Deep Architectures for Baby AI

Yoshua Bengio

August 7-11, 2007

Thanks to: Olivier Delalleau, Nicolas Le Roux, Hugo Larochelle, Pascal Lamblin, Dan Popovici, Aaron Courville, Clarence Simard, Jerome Louradour, Dumitru Erhan

Summary for Day 1

- Motivation: **understanding intelligence**, building AI, scaling to large scale learning of complex functions
- Neural networks and distributed representations
- Convolutional neural nets for vision and temporal data
- The Curse of Dimensionality: what's wrong with local non-parametric estimators (trees, local kernel machines)
- What's wrong with shallow architectures
- Optimizing deep architectures

Grand Goals: Understanding Intelligence, Building AI

- Ambitious goal: using statistical learning to understand intelligence, reach AI
- *Tasks involving intelligence*: visual and auditory perception, language understanding, intelligent control, long-term prediction, discovering and understanding of high-level abstractions...
- **Remains elusive!** (did ML community turn its back on it?)
- 3 considerations, to scale to learning complex tasks:
  - computational efficiency
  - statistical efficiency (nb examples needed)
  - human-labor/evolution effort (complexity of prior)

(Bengio and Le Cun, 2007)

Learning from Complicated Densities

These tasks involve

- Complicated structures: density is a **highly-varying function**
- Data is very high-dimensional (e.g., 200x200 video is 40000 pixels / image = 1.2 million inputs / sec. @ 30 frames/sec.)
- Density possibly concentrates on manifold, but of non-trivial dimensionality (certainly much more than 10 or 50)
- Abstractions MUST be LEARNT, can't spoon-feed all of them
- Many levels of abstractions
- LOTS of examples

Statistical Learning in 2007 $\longrightarrow$ AI?

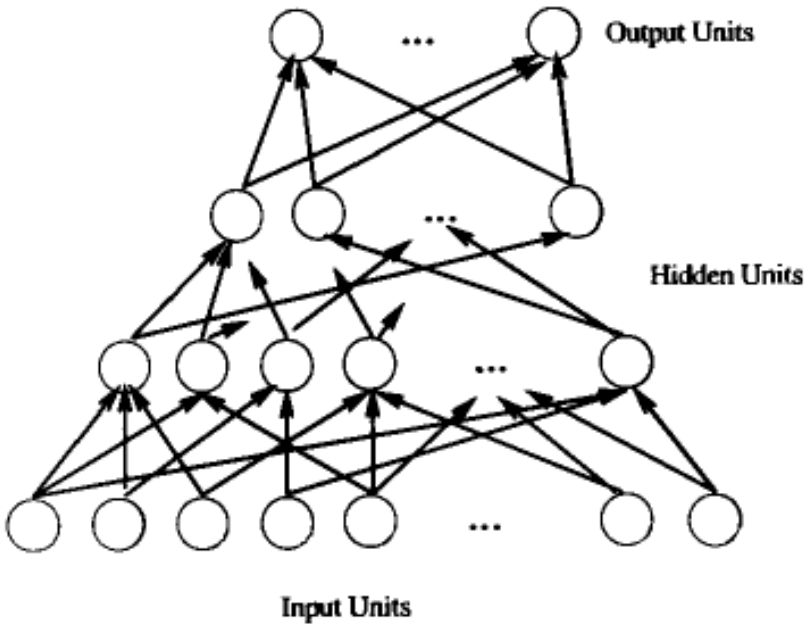
- Lots of progress in learning theory helped us understand concepts of generalization, capacity, overfitting, non-parametric estimation of functions.
- But most *commonly used learning algorithms are severely limited*:
  - Sensitive to the curse of dimensionality / poor statistical scaling with the apparent complexity of the function to be learnt.
  - Training algorithms computational requirements scaling badly with number of training examples (e.g. $O(n^2)$ time and memory).
  - **No algorithm able to discover high-level abstractions**, yet.

Context and Abstractions

- Consider a learning agent operating in the real world
- Context = summary of past experience + current input
- Abstraction
  - = complicated non-linear function of the context that helps to model it or make predictions about future inputs
  - discrete abstraction (= class) corresponds to a large set of possible inputs/contexts
  - higher-level abstractions are defined in terms of lower-level ones
  - higher-level abstractions = functions with many twists and turns, apparently complex, difficult to hand-program
- The number of abstractions that humans master is very large (verbalizable and non-verbalizable)
- *No hope to pre-program / hardwire all: must learn them.*
- **No hope to hand-label data illustrating well enough all of them: need unsupervised or semi-supervised learning.**

Neural Networks

- Means different things to different people.
- Computation performed by the composition of simple neuron-like units: non-linear transformation of linear combination of inputs from other neurons.



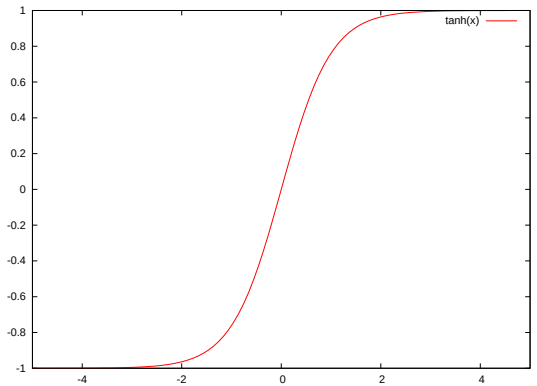
Neural Networks

- Most common: multi-layer neural networks (MLPs), in particular the 2-layer network

$$f_\theta(x) = b + W \tanh(c + Vx)$$

where $x \in \mathbb{R}^d$, $f(x), b \in \mathbb{R}^m$, $W \in \mathbb{R}^{m \times h}$, $c \in \mathbb{R}^h$, $V \in \mathbb{R}^{h \times m}$. Call h=**nb hidden units**, and parameters $\theta = (b, W, c, V)$.

- Can replace $\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$ by $\text{sigmoid(a)} = \frac{1}{1+e^{-a}} = (\tanh(a) + 1)/2$.



Gradient-Based Optimization of a Loss

For each example x_t consider that $f(x_t)$ incurs a loss C_t. We would like to minimize the **expected loss** $E[C_t]$. The **empirical loss** on a *training set* is

$$C = \frac{1}{T} \sum_{t=1}^{T} C_t.$$

The **gradient** $\frac{\partial C}{\partial \theta_i}$ tells us whether we should increase θ_i or decrease it. If can't solve $\frac{\partial C}{\partial \theta} = 0$, minimize **iteratively**.

Batch gradient descent: $\theta \leftarrow \theta - \epsilon \frac{\partial C}{\partial \theta}$

Stochastic or online gradient descent: $\theta \leftarrow \theta - \epsilon \frac{\partial C_t}{\partial \theta}$
SCALES WELL WHEN $T \rightarrow \infty$. Asymptotic convergence is guaranteed only if ϵ is reduced roughly in $1/t$.

Other numerical optimization techniques have been used, e.g. Conjugate Gradients, and convex optimization methods when C is convex in θ (only true when optimizing only last layer of MLP).

Efficient $f(x) \Rightarrow$ Efficient $\frac{\partial f(x)}{\partial x}$

If one can compute $f(x)$ efficiently in $O(N)$ operations, one can also compute $\frac{\partial f(x)}{\partial x}$ in $O(N)$ operations, for any f.

How?

- Expand the computation of $f(x)$ into a flow graph of $O(N)$ elementary computations, with nodes u
- Node u computes a function of its parent nodes and sends its value to children nodes. Nodes are properly ordered to obtain $f(x)$ from x.
- We can compute $\frac{\partial f(x)}{\partial u}$ for every u recursively, starting at $u = f(x)$ and ending at $u = x$.
- Use the chain rule: $\frac{\partial f(x)}{\partial u} = \sum_{v \in children(u)} \frac{\partial f(x)}{\partial v} \frac{\partial v}{\partial u}$ in the inverse order used to compute $f(x)$.

Supervised, Semi-Supervised, Unsupervised

1. **Unsupervised Learning**: the goal is to capture salient features of true unknown distribution of x_t's, or estimate the density $p(x)$. If $f(x)$ estimates the density and integrates to 1, we can choose *negative log-likelihood* (NLL) $C_t = -\log f(x_t) = KL(p||f) + H(p)$. It may be difficult to satisfy the sum to 1 constraint.
2. **Supervised Learning**: C_t compares $f(x_t)$ with some observed target y_t, e.g., $C_t = (f(x_t) - y_t)^2$ for regression $C_t = -\log softmax(f(x_t))_{y_t}$ = conditional NLL for probabilistic classification, with $u = softmax(v) \Leftrightarrow u_i = \frac{e^{v_i}}{\sum_j e^{v_j}}$, and $softmax(f(x_t))_i$ estimates $P(Y_t = i|x_t)$. Often (x_t, y_t) illustrates an abstraction that we want the learning algorithm to capture.
3. **Semi-supervised Learning**: Many unlabeled inputs x_t along with a few (x_t, y_t) pairs available.

RBF Networks and Local Representations

If each neuron of our MLP computes

$$e^{-||x-w||^2/\sigma^2}$$

for input x, with parameters w and σ, we have a Radial Basis Function (RBF) network (Moody and Darken, 1989; Niranjan and Fallside, 1990).

This is a kind of **grandmother cell**: when x is close to the pattern w, the cell is activated strongly.

Small $\sigma \rightarrow 0$ few cells respond. softmax(activations) $\rightarrow$ normalized RBFs, the **nearest neighbors** of x always respond. $\sigma \rightarrow 0$: *one-hot encoding* of the input (the nearest neighbor responds).

Unsupervised learning can be used to set the w: e.g. pick a random subset of examples x_t, or all of them, or a smartly selected subset of the most representative prototypes. This has given rise to Support Vector Machines (SVMs).

Local vs Distributed Representation

RBF net w/ small σ very inefficiently represents the input.

What about large σ? $f(x)$ too smooth.

As $\sigma \to \infty$, we can show that

1. $f(x) \to$ 2nd order polynomial in x, then
2. $f(x) \to$ affine function of x, then
3. $f(x) \to$ constant.

More units are on, but they mostly do the same thing!

Solution?

Let the hidden units be more "orthogonal" (e.g. as in PCA), describing different aspects of the input $\Rightarrow$ **distributed representations** (Hinton, 1986; Rumelhart, McClelland and the PDP Research Group, 1986; Bengio, Ducharme and Vincent, 2001)

Local vs Distributed Representation

To abstract the representation problem a bit, consider the hidden units are just binary and consider two extremes:

1. With h independent binary hidden units (distributed representation) one can represent 2^h different input patterns
2. With h maximally dependent binary hidden units (e.g., one-hot), one can represent only h different input patterns

$\Rightarrow$ **distributed representations can be exponentially more efficient than local ones.** (Bengio and Le Cun, 2007)

More evidence of this later.

Auto-Associators

Auto-Associator = unsupervised learning of internal distributed representation $h(x)$, a.k.a. *auto-encoder*
(Rumelhart, McClelland and the PDP Research Group, 1986; DeMers and Cottrell, 1993; Hinton and Zemel, 1994).
Two functions in an AA:

1. $h(x)$ maps x to distributed representation (encoder)
2. $P(X = x|H = h)$ tries to reconstruct x from h (decoder)

If $P(X = x|H = h(x)) = 1$, we have perfect reconstruction.

Can be implemented with an MLP. Simplest AA, for binary inputs:

$$\begin{aligned} h(x) &= \text{sigmoid}(b + Wx) \\ P(X = x|H = h) &= \prod_i P(X_i = x_i|H = h) \\ P(X_i = 1|H = h) &= \text{sigmoid}(c_i + V_i' h) \end{aligned}$$

often find $V \approx W'$ so may force $V = W'$.

For Gaussian inputs, can take $P(X = x|H = h) \propto e^{-0.5||x-Vh||^2/\sigma^2}$

Convolutional Neural Networks

Convolutional neural networks exploit simple properties of images:

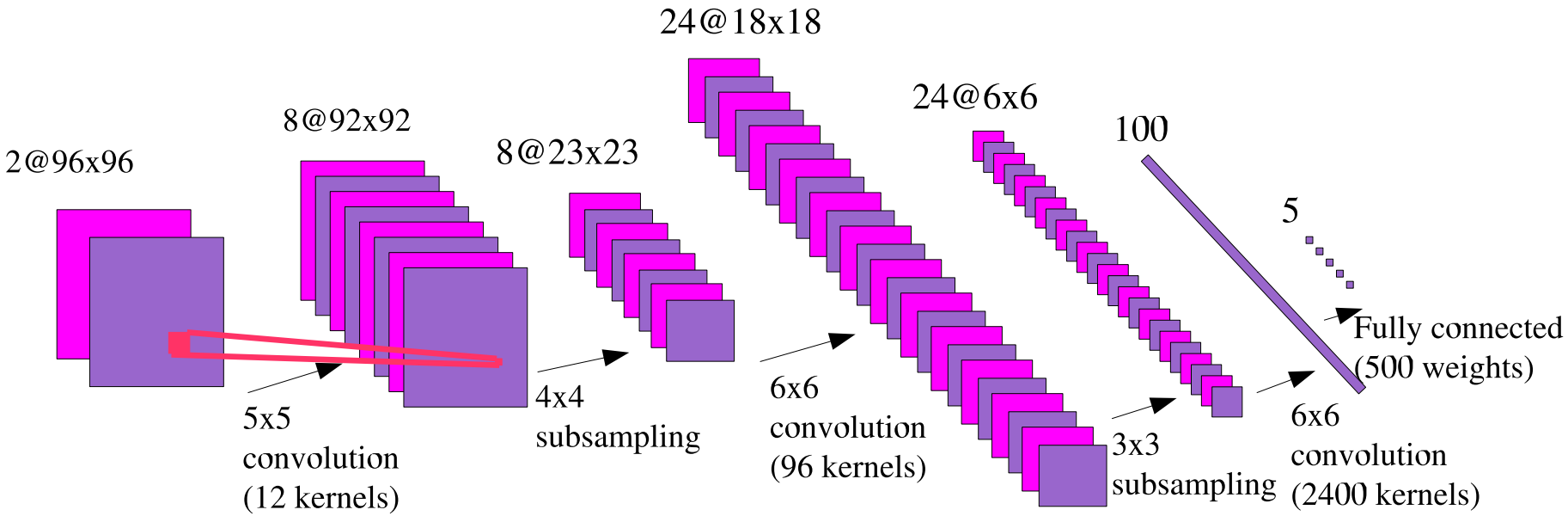
1. translation invariance
2. nearby pixels have stronger statistical dependency
3. hierarchy of local to global abstractions makes sense

This is achieved with three tricks (LeCun et al., 1989; LeCun and Bengio, 1995; LeCun et al., 1998b):

1. **Partial + local connectivity**: hidden neurons only connected to subset of spatially near neurons at previous layer.
2. **Weight sharing**: each neuron gets duplicated at different locations
3. **Pooling / subsampling**: the max or average of nearby neurons is propagated upward, making the network very robust to small distortions of input image

Convolutional Neural Networks

Example of supervised convolutional network successfully trained on handwritten digit image classification tasks:



Neurons at the same layer with the same weights compute a convolution of the image at the previous layer, followed by a non-linearity: result = another image.

Different convolution kernels = different image planes.

Convolutional Neural Networks

Convolutional layer with $K \times K$ convolutions:

$$y_k(i,j) = \tanh(b_k + \sum_{s=1}^{K} \sum_{t=1}^{K} \sum_{l} w_{k,l,s,t} \; x_l(i-s, j-t))$$

Subsampling layer $S \times S$:

$$y_k(i,j) = \tanh(b_k + c_k \sum_{i'=S\times i}^{S\times(i+1)-1} \sum_{j'=S\times j}^{S\times(j+1)-1} x_k(i',j'))$$

Implementation: must pay attention to matching image sizes.

Time-Delay Neural Networks (TDNN): same principle, 1-dimension (time). (Lang and Hinton, 1988)

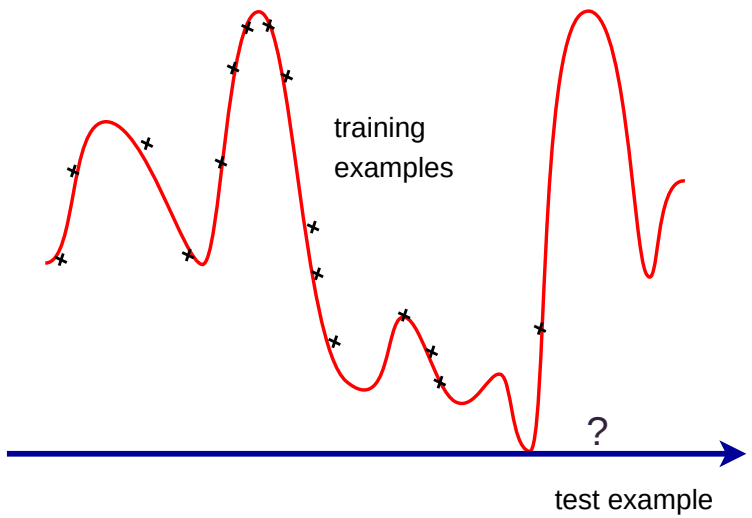
Convolutional Neural Networks

Summary of comparative results (LeCun et al., 1998a):

CLASSIFIER	DEFORMATION	ERROR	Reference
Knowledge-free methods			
2-layer NN, 800 HU, CE		1.60	Simard et al. 2003
3-layer NN, 500+300 HU, CE, reg		1.53	Hinton et al. 2006
SVM, Gaussian Kernel		1.40	Cortes et al 1992
Unsupervised Stacked RBM + backprop		0.95	Hinton et al. 2006
Convolutional nets			
Convolutional net LeNet-5,		0.80	
Convolutional net LeNet-6,		0.70	
Conv. net LeNet-6- + unsup learning		0.60	
Training set augmented with Affine Distortions			
2-layer NN, 800 HU, CE	Affine	1.10	Simard et al. 2003
Virtual SVM deg-9 poly	Affine	0.80	Scholkopf
Convolutional net, CE	Affine	0.60	Simard et al., 2003
Training et augmented with Elastic Distortions			
2-layer NN, 800 HU, CE	Elastic	0.70	Simard et al., 2003
Convolutional net, CE	Elastic	0.40	Simard et al., 2003
Conv. net LeNet-6- + unsup learning	Elastic	0.38	Ranzato et al. 2006

Local vs Non-Local Generalization

- Local generalization: core of most non-parametric ML
- To infer f at x, look for training examples x_i near x
- Simple example: nearest neighbor classifier
- Fails when target function varies a lot compared to how much the training examples "fill the space"



Variations are not arbitrary: they are caused by the structure of the world and the laws of physics.

(Bengio and Monperrus, 2005;
Bengio, Delalleau and Le Roux, 2006)

Smoothness Prior is Not Enough

- Assume target f more likely to be smooth, i.e., if $x \approx y$ then $f(x) \approx f(y)$.
- Useful but unsufficient prior!
- unfit to learn functions that vary a lot, e.g. needed for complex AI tasks, e.g. natural language, machine translation, general-purpose vision, robotics...
- Problem with SVMs with local kernel, manifold learning algorithms (LLE, ISOMAP, kernel PCA), graph-based semi-supervised learning, decision trees...

Local Learning Algorithms

A learned parameter of the model influences the value of the learned function in a local area of the input domain.

With local kernel machine

$$f(x) = \sum_i \alpha_i K(x, x_i),$$

α_i only influences $f(x)$ for x near x_i.

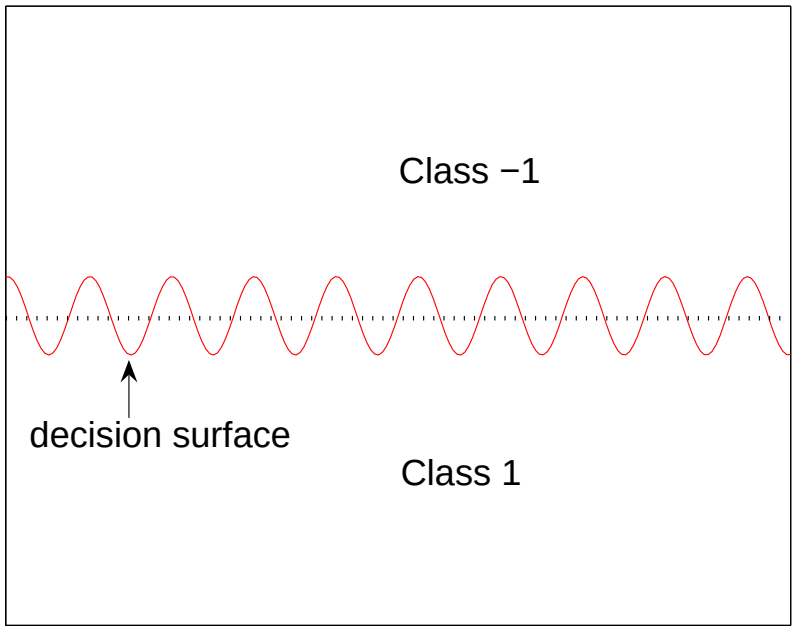
Examples:

- nearest-neighbor algorithms
- local kernel machines
- most non-parametric models except multi-layer neural networks and decision trees and boosted trees

Mathematical Problem with Local Learning

Theorem

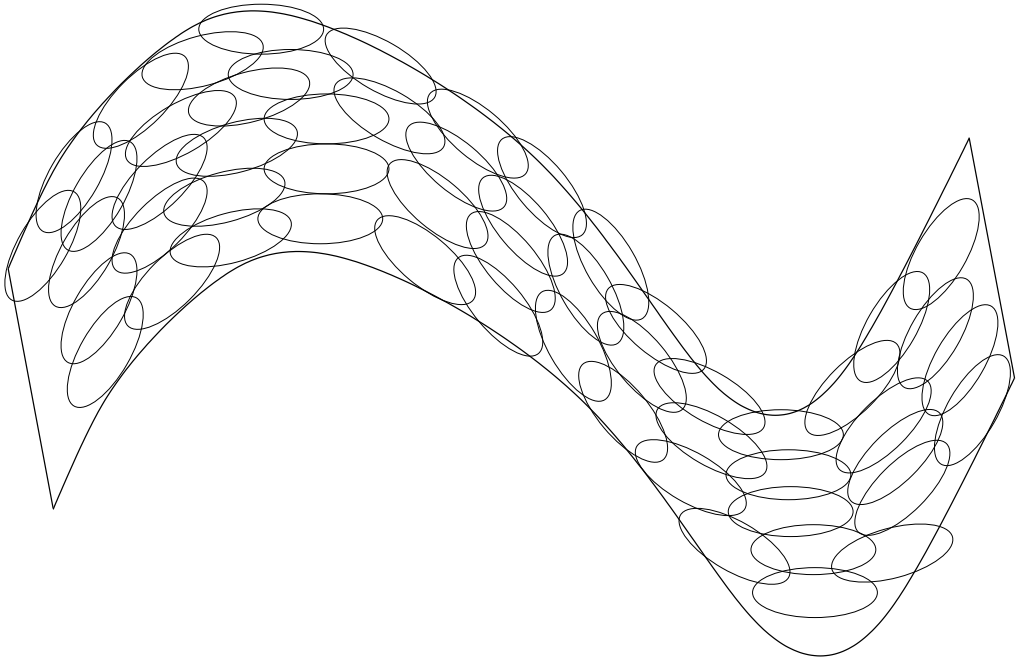
With K the Gaussian kernel and target $f(\cdot)$ changing sign at least $2k$ times along some straight line (i.e. that line crosses the decision surface at least $2k$ times), then at least k examples are required.

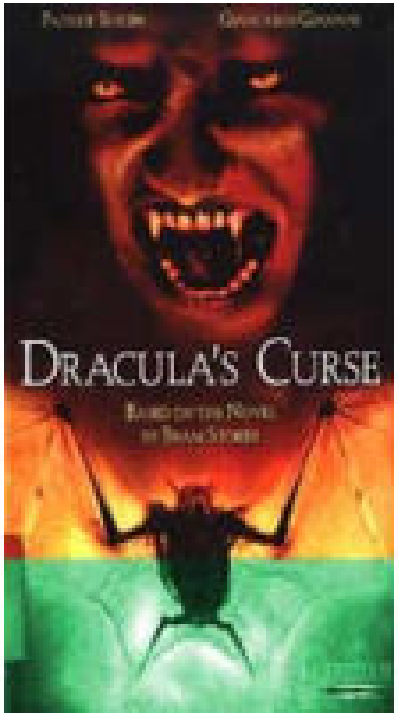


With local kernels, learning a function that has many "bumps" requires as many examples as bumps. (Bengio, Delalleau and Le Roux, 2006; Bengio and Le Cun, 2007)

The Curse of Dimensionality

Mathematical problem with classical non-parametric models





May need to have examples for each probable combination of the variables of interest.

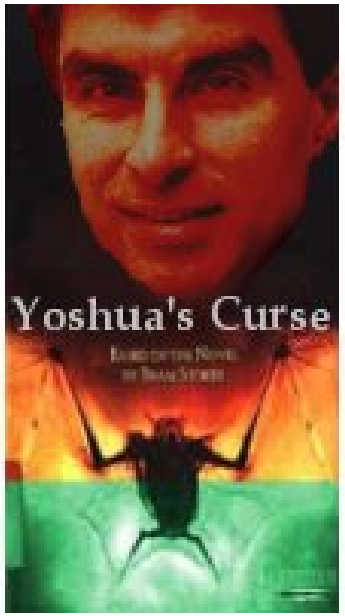
OK for 2 or 3 variables,

$\Rightarrow$ NOT OK for more abstract concepts...

Mathematical Problem with Local Kernels

Theorem

With K the Gaussian kernel, and the goal to learn a maximally changing binary function ($f(x) \neq f(x')$ when $|x - x'| = 1$) with d inputs, then at least 2^{d-1} examples are required.



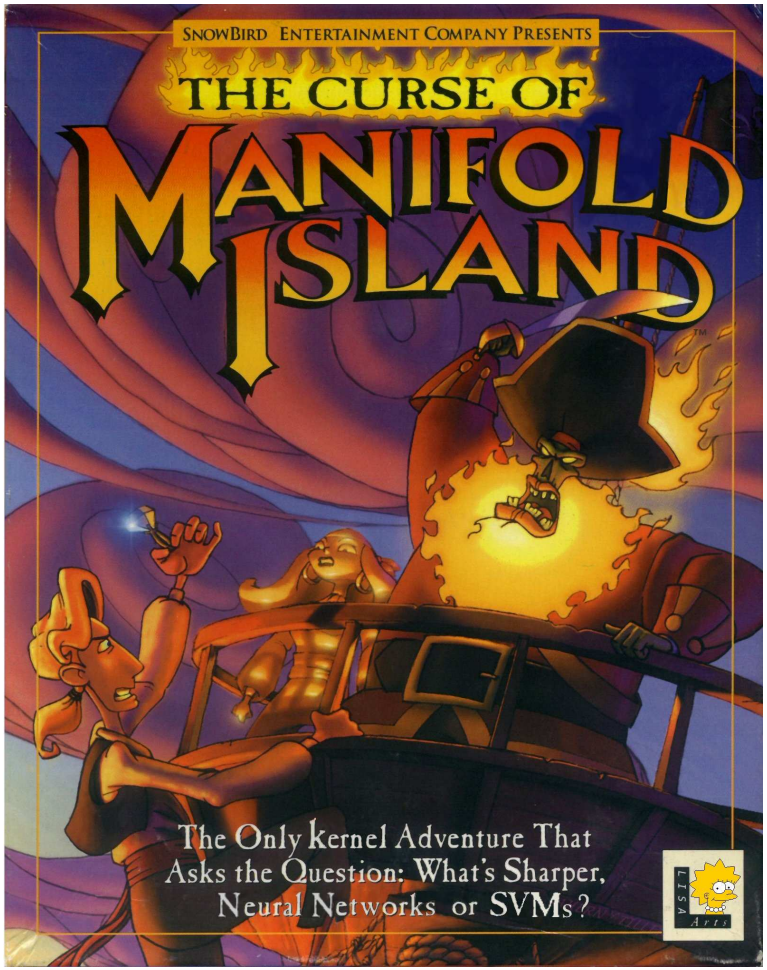
$\Rightarrow$ need to cover the space of possibilities with examples

$\Rightarrow$ may require nb examples exponential in nb inputs

$=$ *strongly negative mathematical results on local kernel machines*

Other similar results in (Bengio, Delalleau and Le Roux, 2006).

Curse of Dimensionality for Manifold Learning

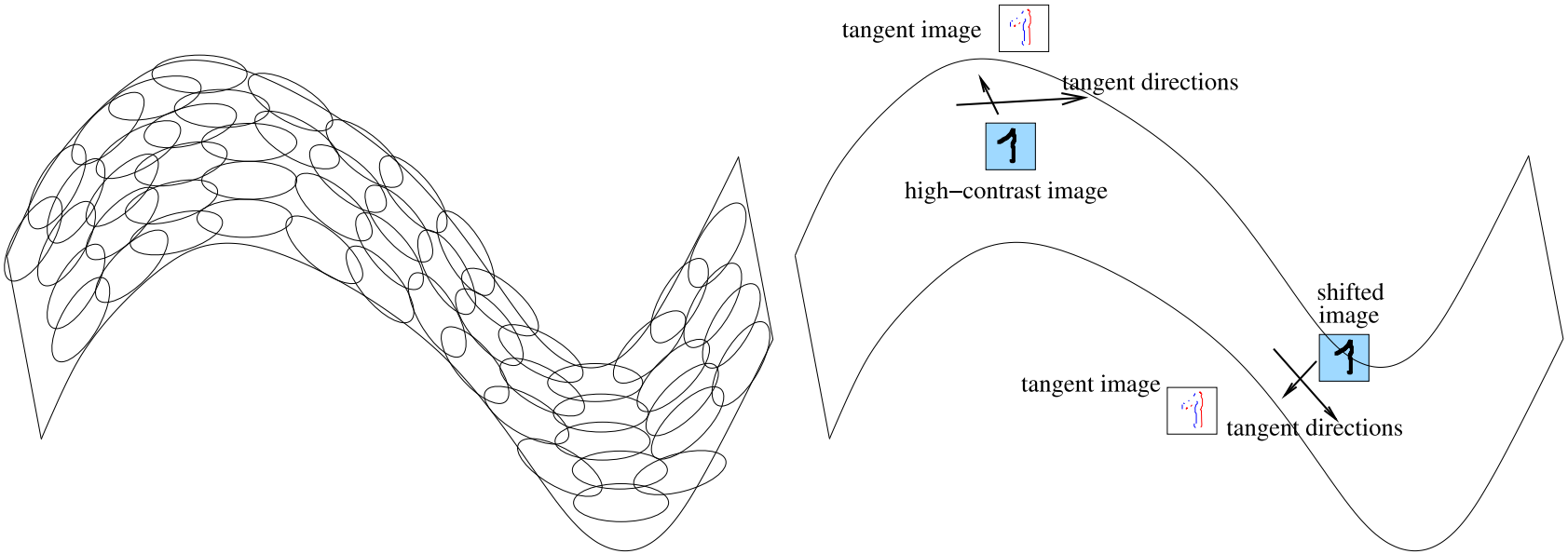


Many manifold learning algorithms are kernel methods with an "equivalent" local kernel:

- LLE (Saul and Roweis, 2002)
- Isomap (Tenenbaum, de Silva & Langford 2000)
- kernel PCA (Schölkopf, Smola and Müller, 1996)
- dim. red. in spectral clustering (Weiss, 1999)
- Laplacian Eigenmaps (Belkin and Niyogi, 2002),
- etc.

However, the kernel K_D is *data-dependent*. (Bengio et al., 2004)

Ex: Translation of a High Contrast Image

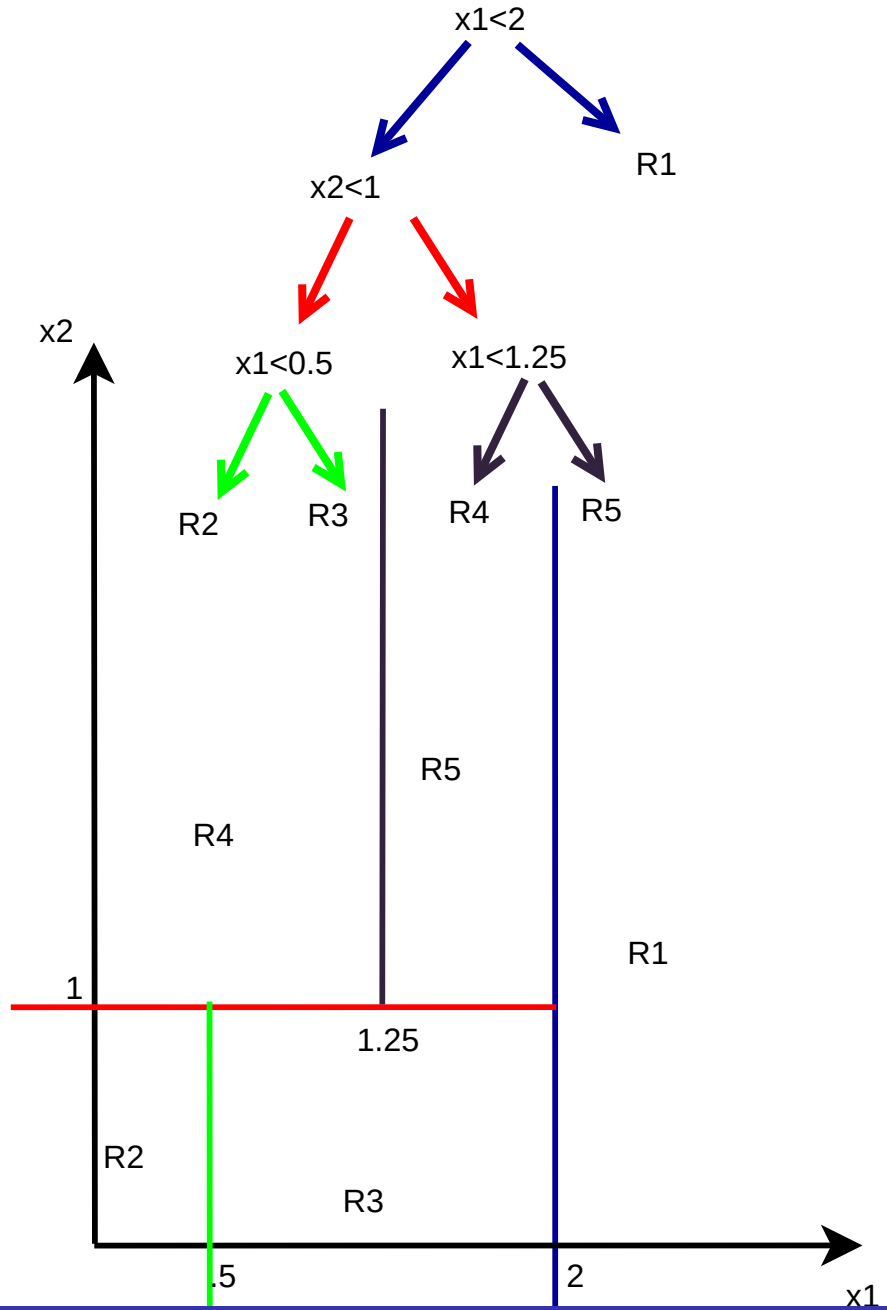


∃ examples of **non-local learning** with **no domain-specific prior knowledge**, successfully learning such manifolds (rotations and translations), (Bengio and Monperrus, 2005; Bengio, Larochelle and Vincent, 2006), generalizing far from training examples.

Decision Trees

(Breiman et al., 1984)

- Non-parametric
- Non-convex optimization but greedy heuristics work well
- Recursively partitions the space, usually axis-wise; puts constant prediction on leaves (rect. regions).



Negative Results on Decision Trees

Decision Trees do not Generalize to New Variations!

Proposition

If N is the number of pieces required to approach a target function with a piece-wise constant approximation with error ϵ, then a constant-leaves decision tree requires at least N examples to obtain generalization error ϵ.

Target fn may have a huge nb of variations, but these may be structured = "simple" expression, yet not tree-learnable.

Corollary

Nb exemples required to train a constant-leaves decision tree can grow exponentially with d (input dim.)

(Bengio, Delalleau and Simard, 2007)

Negative Results on Decision Trees

Lemma

On d-bits parity task, constant-leaves decision tree with axis-aligned decision nodes will have a generalization error of $\frac{1}{2}$ on leaf nodes of depth less than d.

Theorem

On d-bits parity task, constant-leaves decision tree with axis-aligned decision nodes trained with n examples will have generalization error in $[\frac{1}{2} - \frac{n}{2^{d+1}}, 1 - \frac{n}{2^{d+1}}]$.

Corollary

On d-bits parity task, constant-leaves decision tree with axis-aligned decision nodes requires at least $2^d(1 - 2\epsilon)$ examples to reach gen. error $\leq \epsilon$.

Another Justification for Boosting Trees

- Boosted classification trees (Freund and Schapire, 1996):
  $f(x) = \text{sign}(\sum_i \alpha_i T_i(x))$ $\quad$ ($T_i(x)$ = prediction of i-th tree)
- Often generalize better than single trees
- Many explanations have already been proposed.
- Another: above theorems do not apply to boosted trees.

Proposition

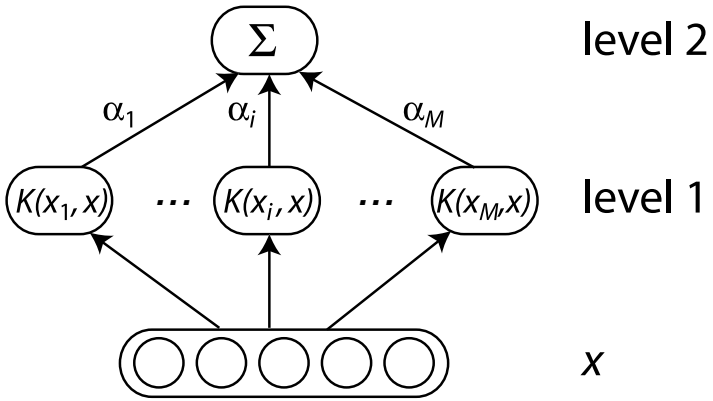
A sum of n trees can represent a nb of piece-wise constant regions exponential in n, whereas a single tree can only represent a number of regions = nb of leaves.

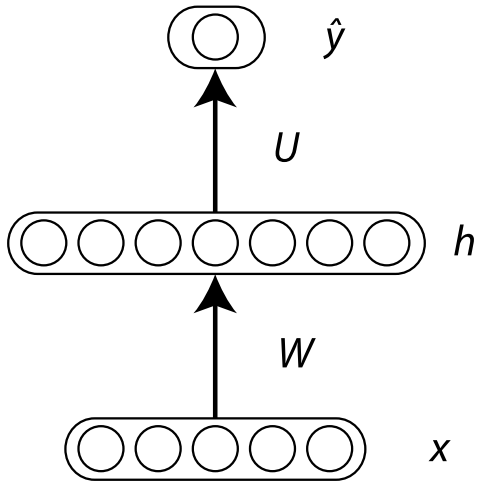
But boosted trees really have depth of only 3...

Depth of Architectures

(Bengio and Le Cun, 2007)
Depth = number of **levels** of composition of adaptable elements:

- kernel machines: shallow
- boosting: generally shallow
- multi-layer neural networks: usually shallow, can be deep?
- decision trees: shallow
- parametric graphical models: human-labor intensive



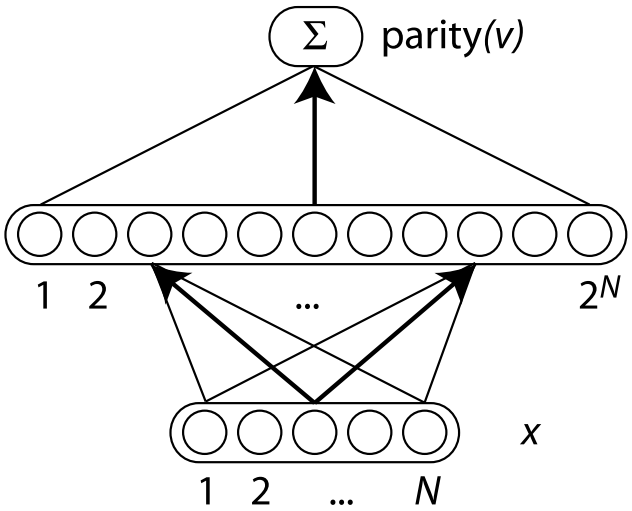


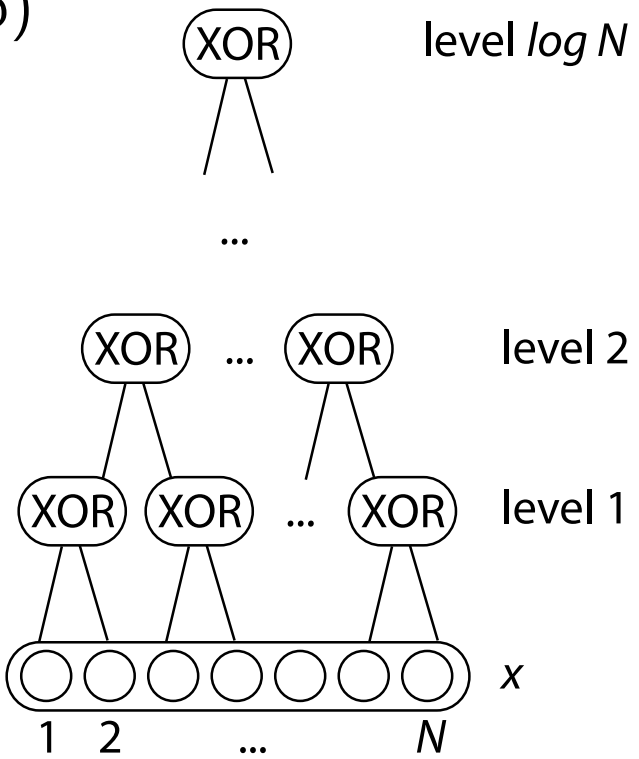
Non-parametric ones can theoretically approximate any continuous function.
But **how efficiently?**
(computational, statistical)

Inefficiency of Shallow Architectures

Mathematical results from complexity theory of boolean circuits:

Functions representable compactly by a deep circuit often need circuits of exponential size to be represented by a shallow circuit (Hastad, 1987; Allender, 1996)





Very fat shallow circuit

$\Rightarrow$ many adjustable elements

$\Rightarrow$ many examples needed

Brain has a deep architecture

Number of levels should not be fixed but data-dependent. See also (Utgoff and Stracuzzi, 2002; Bengio and Le Cun, 2007).

Optimizing Deep Architectures

- Until 2006, we knew no way to train a deep neural net to obtain better results than a shallow one (1 or 2 hidden layer), except for convolutional neural nets (Tesauro, 1992).
- Training seemed to get stuck in sub-optimal solutions, local minima or plateaus or just a too convoluted error surface.
- We still do not know why deeper is ok for convolutional nets.
- We still do not fully understand why it is so difficult to optimize deep architectures by gradient-based techniques.
- But DEPTH seems a necessary condition for statistical efficiency!

What happened in 2006?

Geoff Hinton, Simon Osindero and Yee-Wye Teh published a Neural Computation paper on "A fast learning algorithm for deep belief nets" (2006), that introduces these ideas:

- A deep unsupervised network could be trained greedily, layer by layer.
- Each layer tries to model its inputs.
- Each layer outputs a representation of its input.
- This unsupervised net is a good initialization for a supervised net.

Presumably easier to learn "locally" (within each layer) than having to coordinate all the layers in a deep network.

Summary for Day 2

- Greedy learning of *multiple levels of abstractions*
- Partially supervised training of DBNs
- Dimensionality reduction vs Entropy minimization
- Baby AI School: curriculum for training AIs
- RBM are **universal approximators**
- Global training of DBNs
- *Space-time hierarchy*
- Culture + language: humanity as optimization device

Learning High-Level Abstractions

High-level abstraction =

a highly-varying, complex, but structured **function**

e.g. semantic concept of "chair" is a higher-level abstraction than a recognizer of a particular instance of chair, which is a higher-level abstraction than an oriented edge-detector.

Such high-level abstraction includes a *very large set of possible inputs that can be very different from each other* in terms of raw sensory patterns (e.g. "Euclidean distance" in retinal image).

Greedy Learning of Multiple Levels of Abstractions

- Learning AI $\Rightarrow$ **learning abstractions**
- General principle: **Greedily learning simple things first, higher-level abstractions on top of lower-level ones.**
- **Implicit prior** (Bengio et al., 2007): restrict to functions that
  1. can be represented as a composition of simpler ones such that
  2. the simpler ones can be learned first (i.e., are also good models of the data).

  Coherent with psychological literature (Piaget, 1952).
  We learn baby math before arithmetic before algebra before differential equations . . .
- Also some evidence from neurobiology: (Guillery, 2005) *"Is postnatal neocortical maturation hierarchical?"*.

Experiments on Greedy Layer-Wise Initialization

(Bengio et al., 2007)

	train.	test.
Deep Belief Net, unsupervised pre-training	0%	**1.2%**
Deep net, auto-associator pre-training	0%	**1.4%**
Deep net, supervised pre-training	0%	2.0%
Deep net, no pre-training	.004%	2.4%
Shallow net, no pre-training	.004%	1.9%

Classification error on MNIST digits benchmark training, validation, and test sets, with the best hyper-parameters according to validation error.

Deep nets with 3 to 5 hidden layers.
Selects around 500 hidden units per layer.

Supervised greedy is **too greedy**.
Greedy unsupervised initialization works great.

Is it Really an Optimization Problem?

Why 0 train error even with deep net / no-pretraining?

Classification error on MNIST with 20 hidden units on top layer:

	train.	test.
Deep Belief Net, unsupervised pre-training	.008%	**1.5%**
Deep net, autoassociator pre-training	0%	**1.6%**
Deep net, supervised pre-training	0%	1.9%
Deep net, no pre-training	.59%	2.2%
Shallow net, no pre-training	3.6%	5.0%

Because

1. last fat hidden layer did all the work
2. using a poor representation (output of all previous layers)

Yes it is really an **optimization** problem
and a **representation** problem

Extending RBMs to Continuous Inputs

Easy to *change the energy function and range of values* to obtain different models. (Bengio et al., 2007)

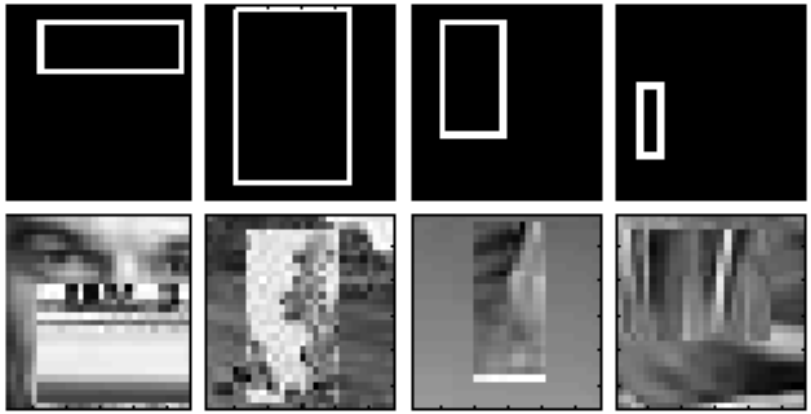
- Continuous range of values: get **exponential** $(0, \infty)$ or truncated exponential $(0, 1)$ density $\propto 1_{u \in (0,1)} e^{-ua}$.
  Consider unit u_i, other layer $\mathbf{z}$, with energy term $-u_i a(\mathbf{z})$.
- Add diagonal quadratic term $-d_i^2 u_i^2$: Gaussian units (Welling, Rosen-Zvi and Hinton, 2005).

Warning: Gaussian and exponential units have linear expectation.

Used truncated exponential density: sigmoidal non-linearity.

$$E[y|\mathbf{z}] = \frac{1}{1-exp(a(\mathbf{z}))} + \frac{1}{a(\mathbf{z})}.$$

Deep Networks vs SVMs

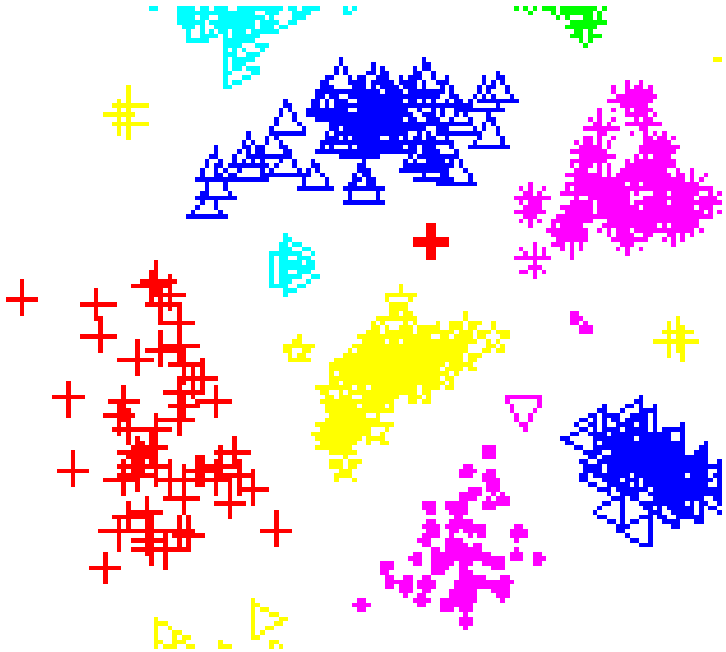


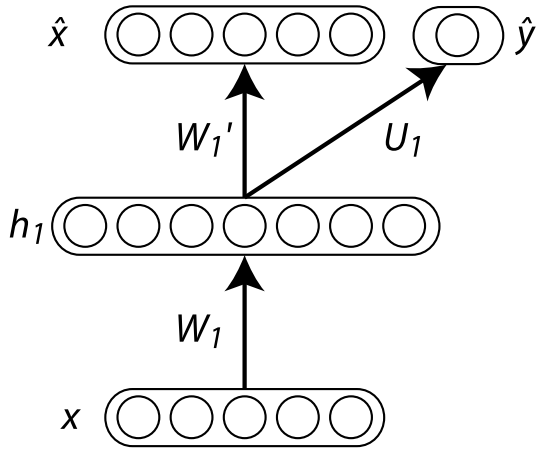
From top to bottom, samples from *rectangles* and *rectangles-image* (Larochelle et al., 2007)

Gradually more complex tasks:

Dataset	**SVM$_{rbf}$**	**SVM$_{poly}$**	**NNet**	**DBN-1**	**SAA-3**	**DBN-3**
mnist-basic	**2.2%**	2.9%	3.2%	3.0%	2.45%	2.4 %
mnist-rot	**10.5%**	12.7%	16.5%	12.7%	11.6%	12.9%
mnist-back-rand	14.1%	16.5%	19.4%	9.2%	11.4%	**5.9%**
mnist-back-img	22.5%	23.6%	29.2%	**16.4%**	23.3%	17.5%
mnist-rot-back-img	31.0%	36.0%	40.8%	31.4%	**23.1%**	26.9%
rectangles	2.43%	3.2%	7.9%	6.0%	**2.1%**	2.6%
rectangles-img	23.5%	24.2%	31.1%	23.4%	23.9%	**22.0%**
convex	19.1%	19.0%	27.7%	20.3%	18.9%	**17.6%**

Combining Supervised & Unsupervised





MNIST: nice clusters in the distribution

$\Rightarrow$ **input distribution structure reveals the target class.**

$f_1(x) = P(Y|x)$ related to $f_2(x) = P(x)$. Otherwise? combine supervised & unsupervised layer-wise greedy initialization. Add the two stochastic gradient updates (Bengio et al., 2007)

Similar to Partial Least Squares

EASY TO DO SEMI-SUPERVISED LEARNING WITH DBNs

More experimental results

(Bengio et al., 2007)

	Abalone MSE	**Cotton** class. error
DBN, Gauss inputs, partially sup	**4.18**	**31.4%**
DBN, Gauss inputs, unsup	4.19	35.8%
DBN, Bin inputs, partially sup	4.28	43.7%
DBN, Bin inputs, unsup	4.47	45.0%
Logistic regression	·	45.0%
Deep Network, no pre-training	4.2	43.0%

MSE on Abalone task and classification error on Cotton task, showing improvement with Gaussian vs binomial units and partial supervision

Representational Ability of RBMs and DBNs

We already know that MLPs are universal approximators of functions, with enough hidden units. What about RBMs and DBNs for distributions?

RBMs become non-parametric distribution estimators when we allow the number of hidden units to vary.

Questions:

- Can a single RBM model any distribution, with enough hidden units?
- How much additional modeling power can a DBN bring by stacking more RBMs on top of a fixed-size RBM?
- What about an infinite number of layers?

RBMs are Universal Approximators

With enough hidden units any distribution can be represented exactly: (paper to appear by Le Roux & Bengio)

Theorem

Any distribution over $\{0,1\}^n$ can be approximated arbitrary well with a RBM with $k+1$ hidden units where k is the number of input vectors whose probability is not 0.

Adding one hidden unit (with proper parameters) increases log-likelihood:

Theorem

Let u be an arbitrary distribution over $\{0,1\}^n$ and let P be a RBM with marginal distribution p over the visible units such that $KL(u||p) > 0$. Then there exists a RBM Q composed of P and an additional hidden unit, with marginal distribution q over the visible units such that $KL(u||q) < KL(u||p)$.

OOS NLL = E[Coding Cost]

Expected coding cost of examples = out-of-sample negative log-likelihood

Coding cost = $E_p[-\log q(x)]$

where p = real distribution, q used to encode and select $-\log q(x)$ bits for pattern x.

Shannon: coding cost minimized when $q = p$, equal entropy.

For a fixed data set, one must transmit both the data codes and the model with all its parameters (learned from that data). (Zemel, 1993; Hinton and Zemel, 1994)

But one can also transmit just the learning algorithm (very small!) and then process the data online (update the model after each example), transmitting **only the online codes**:

Coding cost of $x_t = -\log P(x_t|\theta(x_1, \ldots x_{t-1}))$

Using Jensen

$$\log p(x) = \sum_h p(h|x) \log p(x) = \sum_h p(h|x) \log p(x|h)p(h) + H_{p(h|x)}$$

$$\begin{aligned} \log p(x) &= \log \sum_h p(x|h)p(h) = \log \sum_h q(h|x) \frac{p(x|h)p(h)}{q(h|x)} \\ &\geq \sum_h q(h|x) \log \frac{p(x|h)p(h)}{q(h|x)} = \sum_h q(h|x) \log p(x|h)p(h) + H_q \end{aligned}$$

which shows that the bound is tight when $q(h|x) = p(h|x)$ (bits back)

Optimizing the Variational Bound

Hinton's (2006) variational bound justifies improvement of adding a layer:

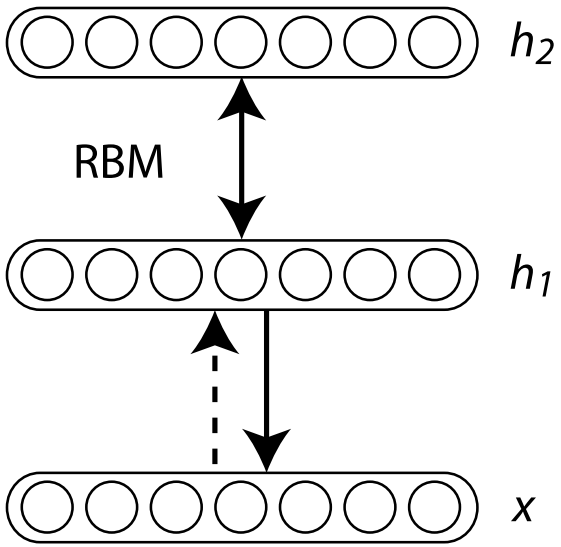
$$\log P(\mathbf{h}^0) \geq \sum_{\mathbf{h}^1} Q(\mathbf{h}^1|\mathbf{h}^0) \left[\log P(\mathbf{h}^1) + \log P(\mathbf{h}^0|\mathbf{h}^1)\right] - \sum_{\mathbf{h}^1} Q(\mathbf{h}^1|\mathbf{h}^0) \log Q(\mathbf{h}^1|\mathbf{h}^0)$$

Greedy strategy: freeze first layer $P(\mathbf{h}^0|\mathbf{h}^1)$ and optimize $P(\mathbf{h}^1)$.

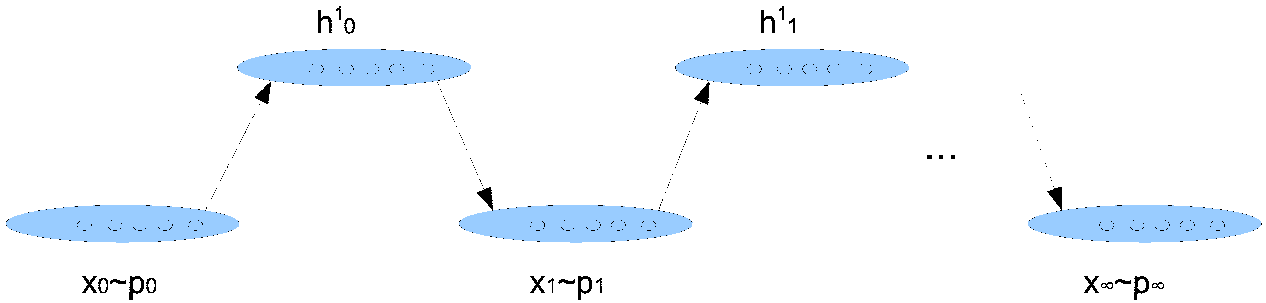
Proposition: Maximizing this variational bound non-parametrically gives us:

$$P^*(\mathbf{h}^1) = \sum_{\mathbf{h}_0} p^0(\mathbf{h}^0) Q(\mathbf{h}^1|\mathbf{h}^0)$$

where p^0 = input data. Using our first theorem, $\exists$ RBM achieving $P^*(\mathbf{h}^1)$.



Assuming Memory-Based Top Layer



p^k = distribution over x_k after k Gibbs up-down steps.
p^0 = data distribution, p^∞ = model distribution.

Proposition

In 2-layer DBN, using a 2nd layer RBM achieving $P^*(\mathbf{h}^1)$, $KL(\text{data} = p^0 || \text{model} = p^\infty) = KL(p^0||p^1)$.

The best we can hope when maximizing the bound, when adding a second RBM, is $KL(p^0||p^1)$ of the 1st RBM.

$\Rightarrow$ To get $KL(p^0||p^1) = 0$ need $p^0 = p^1$. Obtained when the first RBM perfectly fits the data, so that the second layer $\mathbf{h}^2$ seems useless. It can also be achieved when the first RBM has infinite weights (deterministic) and just codes x in h perfectly.

Memory-Based Top Layer: Discussion

Puzzle: the best that can be achieved by adding a second layer (wrt variational bound) only depends on the first layer's $KL(p^0||p^1)$, which decreases with increasing capacity of the first layer. This is generally an improvement over having only the first layer ($KL(p^0||p^\infty) \geq KL(p^0||p^1)$). But if the first layer becomes large, we do not need the second layer!

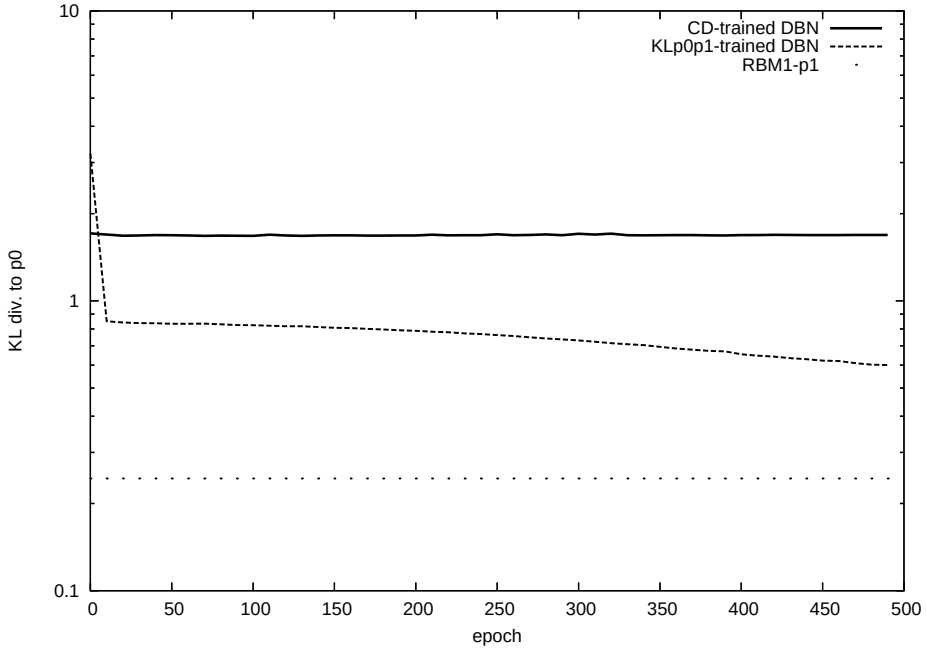
Does that mean that adding layers is useless? We believe not.

- We maximized the variational bound rather than likelihood.
- Even if adding layers does not allow us to perfectly fit the data, the distribution of our model is closer to the empirical distribution than a single lean RBM.
- Extra layers regularize and **hopefully learn better representations.**

This also suggests using **alternative criteria to train DBNs**, that approximate $KL(p^0||p^1)$, and which can be computed before $\mathbf{h}^2$ is added, but that unlike Contrastive Divergence, take into account the fact that more layers will be added later.

Minimizing $KL(p^0||p^1)$

Minimize
$KL(p^0||p^1) = -\log n - \frac{1}{n}\sum_{i=1}^{n} \log \frac{1}{n}\sum_{j=1}^{n}\sum_{h} P(x_i|h)P(h|x_j)$
on 1-D ball data.



KL divergence w.r.t. epochs, b/w empirical p_0 and (top) DBN trained greedily with CD, (middle) DBN trained greedily with $KL(p_0||p_1)$ on the 1st layer, and CD on 2nd, (bottom) distribution p_1 of 1st RBM, which should bound the other two curves.

Is there a good tractable approximation/gradient?

Global Unsupervised Optimization of Deep Architectures

1. Wake-sleep (Geoff will talk about it): train $P(x|h)$ from samples of $Q(h|x)p^0(x)$, and train $Q(h|x)$ from samples of $P(h,x)$.
2. Full auto-associator (Russ talked about it): minimize reconstruction error of the big auto-associator (unfolding at top layer).
3. All auto-associators: minimize reconstruction error of all the intermediate auto-associators reconstructing $h_i(x)$ using k layers above, for all values of k and i.
4. Single-Term Bound: Look for transformation $h(x)$ of the data that preserves all the interesting information in x while making distribution $h(x)$ easy to model (low entropy).
5. Variational Bound: newly discovered by Geoff...

Optimizing a Transformation

Consider a model with two blocks $P_2(h)$ and $P_1(x|h)$ (each could have layers). If h is discrete,

$$\log P(x) \geq B(x) = \log P_1(x|h = f(x))P_2(h = f(x))$$

There are 3 "parameters": P_1, P_2, and f.

Transform x into easier to model h without throwing away information

- P_2 optimized as in DBN, but we propagate gradient on $f(x)$
- P_1 optimized easily because trained with $(h = f(x), x)$ supervised pairs.
- Gradient on f is obtained by propagating gradients from $\log P_1(x|h = f(x))$ and $\log P_2(h = f(x))$.
- Magically, $\frac{\partial \log P_2(h=f(x))}{\partial f(x)} = \frac{\partial \log G_2(h=f(x))}{\partial f(x)}$ where $G_2(h) = -FE_2(h) = \log P_2(h) + \log Z_2$, computed easily.

Optimizing a Coding Cost

Optimizing a transformation = optimizing coding cost

We start with the coding cost of the raw data using some model P (say an RBM). Cost of $x_t = -logP_x(x_t|\theta(x_{t-1}, x_{t-2}, \ldots))$.

We look for a transformation $f(.)$ such that the coding cost = entropy is reduced.

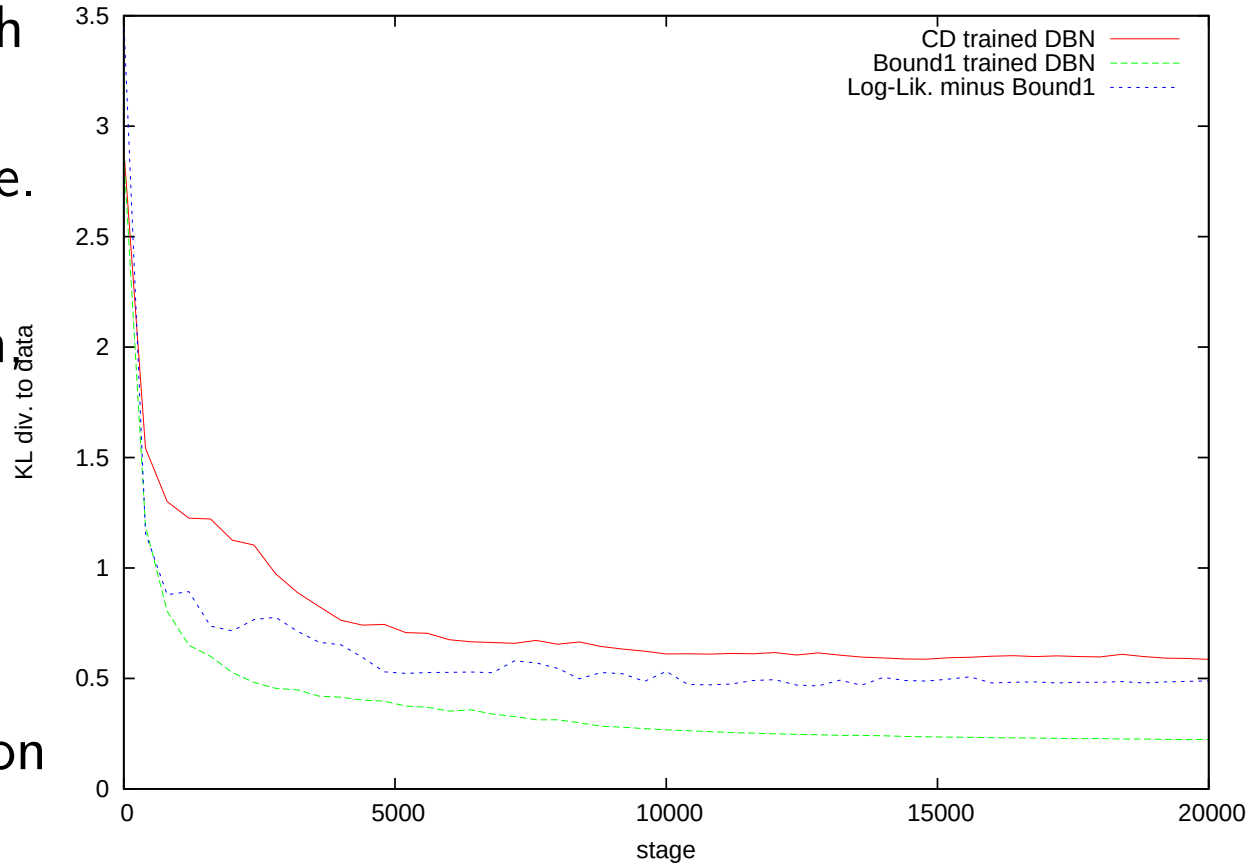
- Pay for the loss of bits from x to $f(x)$, via $\log P_{x|h}(x_t|f(x_t), \theta(x_{t-1}, x_{t-2}, \ldots))$.
- Pay for encoding $f(x_t)$: $-\log P_h(f(x_t)|\theta(x_{t-1}, x_{t-2}, \ldots))$.

Note 1: 2 cases, f is invertible or not.

Note 2: $f(x_t)$ can be pseudo-random without encurring a penalty, even if $f(x) \neq P(h|x)$.

Results of Single-Term Bound on Toy Task

- Toy task:
  1-dimensional ball on a 10-bit screen with size 1,2 or 3, and color black or white.
- 10 inputs, 5 penultimate hidden, 10 top hidden
- Can measure true likelihood exactly
- Compare greedy training (CD only on each layer) with optimizing the 1-term Bound.

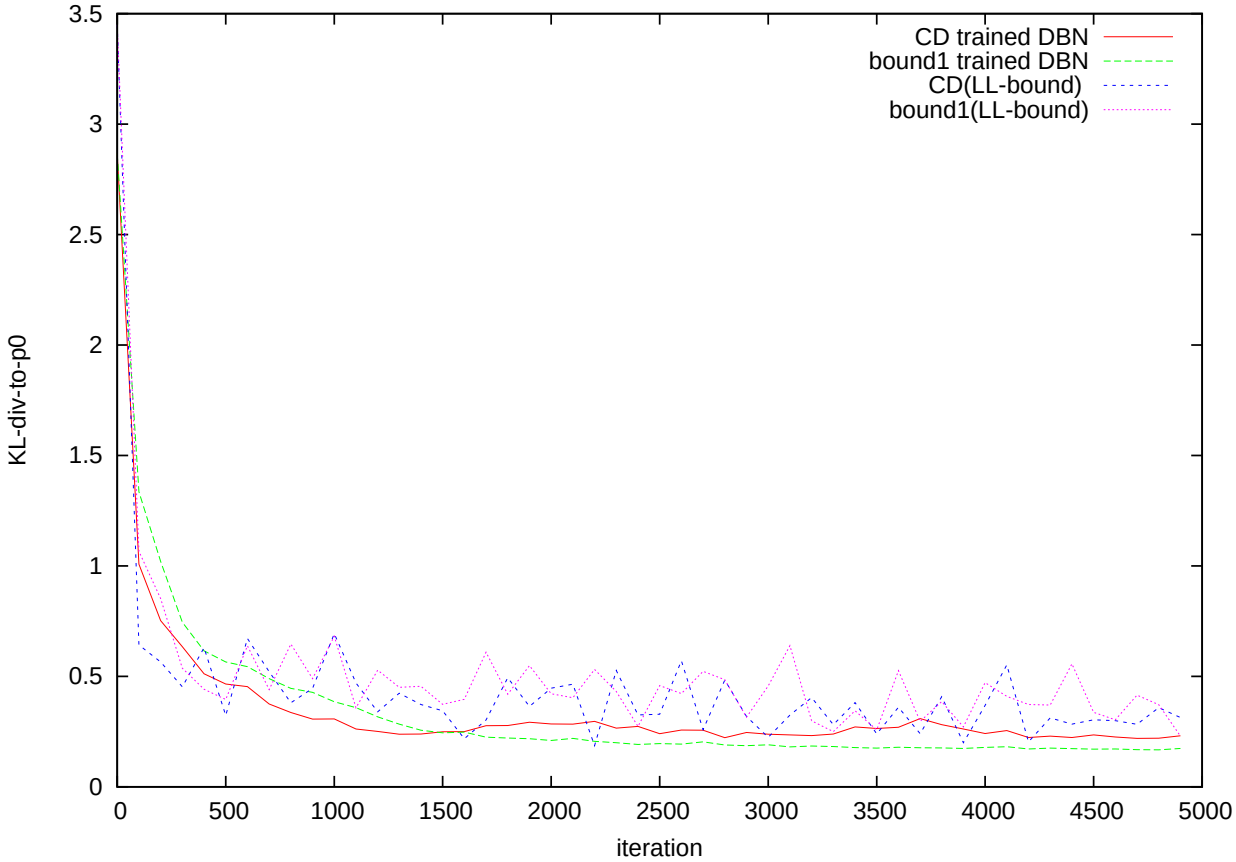


Actually, optimizing the learning rates schedules properly and separately on both systems, differences become smaller...

The Bound Becomes Tighter

As $B(x)$ is increased the bound becomes tighter!

This is true even when we only train with CD!



Unbiased Gradient Estimator of the Variational Bound

Geoff's new bound: maximizing variational bound globally!

$$\log P(x) \geq \mathcal{G}(x) = G_1(x) + \sum_{h_1,\ldots,h_L} Q(h|x)\Delta G(h) - \log Z_{L+1}$$

where $\Delta G(h) = \sum_{i=1}^{L} G_{i+1}(h_i) - G_i(h_i)$, with $G_{i+1}(h_i)$ = marginal goodness of input (h_i) of RBM $i+1$, $G_i(h_i)$ = marginal goodness of output (h_i) of RBM i, and $Q(h|x) = \prod_{i=1}^{L} P_i(h_i|h_{i-1})$.

Sketch of proof: Jensen + cancellation of $\log Z_i$'s, e.g. for 2 RBMs

$$\log P(x) \geq H_Q + \sum_h P_1(h|x)(\log P_2(h) + \log P_1(x|h))$$

and use $\log P_1(x) = H_Q + \sum_h P_1(h|x)(\log P_1(h) + \log P_1(x|h))$ to substitute $\sum_h P_1(h|x) \log P_1(x|h)$ above:

$$\log P(x) \geq H_Q + \sum_h P_1(h|x)(\log P_2(h) + \log P_1(x) - H_Q - \log P_1(h))$$

$$= \log P_1(x) + \sum_h P_1(h|x)(\log P_2(h) - \log P_1(h))$$

$$= G_1(x) + (\sum_h P_1(h|x)(G_2(h) - G_1(h))) - \log Z_2$$

Unbiased Gradient Estimator of the Variational Bound

$\Rightarrow$ can obtain unbiased gradient of $G(x)$ wrt lower RBMs parameters (top RBM can be trained by CD), sampling from $Q(h|x)$, using straightforward differentiation:

$$\frac{\partial \mathcal{G}(x)}{\partial W^l_{ij}} = E_Q[P_l(h_{li} = 1|h_{l-1})h_{l-1,j} - P_l(h_{l-1,j} = 1|h_l)h_{li} + (h_{li} - P_l(h_{li} = 1|h_{l-1}))h_{l-1,j}\Delta G(h)]$$

Note sampling from Q is straightforward (stochastic feedforward path).

Problem noted by Tim: information about good value of samples h for higher RBMs is *propagated through a single global scalar* $\Delta G(h)$.

Solution: Geoff's intuitive mean-field version...

Dream Data As Negative Examples

Examples generated by the model can serve as negative examples, and a **classifier** that helps separate positive (real) from negative (dream) examples can be used to improve the total free energy.

$$\begin{aligned} p_n(x) &= \frac{1}{Z_n} q_n(x) = \frac{1}{Z_n} e^{G_n(x)} = \frac{1}{Z_n} e^{\sum_{i=1}^{n} w_i g_i(x)} \\ \frac{\partial \log p(x)}{\partial w_n} &= 0 \end{aligned}$$

$\Rightarrow w_n = \sigma^{-1}(E_{data}[g_n(x)]) - \log E_{p_{n-1}}[g_n(x)] + \log E_{p_{n-1}}[1 - g_n(x)]$
i.e. only doing averages over real and fantasy data (generated before adding n-th term $w_n g_n(x)$) obtain w_n, given g_n.

Separating positive examples from the rest $\Rightarrow$ good likelihood.

Dream Data As Negative Examples

See also (Welling, Zemel and Hinton, 2003) (**self-supervised boosting**)

Goodness $G_n(x)$ is a classifier of fantasy vs real examples.

Taking $w_n \longrightarrow 0$, one can show that the optimal maximum likelihood g_n separates real examples from fantasy generated by previous model p_{n-1}.

$\Rightarrow$ can apply classifier technology (e.g. back-prop) to improve an energy-based model.

In general, not clear how to sample efficiently from energy-based models. Special case of RBM works if we take units in pairs with the opposite weights and bias. Gives a recipe for incrementally adding hidden units to an RBM.

Why Dimensionality Reduction May Be Wrong

- Usual story for dimensionality reduction: unsupervised transformation into more relevant space which captures the main factors of variation = manifold. Throw noise out.
- *Manifold near which the data concentrates = set of constraints*
- violations of these constraints can be very important
- **Dimensionality reduction forces EVERY input example to be compressed into the same nb bits**
- Better: want small **AVERAGE** nb of bits per example: want *small entropy*
- Imagine animal with mostly boring useless images (take few bits) and few important and complicated images (prey, predator)... who will survive?

A Curriculum for Training AIs?

- Learning high-level abstractions is fundamentally difficult (NP-hard in the worst case).
- Experiments suggest that while trying to optimize directly a deep architecture is difficult, one can **first teach it the lower-level concepts and once they are mastered show it more advanced concepts** based on previously learned ones.
- Humans do not learn very abstract concepts early on! It takes 20 years of guided learning to obtain a math major!
- Why should we expect computers to need less help?
- Future AI research may require help from educators to design an appropriate sequence of learning tasks for AIs.

Critical Mass Hypotheses

Two forms:

- As we train with more and richer data, using models with more and more capacity (but maybe not much more sophisticated than current DBN technology), abstractions that were apparently difficult to learn will be learned more easily (because of the reinforcing signals from many sources about the usefulness of these abstractions).
- As we build better models, there will come a time when the learning systems can take much better advantage of the massive amount of image/video/text data available from human culture (e.g. through the web).

Nailing the Semantics

What is missing from today's AI and ML research? Nailing the semantics.

What would be more impressive in terms of AI?

1. Beating all other language models by ϵ in perplexity? or
2. Demonstrating a learning algorithm that can *nail the semantics* of a "baby" virtual world (where current state-of-the-art still fails miserably).

I vote for the 2nd: work with a small vocabulary, language data associated with **meaning** (e.g., images).

+ no guarantee that low perplexity $\Rightarrow$ language understanding!

The Baby AI Project

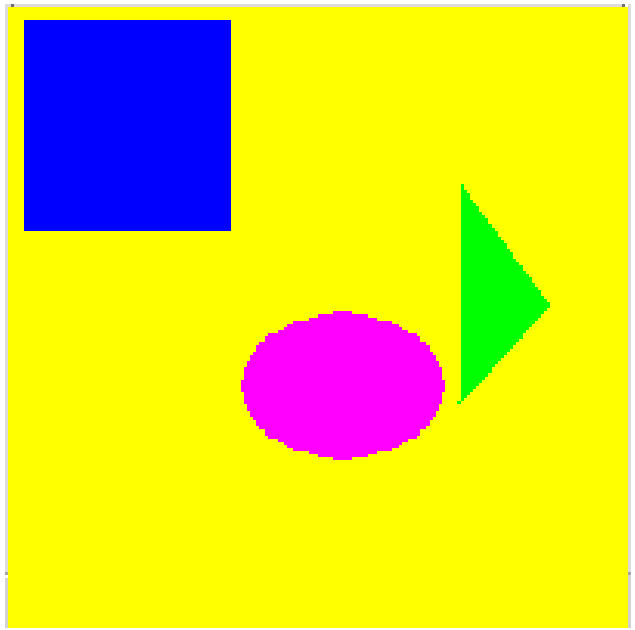
Learning multiple levels of representation / abstraction from multiple modalities: language + video

- Huge amounts of data → online learning
- Deep architectures = multiple levels of abstractions (both context and input)
  necessary to avoid curse of dimensionality
- Learn more abstract concepts on top of / after simpler ones
  e.g. first simple static shapes, then video + naming actions & movement
- Teach the Baby AI with a sequence of tasks = curriculum
- Use mostly unlabeled data, exploit the little labeled data as hints
- AI should understand semantics of language + images: multi-modal architectures

We already have a data generator for a simple "block world" (at different degrees of complexity) with images and natural language.

Baby AI Datasets

We created programs that generate artificial images and accompanying text.
Images contain 1, 2, 3 or 4 objects (2-D shapes).
Factors of variation: size, position, rotation, shape, color, combinatorics over multiple objects.



As many examples as wanted can be generated.

Tasks are already very challenging for current learning algorithms.

Baby AI Datasets

Text is "natural language"-like. One or few sentences about the image. Statement or question + answer pair.

Statement + image pairs can be used to set up match/nomatch classification tests.

Question/answer pairs can be used to set up more standard classification problems (question + image $\Rightarrow$ answer) for comparisons with state-of-the-art classifiers.

Topic	**Question given to the computer**	**Answer**
Color	There is a small triangle. What color is it?	Green
Shape	What is the shape of the green object?	Triangle
Location	Is the blue square at the top or at the bottom?	At the top
Size	There is a triangle on the right.	
	Is it rather small or bigger?	Small
Size (relative)	Is the square smaller or bigger than the triangle?	Bigger

Datasets and python scripts will soon be released on our web page.

Space-Time Hierarchy

Efficient information propagation $\Rightarrow$ Space-Time Hierarchy

Unsupervised learning / discovering the dependencies between N (large) variables organized topologically in 1D (long sequence) or 2D (big image).

- Fully connected graphical model with latent variable (e.g. huge RBM): no depth, too many parameters. Every variable 2 hops away from every other one.
- Locally connected (dynamical Bayes net, recurrent net, MRF): most variable pairs are $O(N)$ or $O(\sqrt{N})$ hops from each other.
- **Hierarchical structure: all variables are no more than $O(\log N)$ hops from each other.**
  Gradient needs only to propagate through $O(\log N)$ levels.

Hierarchical structure useful EVEN IF THE DATA DO NOT HAVE AN INHERENT HIERARCHICAL STRUCTURE.

Culture and Language as Optimization Techniques

Since optimizing brains is NP-hard and frought with local minima, nature may well have used culture and language to make the whole human species a large optimization machine for the space of abstractions (= ideas) helpful to humans:

- Abstractions that have worked in the past are taught to us through language.
- Human society (over generations) performs a random search in the space of neurally implementable abstractions.
- Language is a *low-capacity discrete channel* to hint at the good abstractions previously discovered. Only high-level and discrete abstractions can be verbalized.
- We propagate what we believe are good ideas as well as our new ideas (even more important).
- We can only verify the utility of some of them...

Previous work in this direction: (Hutchins and Hazlehurst, 1995)

Conclusions

- For AI $\Rightarrow$ must learn **high level abstractions** efficiently
  $\Rightarrow$ deep architectures (statistical efficiency)
- Local or shallow learning algorithms (kernels, trees, local non-parametric) are hit by the curse of dimensionality
- Deep architectures not trainable? computational efficiency? new methods appear to **break through the obstacle**
- Basic principle: greedy layer-wise unsupervised
- RBMs as building blocks. Are universal approximators for discrete distributions. CD works well.
- Deep neural networks can learn high-level abstractions by first learning simpler concepts and then composing them $\Rightarrow$ curriculum
- Interesting open questions remain about global training of DBNs.
- Training a deep net is hard: higher-level abstractions may require parallel search, analog to human society as learning device.

Allender, E. (1996).
Circuit complexity before the dawn of the new millennium.
In *16th Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 1–18. Lecture Notes in Computer Science 1180.

Belkin, M. and Niyogi, P. (2002).
Laplacian eigenmaps and spectral techniques for embedding and clustering.
In Dietterich, T., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA. MIT Press.

Bengio, Y., Delalleau, O., and Le Roux, N. (2006).
The curse of highly variable functions for local kernel machines.
In Weiss, Y., Schölkopf, B., and Platt, J., editors, *Advances in Neural Information Processing Systems 18*, pages 107–114. MIT Press, Cambridge, MA.

Bengio, Y., Delalleau, O., Le Roux, N., Paiement, J.-F., Vincent, P., and Ouimet, M. (2004).
Learning eigenfunctions links spectral embedding and kernel PCA.
Neural Computation, 16(10):2197–2219.

Bengio, Y., Delalleau, O., and Simard, C. (2007).
Decision trees do not generalize to new variations.
Technical report, Dept. IRO, Université de Montréal.

Bengio, Y., Ducharme, R., and Vincent, P. (2001).
A neural probabilistic language model.
In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 932–938. MIT Press.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003).
A neural probabilistic language model.
Journal of Machine Learning Research, 3:1137–1155.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007).
Greedy layer-wise training of deep networks.
In Schölkopf, B., Platt, J., and Hoffman, T., editors, *Advances in Neural Information Processing Systems 19*. MIT Press.

Bengio, Y., Larochelle, H., and Vincent, P. (2006).
Non-local manifold parzen windows.
In Weiss, Y., Schölkopf, B., and Platt, J., editors, *Advances in Neural Information Processing Systems 18*, pages 115–122. MIT Press.

Bengio, Y. and Le Cun, Y. (2007).
Scaling learning algorithms towards AI.
In Bottou, L., Chapelle, O., DeCoste, D., and Weston, J., editors, *Large Scale Kernel Machines*. MIT Press.

Bengio, Y. and Monperrus, M. (2005).
Non-local manifold tangent learning.
In Saul, L., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 17*. MIT Press.

Bishop, C. (2006).
Pattern Recognition and Machine Learning.
Springer.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984).
Classification and Regression Trees.
Wadsworth International Group, Belmont, CA.

DeMers, D. and Cottrell, G. (1993).
Non-linear dimensionality reduction.
In Giles, C., Hanson, S., and Cowan, J., editors, *Advances in Neural Information Processing Systems 5*, pages 580–587, San Mateo CA. Morgan Kaufmann.

Freund, Y. and Schapire, R. E. (1996).
Experiments with a new boosting algorithm.
In *Machine Learning: Proceedings of Thirteenth International Conference*, pages 148–156.

Guillery, R. (2005).
Is postnatal neocortical maturation hierarchical?
Trends in Neuroscience, 28(10):512–517.

Hastad, J. T. (1987).
Computational Limitations for Small Depth Circuits.
MIT Press.

Hinton, G. (1986).
Learning distributed representations of concepts.
In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 1–12, Amherst 1986. Lawrence Erlbaum, Hillsdale.

Hinton, G. (2002).
Training products of experts by minimizing contrastive divergence.
Neural Computation, 14:1771–1800.

Hinton, G., Dayan, P., Frey, B., and Neal, R. (1995).
The wake-sleep algorithm for unsupervised neural networks.
Science, 268:1558–1161.

Hinton, G. and Ghahramani, Z. (1997).
Generative models for discovering sparse distributed representations.
Philosophical Transactions of the Royal Society of London, B(352):1177–1190.

Hinton, G. E., Osindero, S., and Teh, Y. (2006).
A fast learning algorithm for deep belief nets.
Neural Computation, 18:1527–1554.

Hinton, G. E., Osindero, S., Welling, M., and Teh, Y.-W. (2006).
Unsupervised discovery of non-linear structure using contrastive backpropagation.
Cognitive Science, 30(4).

Hinton, G. E. and Salakhutdinov, R. (2006).
Reducing the Dimensionality of Data with Neural Networks.
Science, 313:504–507.

Hinton, G. E. and Zemel, R. S. (1994).
Autoencoders, minimum description length, and Helmholtz free energy.
Advances in Neural Information Processing Systems, 6:3–10.

Hutchins, E. and Hazlehurst, B. (1995).
How to invent a lexicon: the development of shared symbols in interaction.
In Gilbert, N. and Conte, R., editors, *Artificial Societies: the computer simulation of social life*, pages 157–189. London: UCL Press.

Lang, K. J. and Hinton, G. E. (1988).
The development of the time-delay neural network architecture for speech recognition.
Technical Report CMU-CS-88-152, Carnegie-Mellon University.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. (2007).
An empirical evaluation of deep architectures on problems with many factors of variation.
In *Twenty-fourth International Conference on Machine Learning (ICML'2007)*.

LeCun, Y. and Bengio, Y. (1995).
Convolutional networks for images, speech, and time-series.
In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*. MIT Press.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., and Jackel, L. (1989).
Backpropagation applied to handwritten zip code recognition.
Neural Computation, 1(4):541–551.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998a).
Gradient-based learning applied to document recognition.
Proceedings of the IEEE, 86(11):2278–2324.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998b).
Gradient based learning applied to document recognition.
Proceedings of the IEEE, 86(11):2278–2324.

Moody, J. and Darken, C. (1989).
Fast learning in networks of locally-tuned processing units.
Neural Computation, 1:281–294.

Niranjan, M. and Fallside, F. (1990).
Neural networks and radial basis functions in classifying static speech patterns.
Computer Speech and Language, 4:275–289.

Orr, G. and Muller, K.-R., editors (1998).
Neural networks: tricks of the trade, volume 1524 of *Lecture Notes in Computer Science*.
Springer-Verlag Inc., New York, NY, USA.

Paccanaro, A. and Hinton, G. (2000).
Extracting distributed representations of concepts and relations from positive and negative propositions.
In *Proceedings of the International Joint Conference on Neural Network, IJCNN'2000*, Como, Italy. IEEE, New York.

Piaget, J.-P. (1952).
The origins of intelligence in children.
International Universities Press, New York.

Rumelhart, D., Hinton, G., and Williams, R. (1986).
Learning representations by back-propagating errors.
Nature, 323:533–536.

Rumelhart, D., McClelland, J., and the PDP Research Group (1986).
Parallel Distributed Processing: Explorations in the Microstructure of Cognition, volume 1.
MIT Press, Cambridge.

Saul, L. and Roweis, S. (2002).
Think globally, fit locally: unsupervised learning of low dimensional manifolds.
Journal of Machine Learning Research, 4:119–155.

Schölkopf, B., Smola, A., and Müller, K.-R. (1996).
Nonlinear component analysis as a kernel eigenvalue problem.
Technical Report 44, Max Planck Institute for Biological Cybernetics, Tübingen, Germany.

Tenenbaum, J., de Silva, V., and Langford, J. (2000).
A global geometric framework for nonlinear dimensionality reduction.
Science, 290(5500):2319–2323.

Tesauro, G. (1992).
Practical issues in temporal difference learning.
Machine Learning, 8:257–277.

Utgoff, P. and Stracuzzi, D. (2002).
Many-layered learning.
Neural Computation, 14:2497–2539.

Weiss, Y. (1999).
Segmentation using eigenvectors: a unifying view.
In *Proceedings IEEE International Conference on Computer Vision*, pages 975–982.

Welling, M., Rosen-Zvi, M., and Hinton, G. (2005).
Exponential family harmoniums with an application to information retrieval.
In Saul, L., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 17*. MIT Press.

Welling, M., Zemel, R., and Hinton, G. E. (2003).
Self-supervised boosting.
In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*. MIT Press.

Zemel, R. S. (1993).
A Minimum Description Length Framework for Unsupervised Learning.
PhD thesis, University of Toronto.