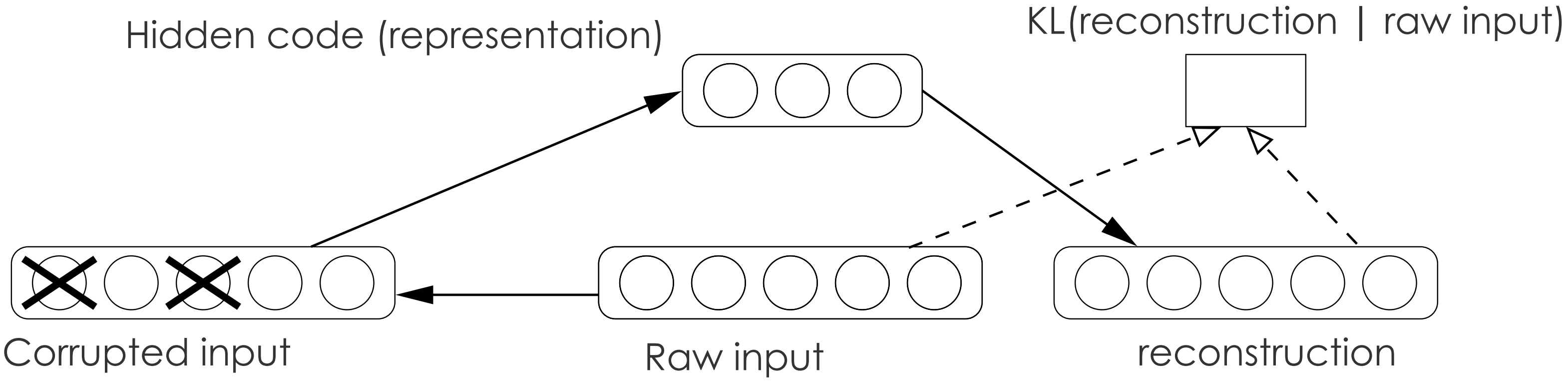
- In general: consider what are appropriate loss functions to predict each of the input variables,

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

Denoising Auto-Encoder

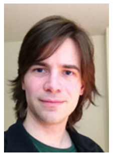
(Vincent et al 2008)

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



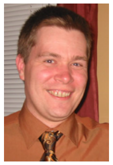
- Encoder & decoder: any parametrization
- As good or better than RBMs for unsupervised pre-training

Denoising Auto-Encoder



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

  r(x)-x ∝ dlogp(x)/dx



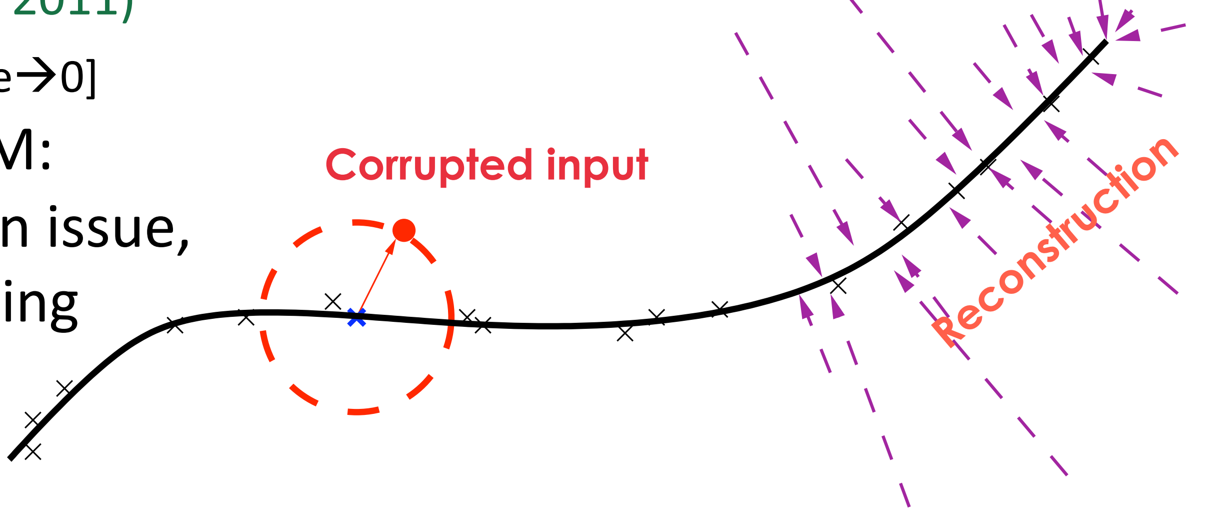
- Some DAEs correspond to a kind of Gaussian RBM with *regularized* Score Matching (Vincent 2011)

  [equivalent when noise→0]

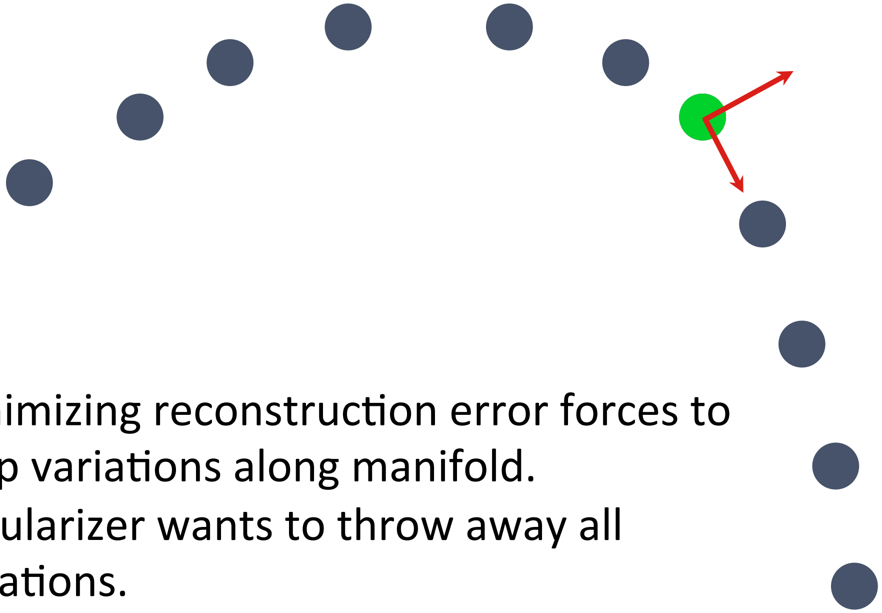
- Compared to RBM:

No partition function issue,
+ can measure training criterion

prior: examples concentrate near a lower dimensional "manifold"

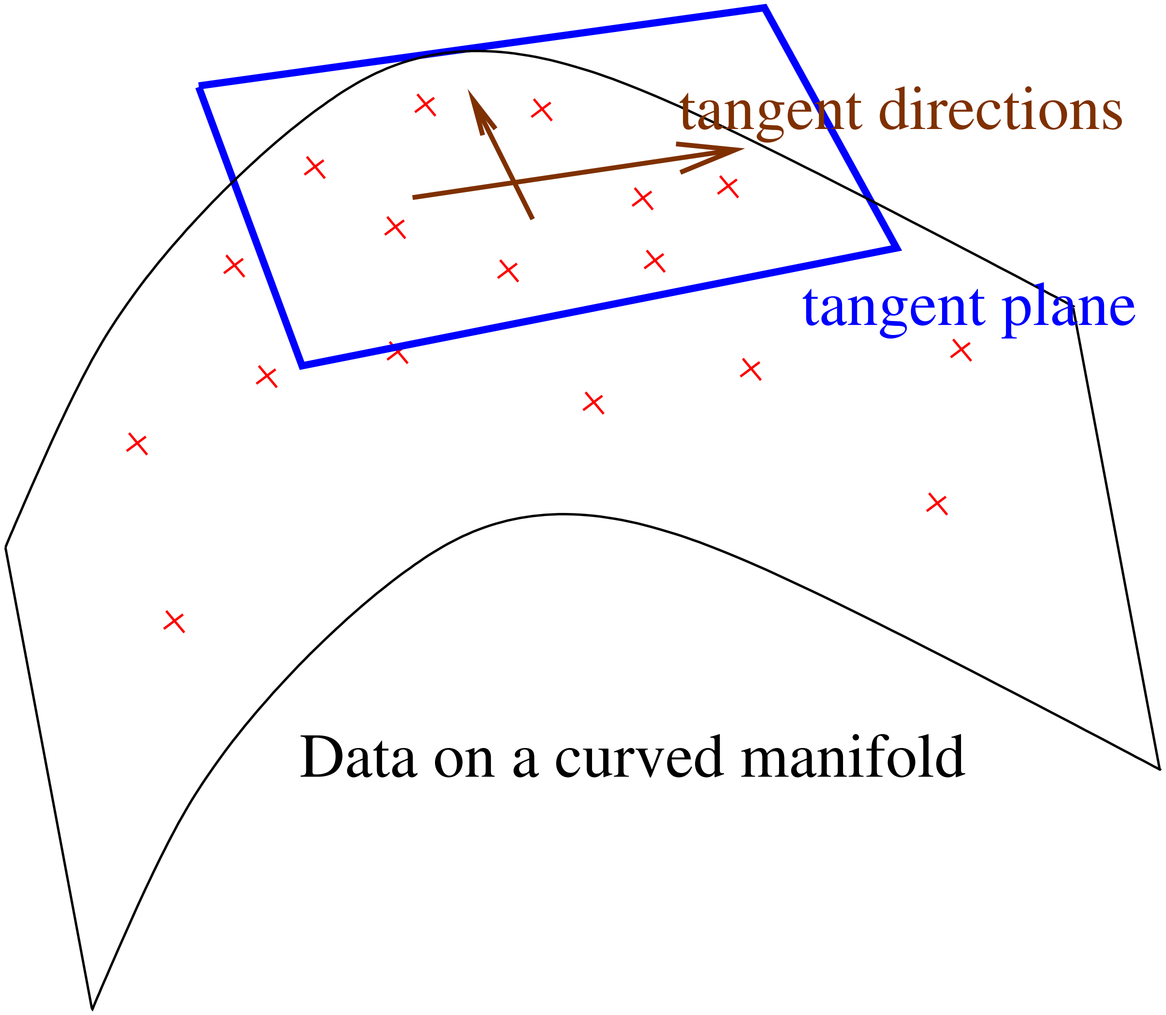


Auto-Encoders Learn Salient Variations, like a non-linear PCA

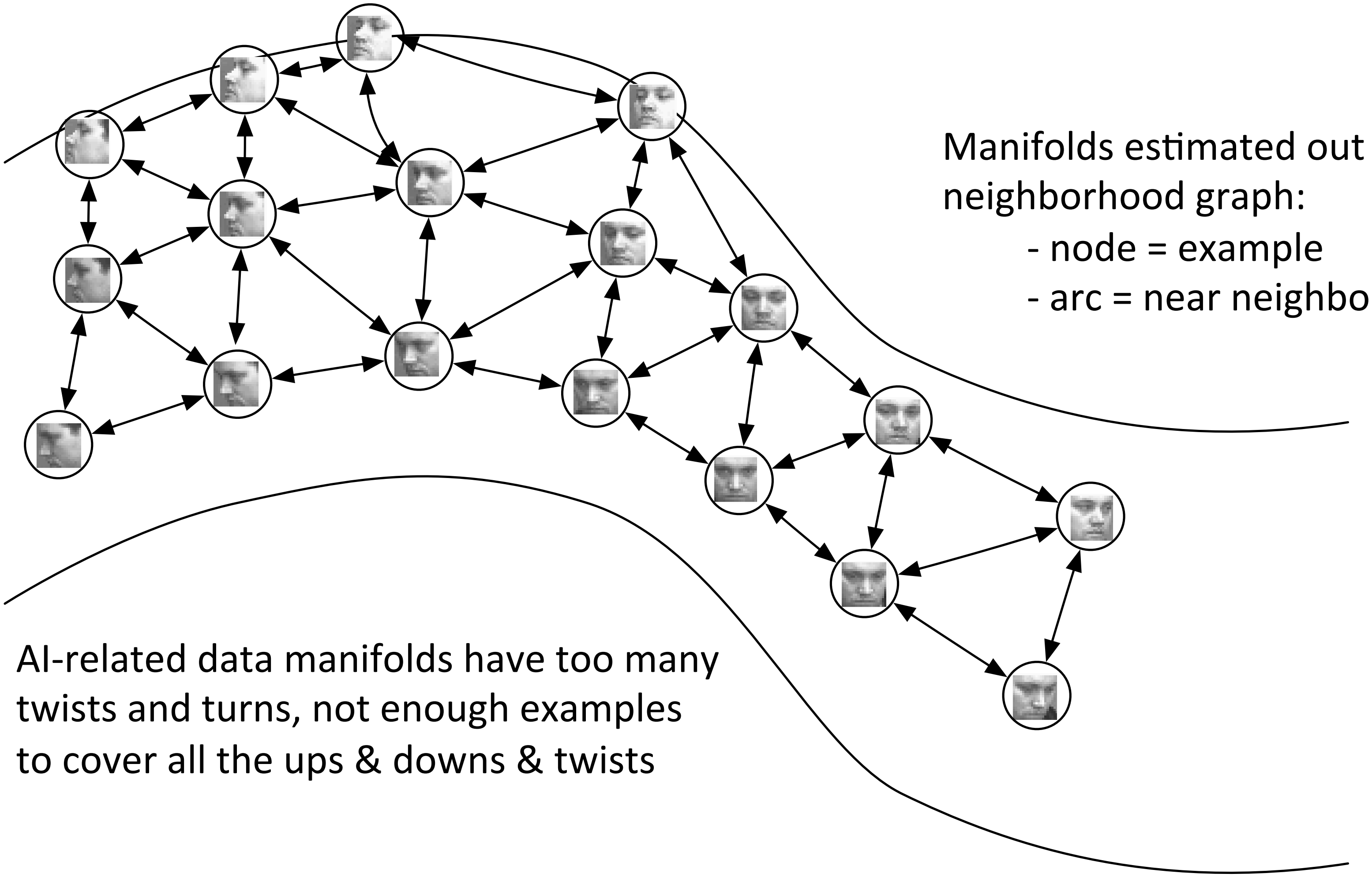


- Minimizing reconstruction error forces to keep variations along manifold.
- Regularizer wants to throw away all variations.
- With both: keep ONLY sensitivity to variations ON the manifold.

Manifold Learning = Representation Learning



Non-Parametric Manifold Learning: hopeless without powerful enough priors



Manifolds estimated out of the neighborhood graph:

- node = example
- arc = near neighbor

AI-related data manifolds have too many twists and turns, not enough examples to cover all the ups & downs & twists

First Theoretical Results on Probabilistic Interpretation of Auto-Encoders

(Vincent 2011, Alain & Bengio 2013)

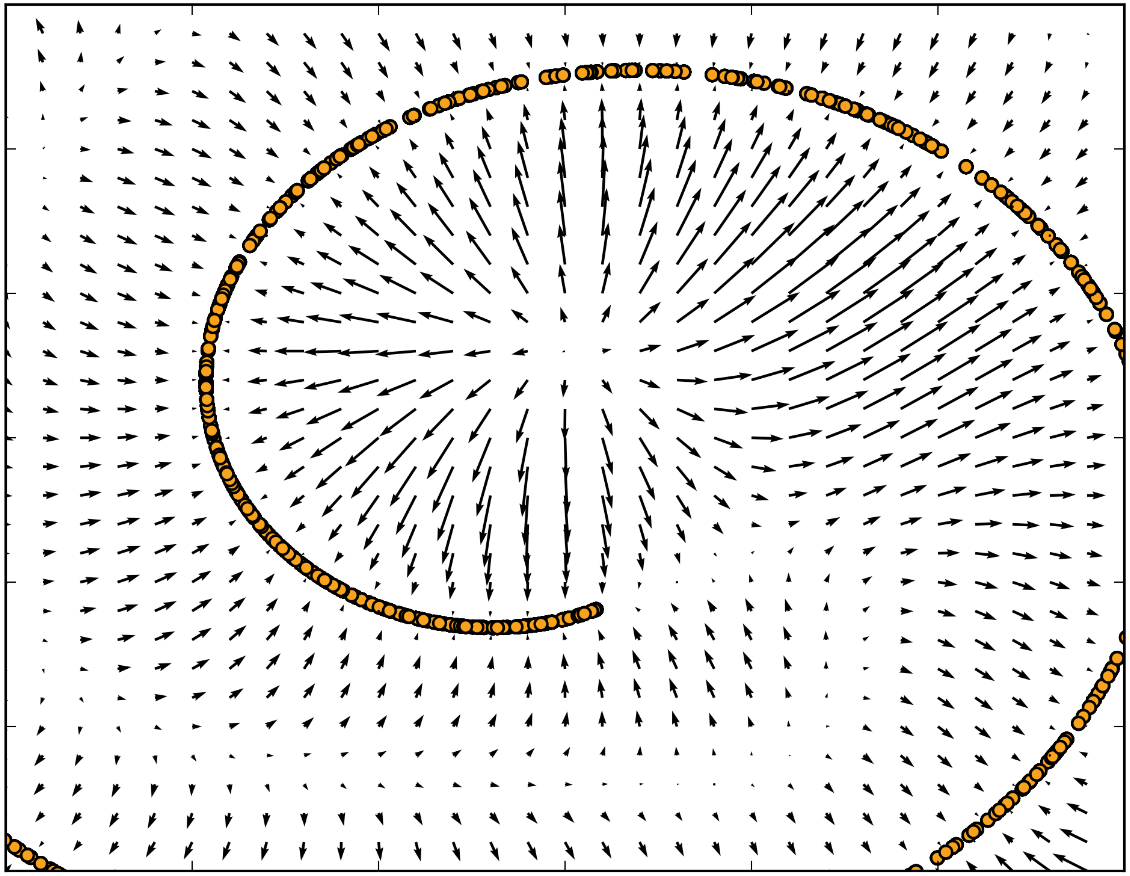
- Continuous *X*
- Gaussian corruption
- Noise σ → 0
- Squared reconstruction error $||r(X\text{+noise})-X||^2$

$(r(X)-X)/\sigma^2$ estimates the score $d \log p(X) / dX$

- Langevin + Metropolis-Hastings can be used to approximately sample from such a model, but mixing was poor

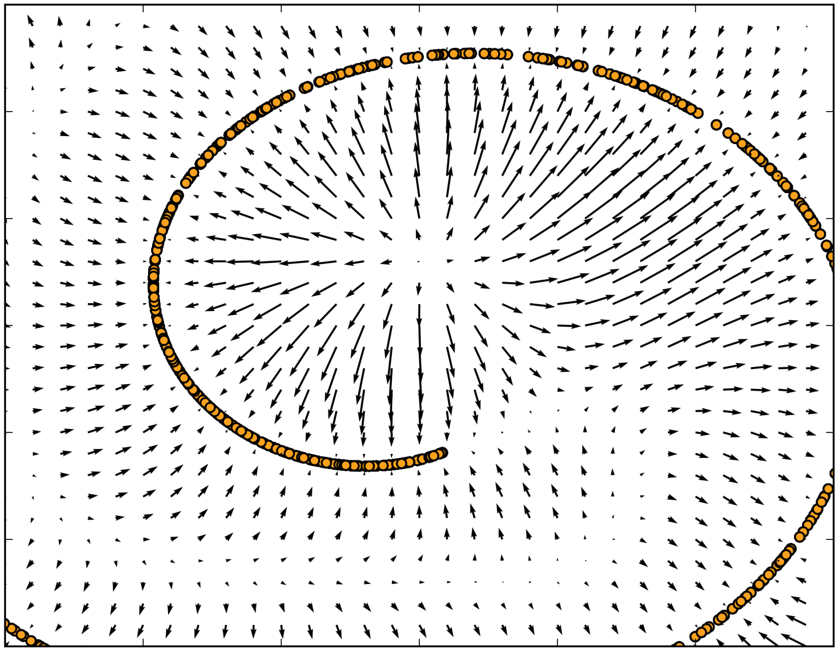
Learning a Vector Field that Estimates a Gradient Field

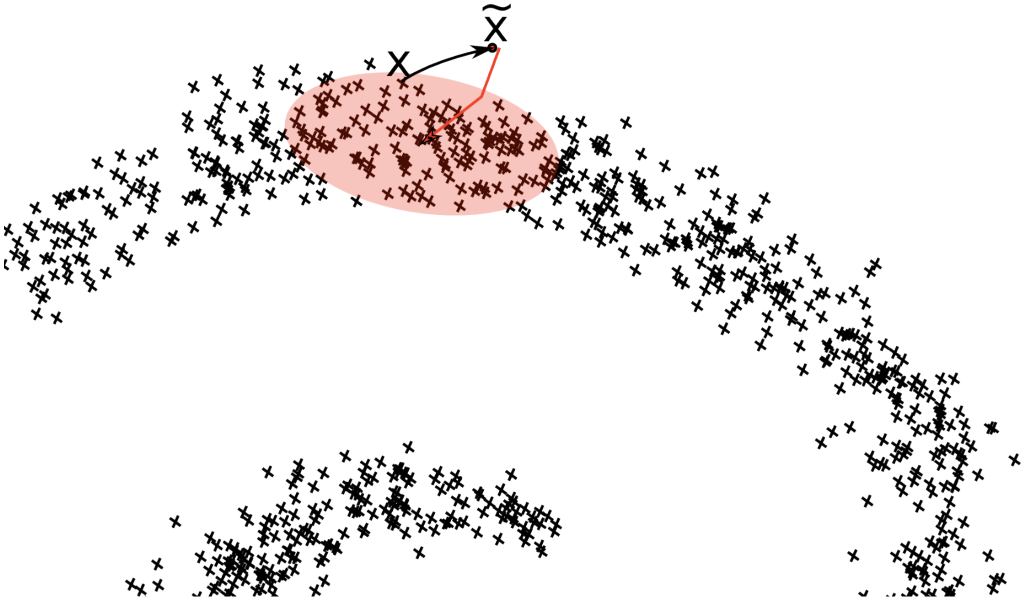
- Continuous inputs
- Gaussian corruption
- Squared error
- Reconstruction(x)-x estimates dlogp(x)/dx
- Zero reconstruction error could be either local min or local max of density



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)





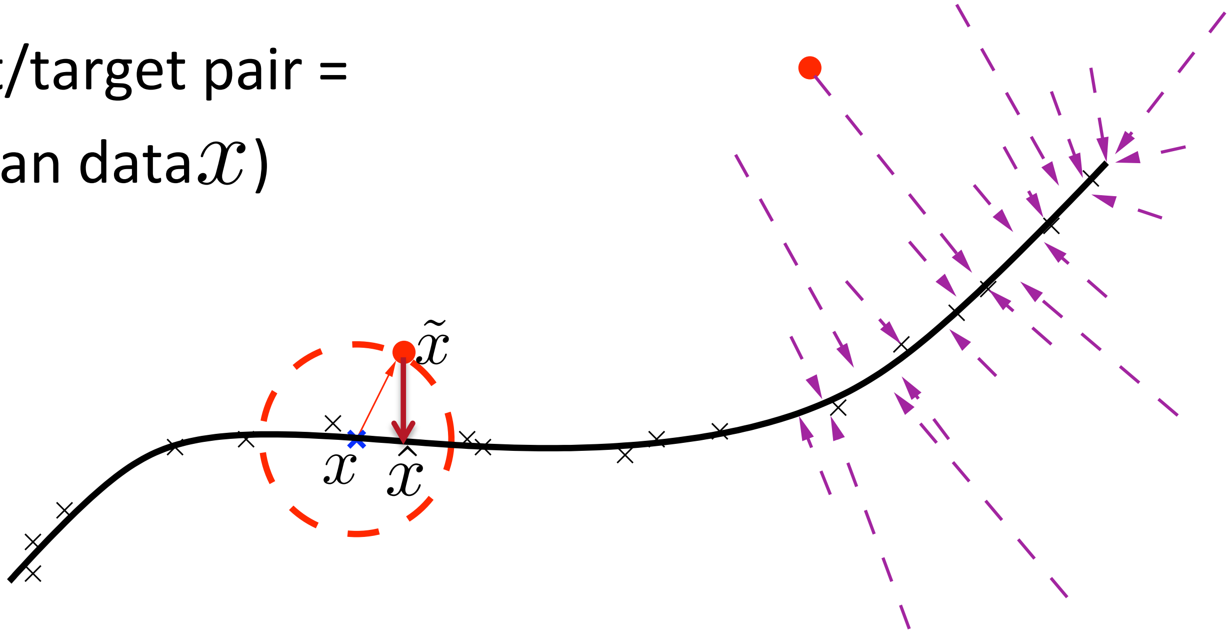
Denoising Auto-Encoders Learn a Small Move Towards Higher Probability

- Reconstruction $\hat{x}$ points in direction of higher probability

$$\hat{x} - x \propto \frac{\partial \log P(x)}{\partial x}$$

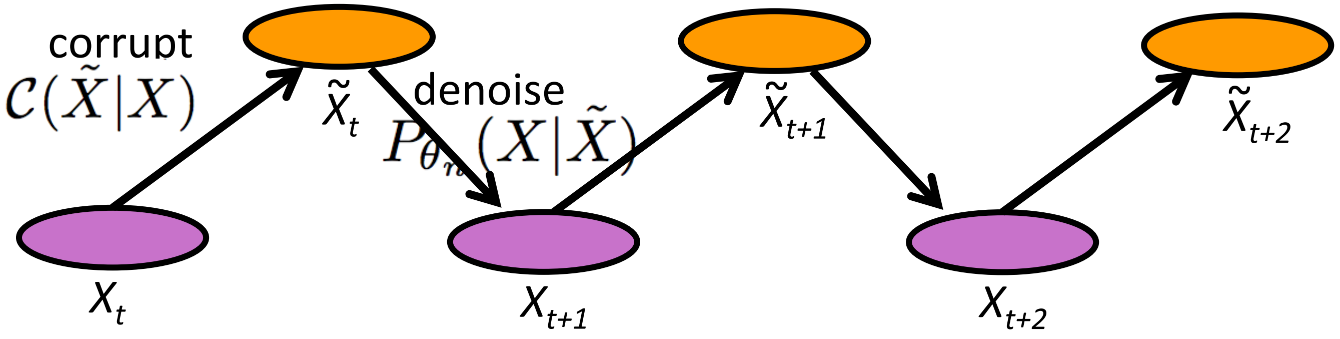
gradient

- Trained with input/target pair = (corrupted $\tilde{x}$ → clean data x)



Denoising Auto-Encoder Markov Chain (Bengio et al NIPS'2013)

- $\mathcal{P}(X)$: 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 $\tilde{X} \sim \mathcal{C}(\tilde{X}|X)$ $X \sim P_{\theta_n}(X|\tilde{X})$



- Theorem: stationary distribution of T_n estimates $\mathcal{P}(X)$, converges to it for $n \rightarrow \infty$ if DAE converges to $P(X|\tilde{X})$

How do humans generalize from very few examples?

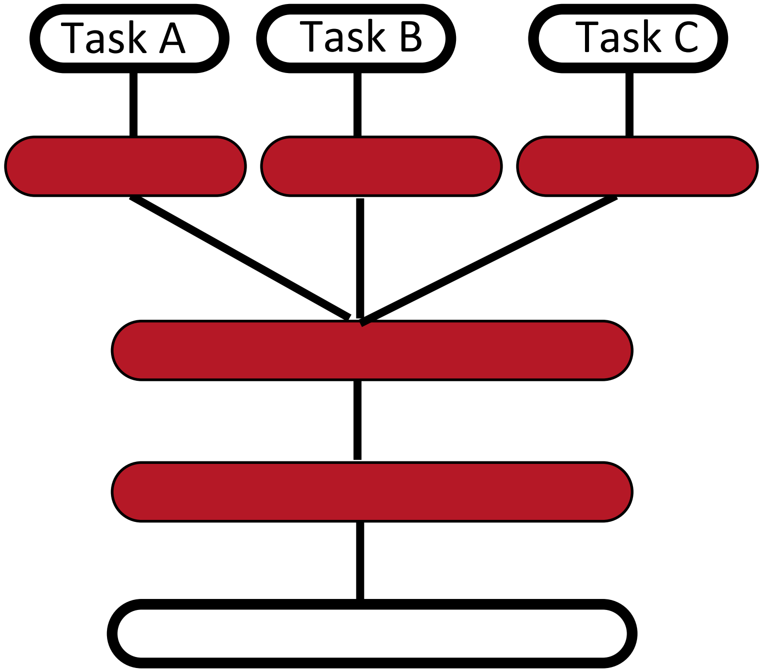
- They **transfer** knowledge from previous learning:
  - Representations
  - Explanatory factors
- Previous learning from: unlabeled data

  + labels for other tasks
- **Prior: shared underlying explanatory factors, in particular between P(x) and P(Y|x)**

Transfer Learning & 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**

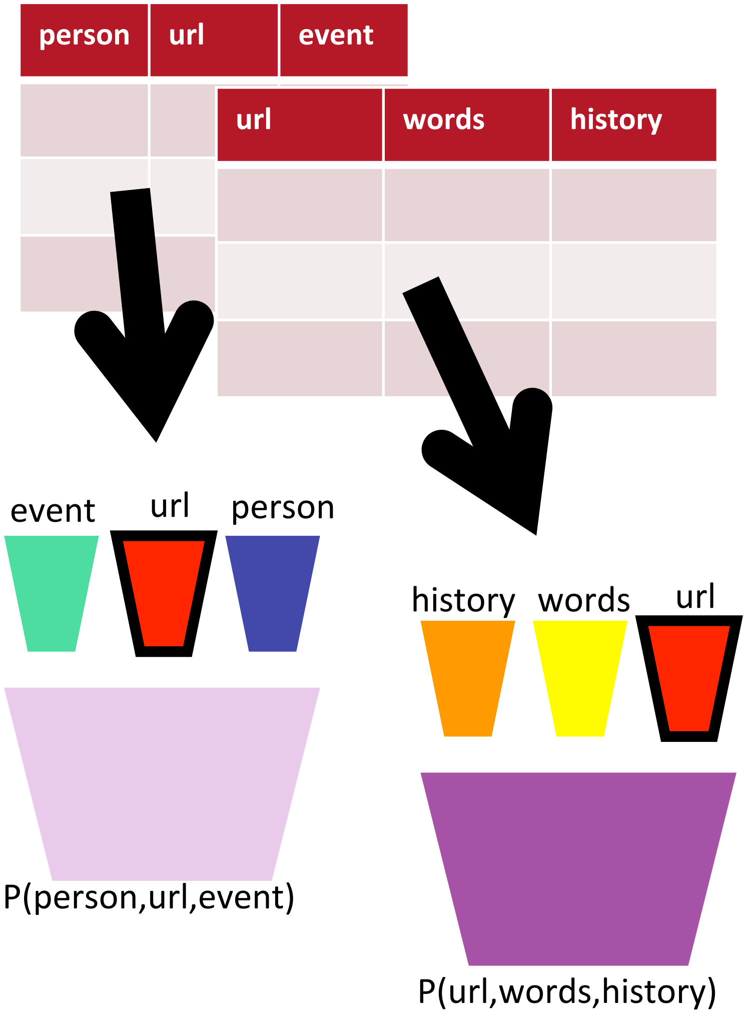


E.g. dictionary, with intermediate concepts re-used across many definitions

Prior: shared underlying explanatory factors between tasks

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

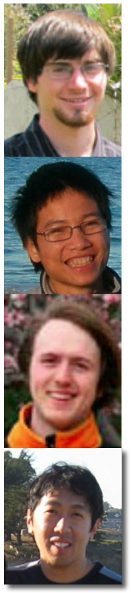
- Which invariances?
- Alternative: learning to disentangle factors
- Good disentangling →

  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?

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

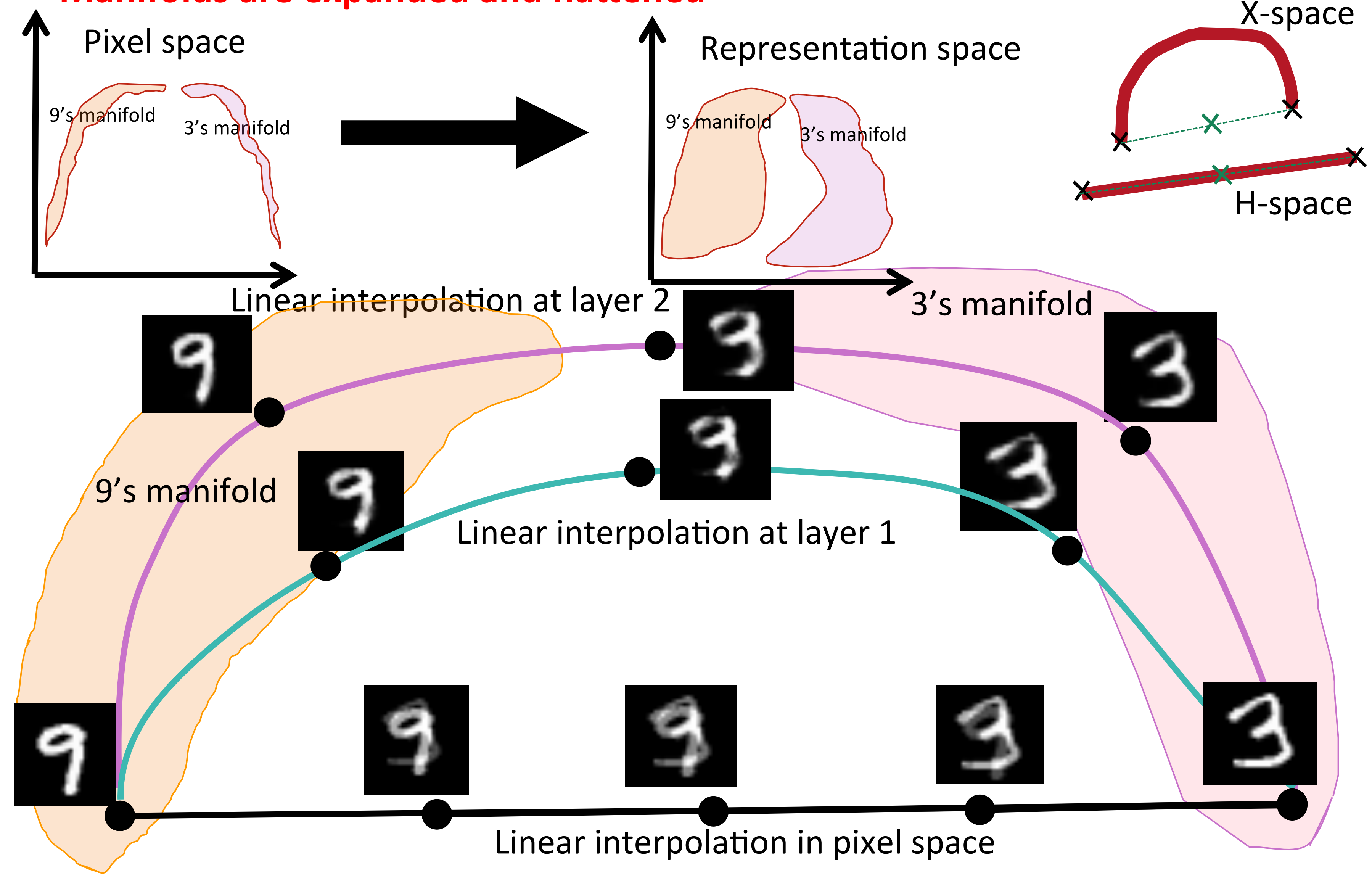
- Exploit those **hints** to **disentangle** better!
- (Becker & Hinton 1993, Wiskott & Sejnowski 2002, Hurri & Hyvarinen 2003, Berkes & Wiskott 2005, Mobahi et al 2009, Bergstra & Bengio 2009)

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)

Space-Filling in Representation-Space

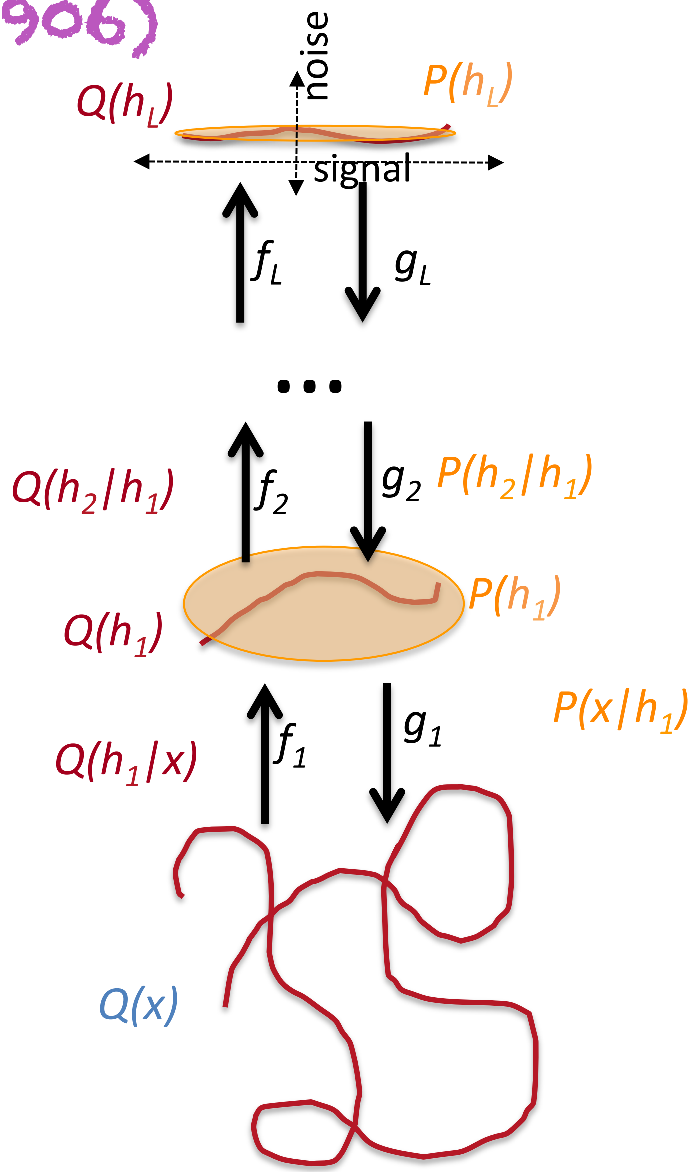
- **Deeper representations ➔ abstractions ➔ disentangling**
- **Manifolds are expanded and flattened**



Extracting Structure By Gradual Disentangling and Manifold Unfolding (Bengio 2014, arXiv 1407.7906)

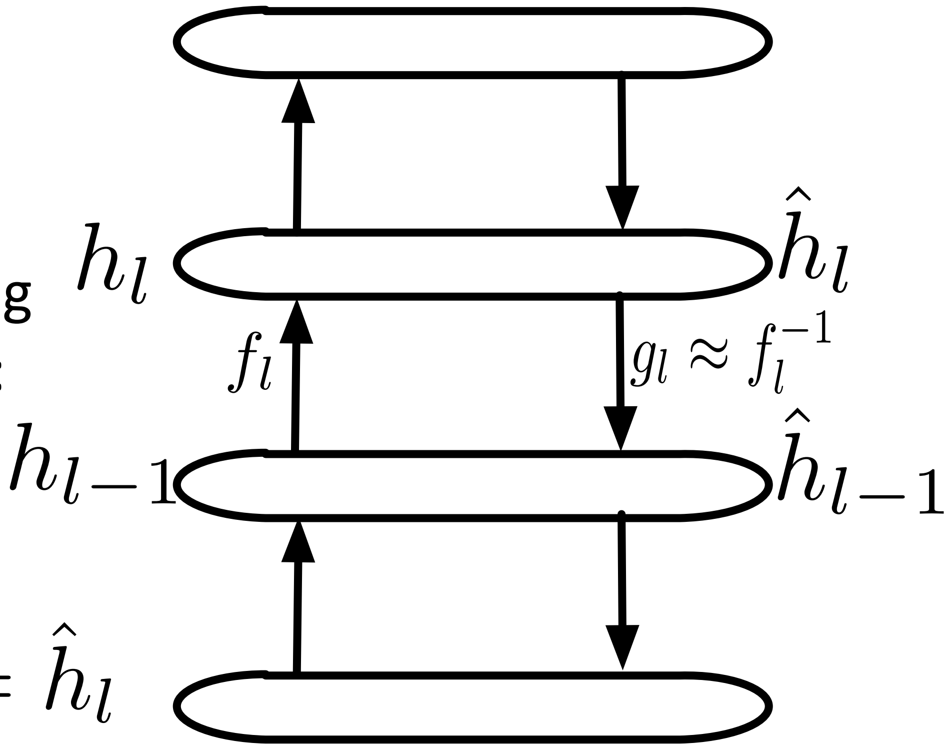
Each level transforms the data into a representation in which it is easier to model, unfolding it more, contracting the noise dimensions and mapping the signal dimensions to a factorized (uniform-like) distribution.

$$\min KL(Q(x,h)||P(x,h))$$



Target Prop
(Bengio 2014, arXiv 1407.7906)

- Instead of propagating the effect of an infinitesimal change, propagate a target that would be
  - Near the original value
  - Yielding to a lower loss
- Can be obtained by maintaining each layer as an auto-encoder:



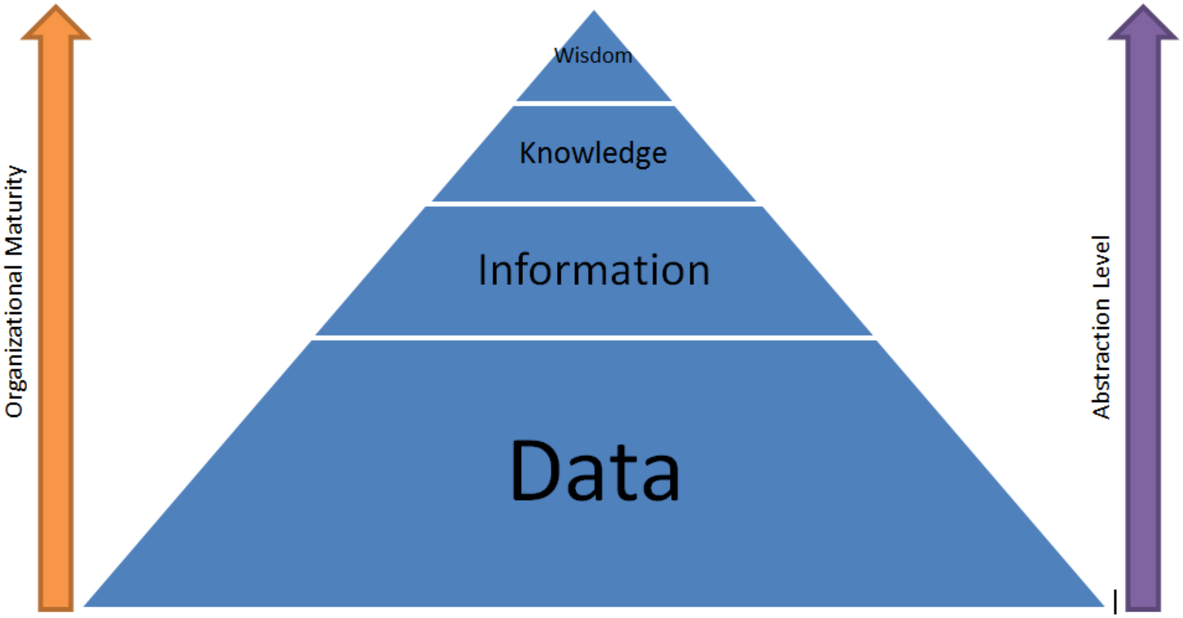
good target $\hat{h}_{l-1}$ s.t.

$$f_l(\hat{h}_{l-1}) = f_l(g_l(\hat{h}_l)) = \hat{h}_l$$

Can potentially deal with highly non-linear or even discrete functions

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 become a crucial machine learning tool:
  - Int. Conf. on Learning Representation 2013 & 2014 a huge success!

    Conference & workshop tracks, open to new ideas ☺
- Industrial applications (Google, IBM, Microsoft, Baidu, Facebook, Samsung, Yahoo, Intel, Apple, Nuance, BBN, …)
- Potential for more breakthroughs and approaching the "understanding" part of AI by
  - Scaling computation
  - Numerical optimization (better training much deeper nets, RNNs)
  - Bypass intractable marginalizations and exploit broad priors and layer-wise training signals to learn more disentangled abstractions for unsupervised & structured output learning

LISA team: Merci! Questions?