Learning to represent natural language

Yoshua Bengio

April 26-27, 2014

ICML' 2014
Workshop on Knowledge-Powered Deep Learning for Text Mining

Beijing, China

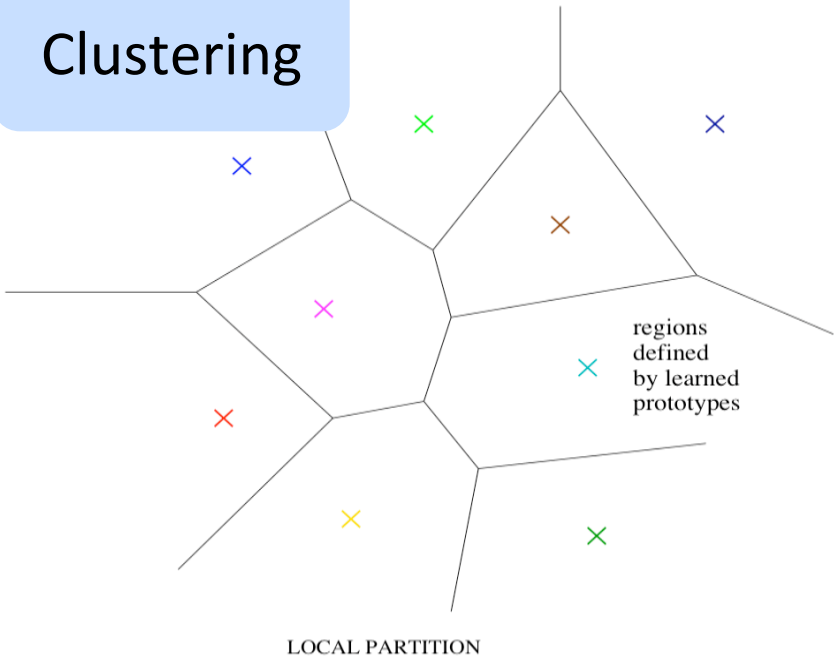
Université de Montréal

LISA

Statistical Considerations

Non-distributed representations

Clustering



- N-grams, Clustering, Nearest-Neighbors, RBF SVMs, local non-parametric density estimation & prediction, decision trees, etc.

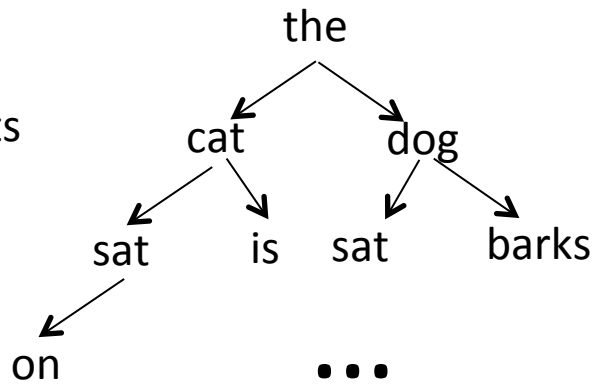
- Parameters for each distinguishable region
- **# of distinguishable regions is linear in # of parameters**

→ No non-trivial generalization to regions without examples

Why N-grams have poor generalization

- For fixed N, the function P(next word | last N-1 words) is learned purely from the instances of the specific N-tuples associated with each possible (N-1)-word context. No generalization to other sequences of N words.
- With back-off / smoothing models, there is some (limited) generalization arising from shorter n-grams, for which there is more data, at the price of less specific predictions.

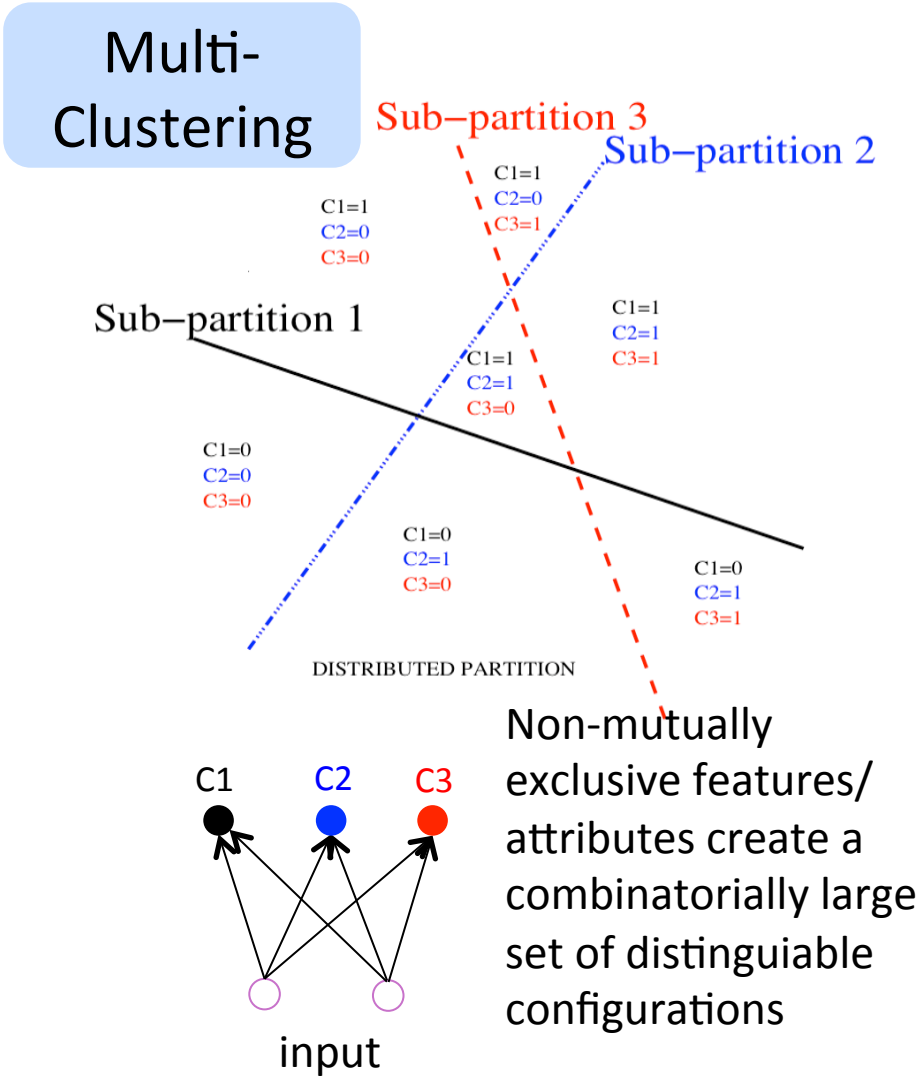
No sharing, where lots would be possible



The power of distributed representations

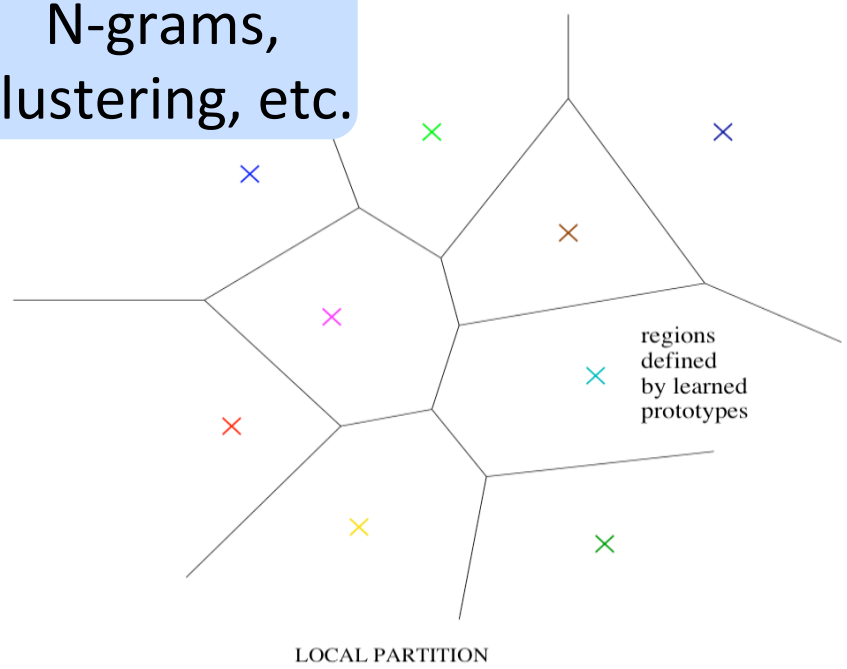
- Factor models, PCA, RBMs, Neural Nets, Sparse Coding, Deep Learning, etc.

- Each parameter influences many regions, not just local neighbors
- **# of distinguishable regions grows almost exponentially with # of parameters**
- **GENERALIZE NON-LOCALLY TO NEVER-SEEN REGIONS**

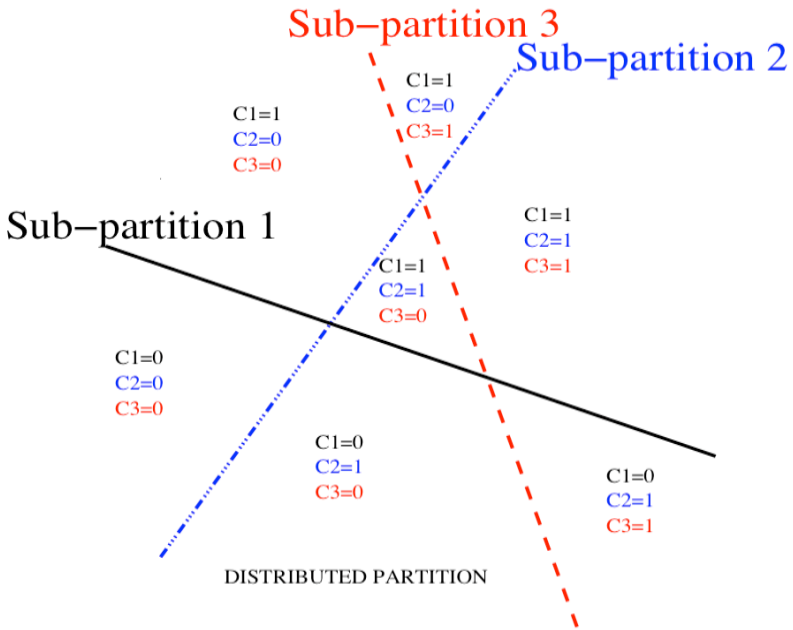


The power of distributed representations

N-grams, clustering, etc.

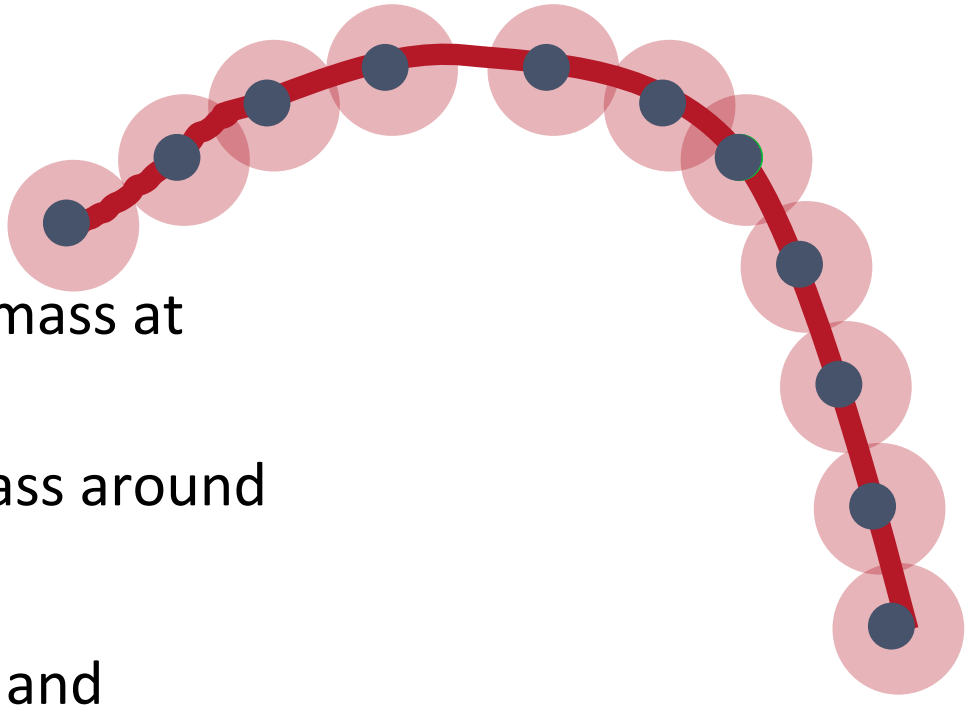


Learned attributes/ embeddings



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

Putting Probability Mass where Structure is Plausible

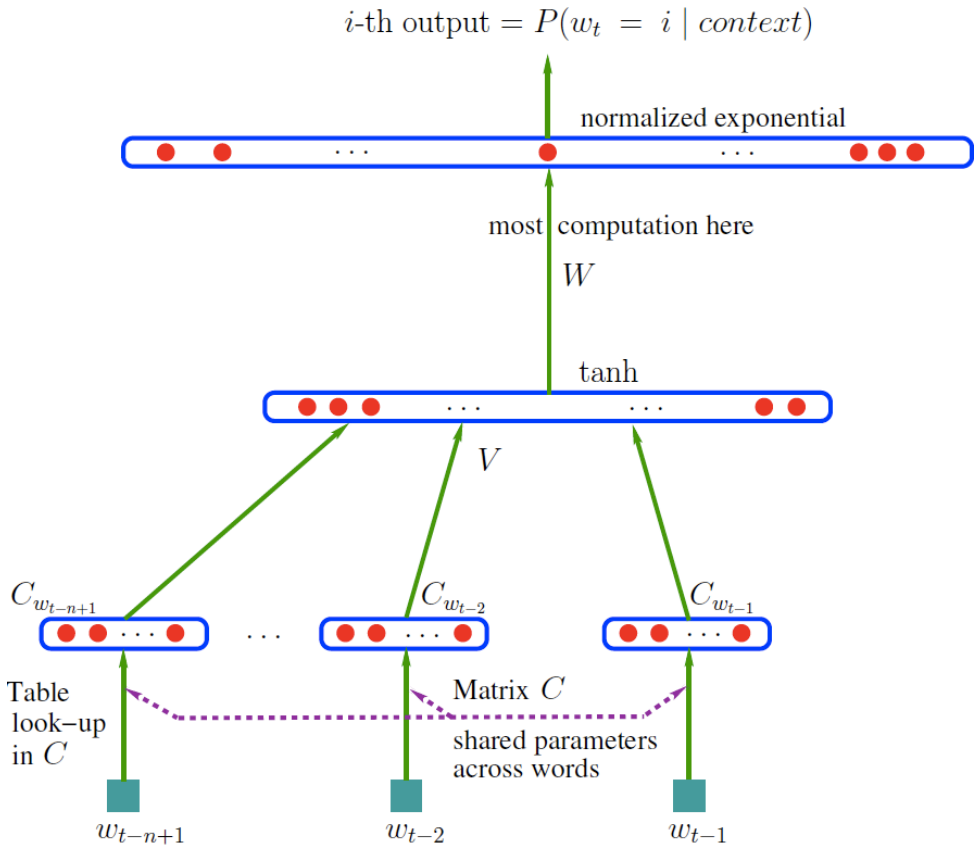


- Empirical distribution: mass at training examples
- Smoothness: spread mass around
- Insufficient
- Guess some 'structure' and generalize accordingly

From the Dark Ages of Neural Nets: the Neural Language Model



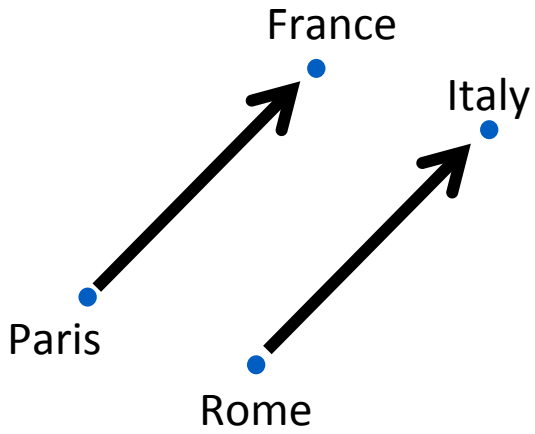
- *Bengio et al NIPS'2000 and JMLR 2003 "A Neural Probabilistic Language Model"*
  - Each word represented by a distributed continuous-valued code vector = embedding
  - Generalizes to sequences of words that are semantically similar to training sequences



(Bengio et al NIPS'2000, JMLR 2003)
Learning Neural Word Embeddings

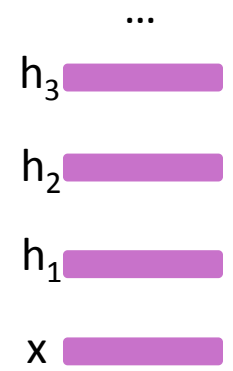
Analogical Representations for Free (Mikolov et al, ICLR 2013)

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



Deep Representation Learning

Learn multiple levels of representation of increasing complexity/abstraction



- theory: exponential gain
- brains are deep
- cognition is compositional
- Better mixing (Bengio et al, ICML 2013)

- **They work! SOTA on industrial-scale AI tasks (object recognition, speech recognition, language modeling, music modeling)**

Deep Architectures are More Expressive

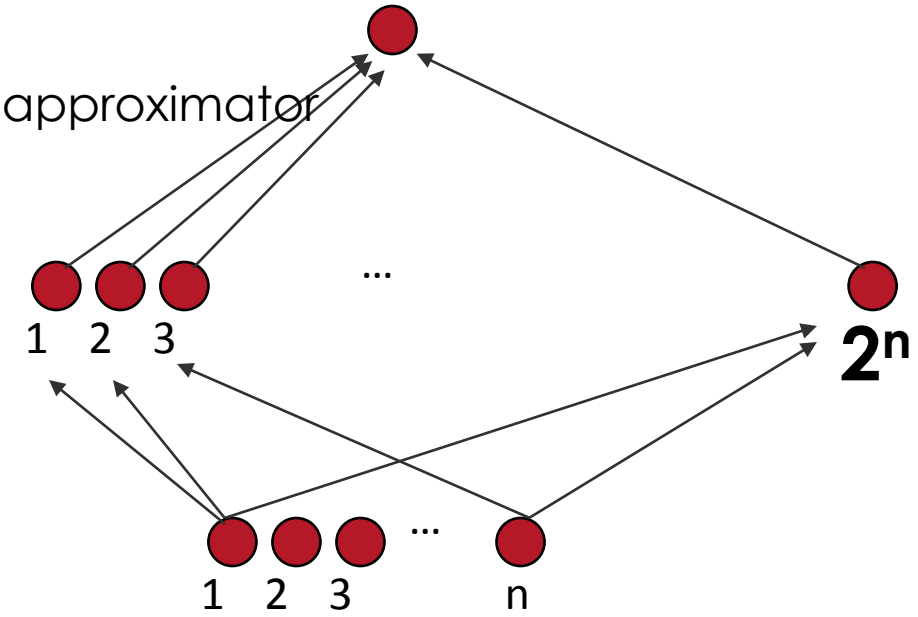
Theoretical arguments:

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

RBMs & auto-encoders = universal approximator

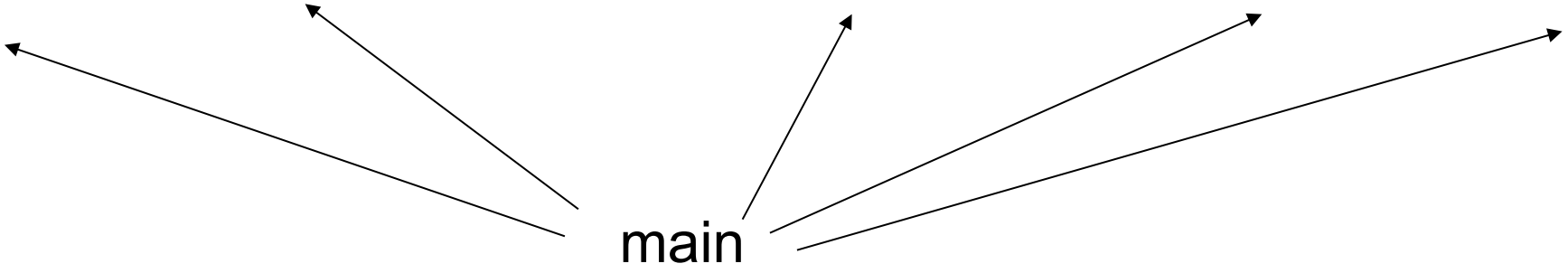
Theorems on advantage of depth:
(Hastad et al 86 & 91, Bengio et al 2007, Bengio & Delalleau 2011, Braverman 2011, Pascanu et al 2014)

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

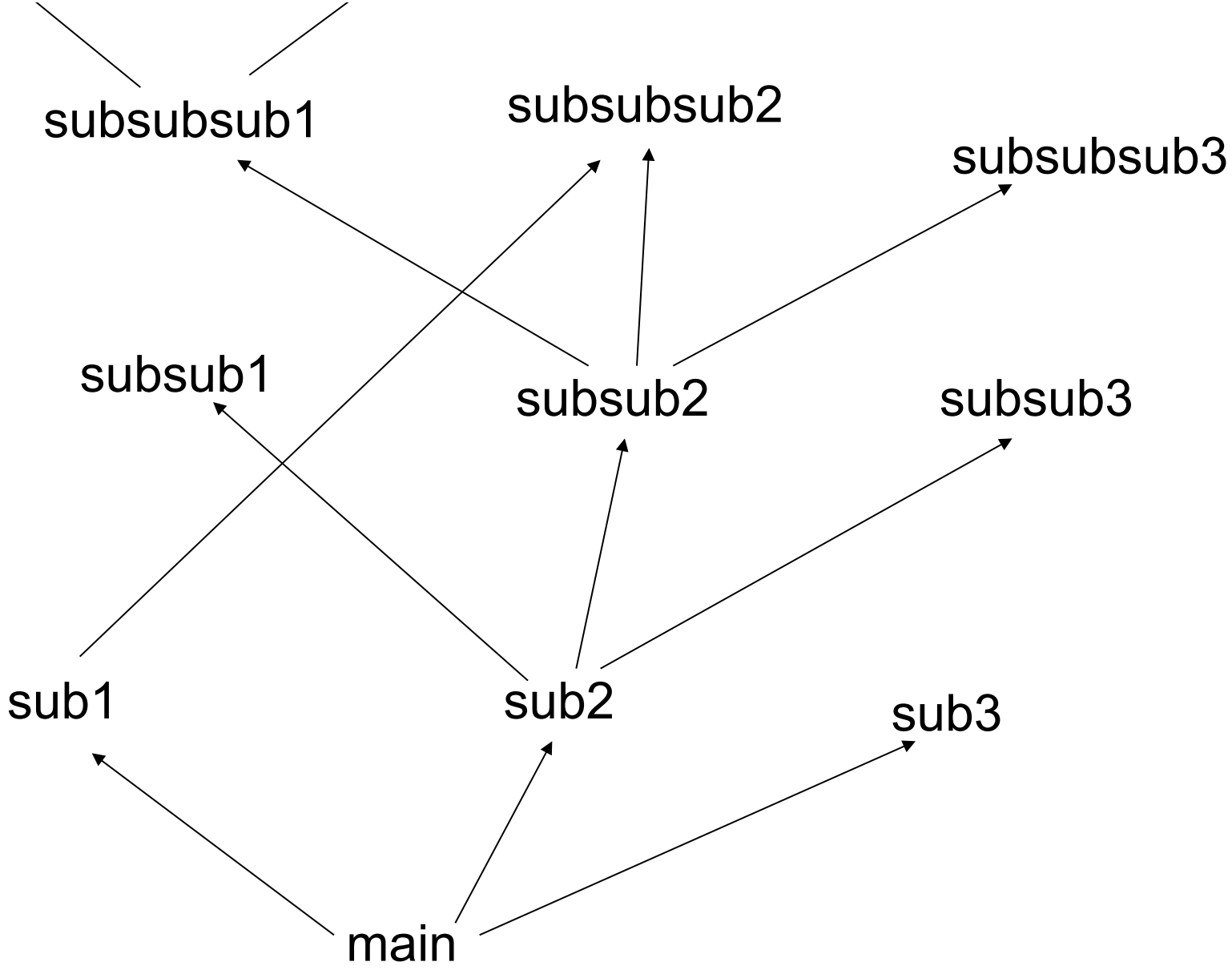


subroutine1 includes
subsub1 code and
subsub2 code and
subsubsub1 code

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



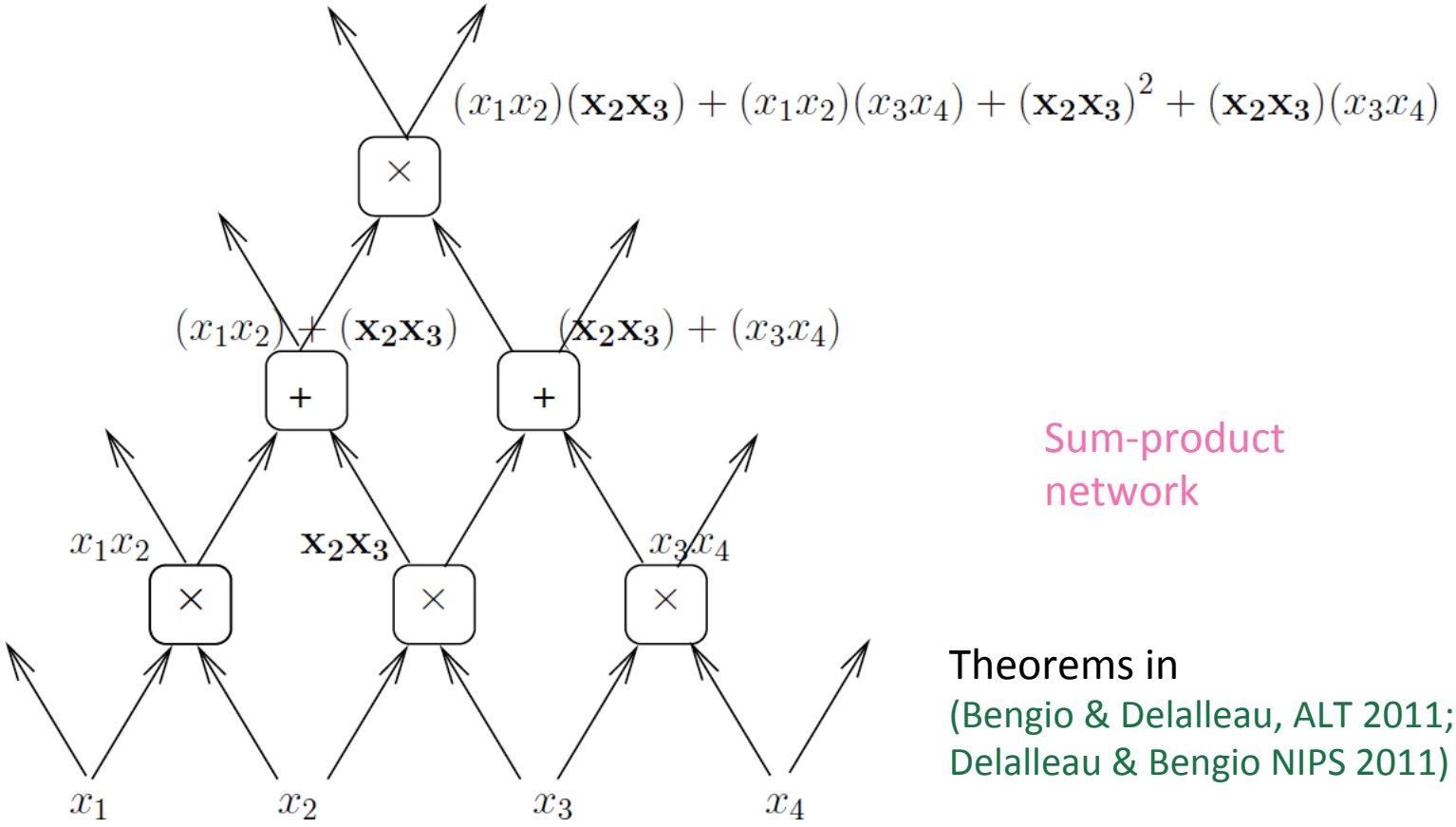
“Shallow” computer program



“Deep” computer program

Sharing Components in a Deep Architecture

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



Sum-product network

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

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

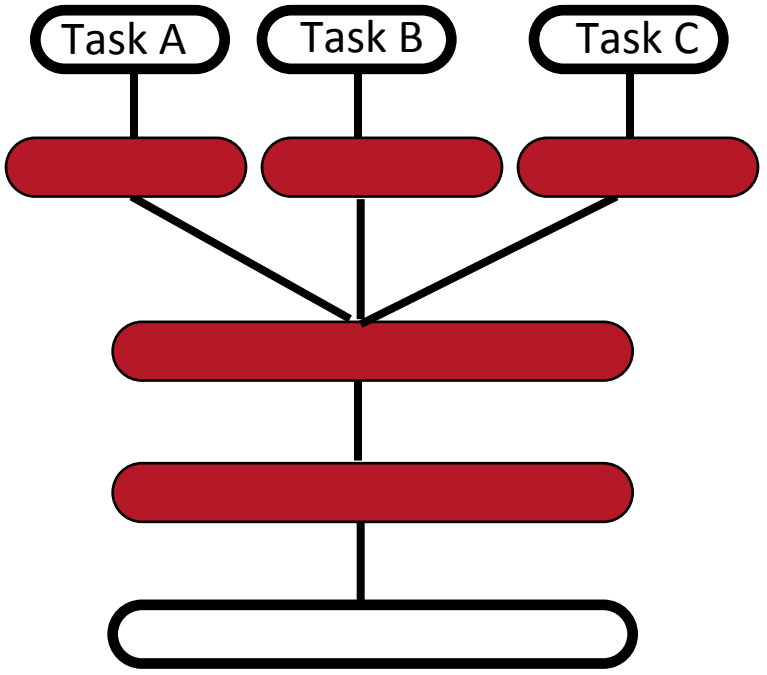
Deep architecture: multiple levels of feature learning

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

Re-Using Features across Tasks: 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**

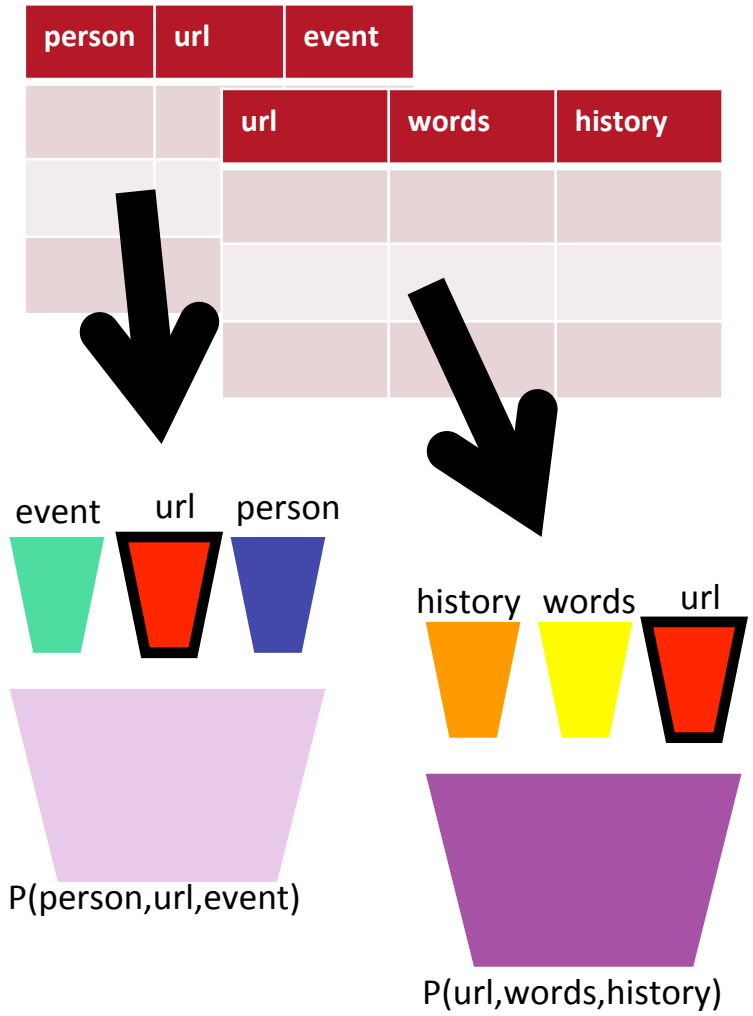


E.g. dictionary, with intermediate concepts re-used across many definitions

Prior: shared underlying explanatory factors between tasks

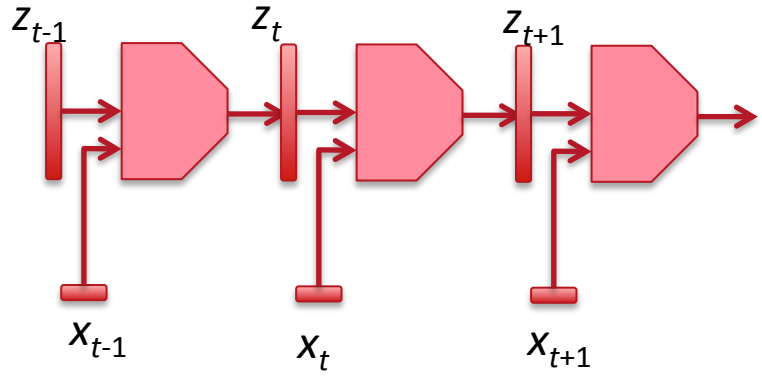
Shared Representations of Entities

- 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**

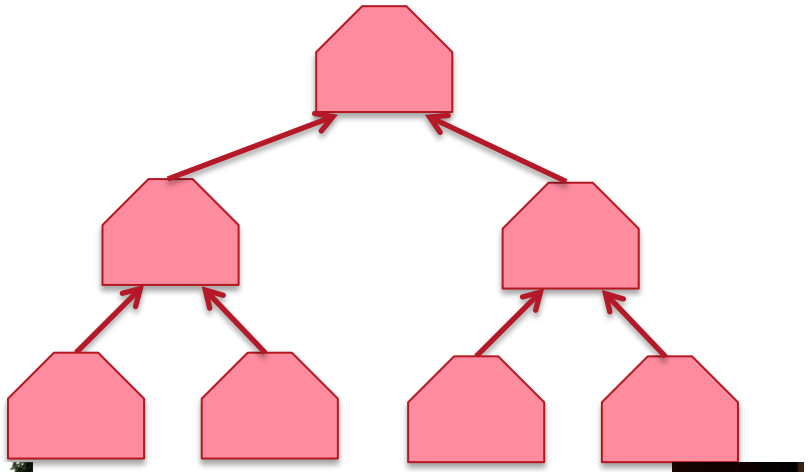


Re-Using Operators Across Time and Levels: Handling the compositionality of human language and thought

- Human languages, ideas, and artifacts are composed from simpler components



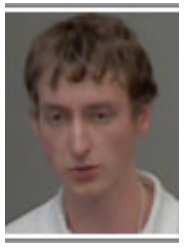
- **Recursion**: the same operator (same parameters) is applied repeatedly on different states/components of the computation



- Result after unfolding = deep computation / representation

(Bottou 2011, Socher et al 2011)

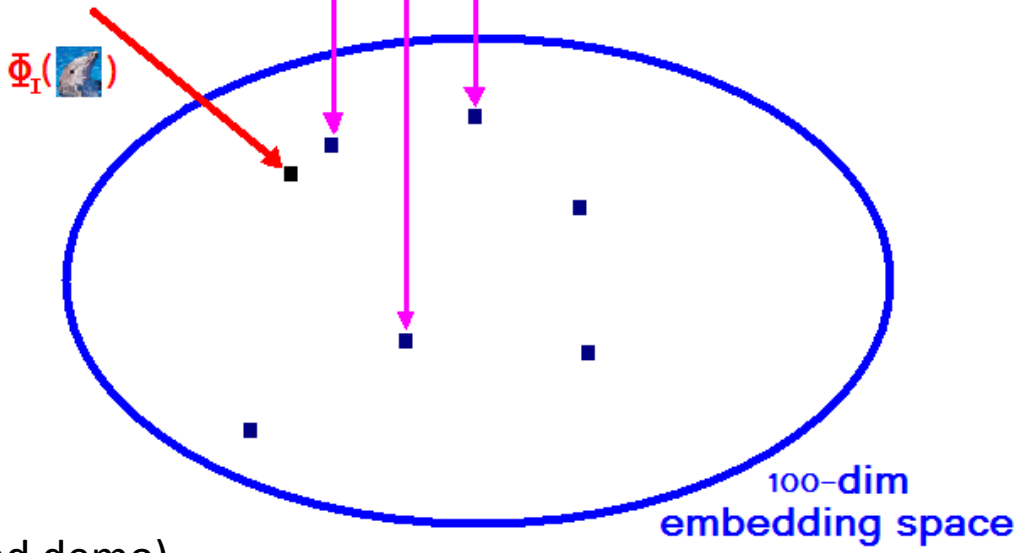
Sharing a Common Representation Space Across Modalities



Google:

S. Bengio, J. Weston & N. Usunier

(IJCAI 2011, NIPS'2010, JMLR 2010, ML J. 2010)



More recently, Salakhutdinov's work (and demo) on multi-modal representation learning from images and text, NIPS'2012, ICML'2014

http://deeplearning.cs.toronto.edu

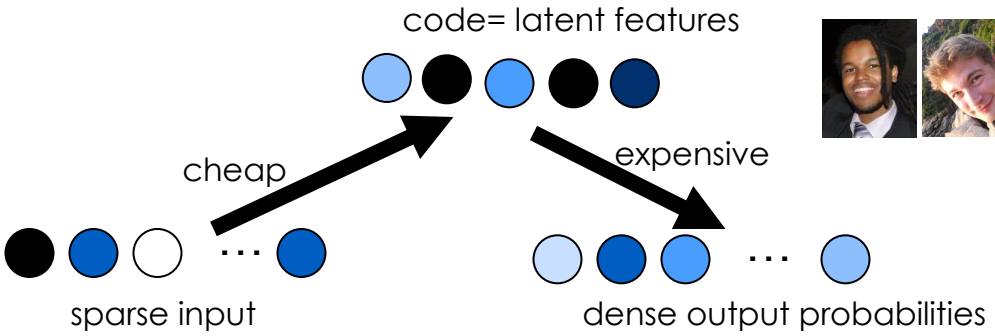
Computational Considerations

Conditional Computation on the Output Layer for Large Vocabularies

- 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 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

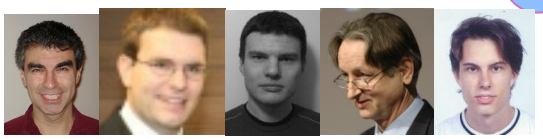
- Auto-encoders and RBMs reconstruct the input, which is sparse and high-dimensional; Language models have a huge output space (1 unit per word).

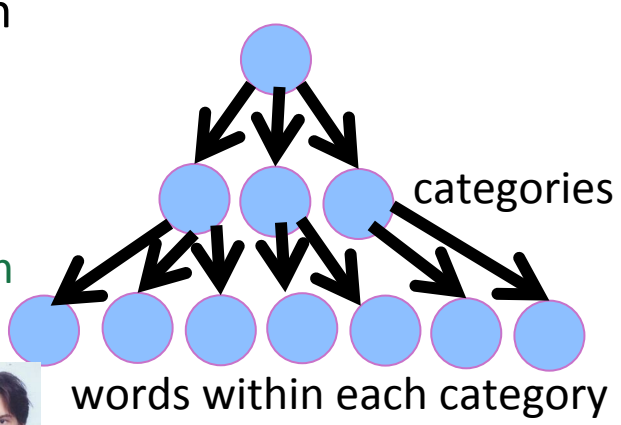


Alternatives to likelihood not requiring the compute the normalization constant, e.g. NCE (Mnih&Kavukcuoglu NIPS 2013)

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

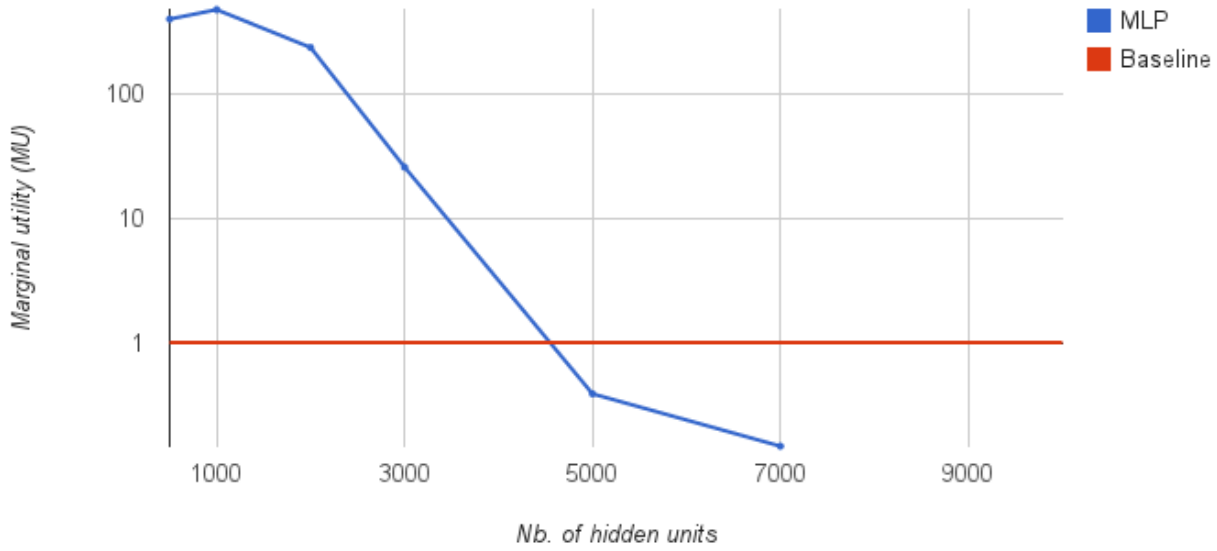
- (Collobert & Weston, ICML 2008) sample a ranking loss
- Decompose output probabilities hierarchically (Morin & Bengio 2005; Blitzer et al 2005; Mnih & Hinton 2007,2009; Mikolov et al 2011)





Optimization & Underfitting

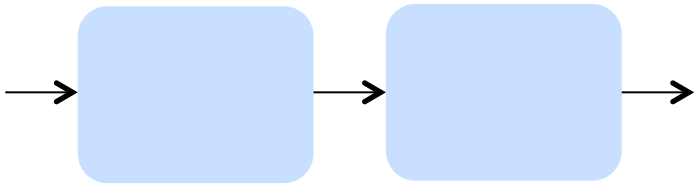
- On large datasets, major obstacle is underfitting
- **Marginal utility** of wider MLPs decreases quickly below memorization baseline



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

Guided Training, Intermediate Concepts

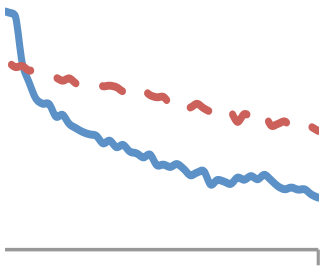
- In (Gulcehre & Bengio ICLR'2013) we set up a task that seems almost impossible to learn by shallow nets, deep nets, SVMs, trees, boosting etc
- Breaking the problem in two sub-problems and pre-training each module separately, then fine-tuning, nails it
- *Need prior knowledge to decompose the task*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

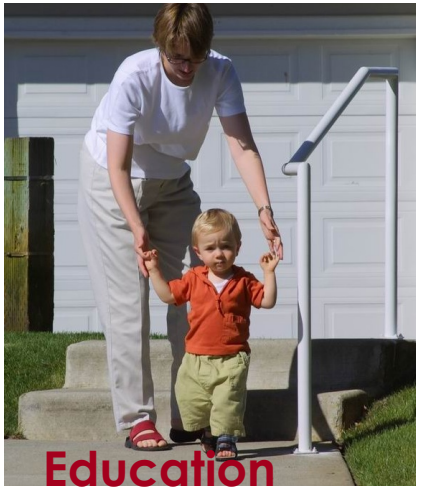


Order & Selection of Examples Matters

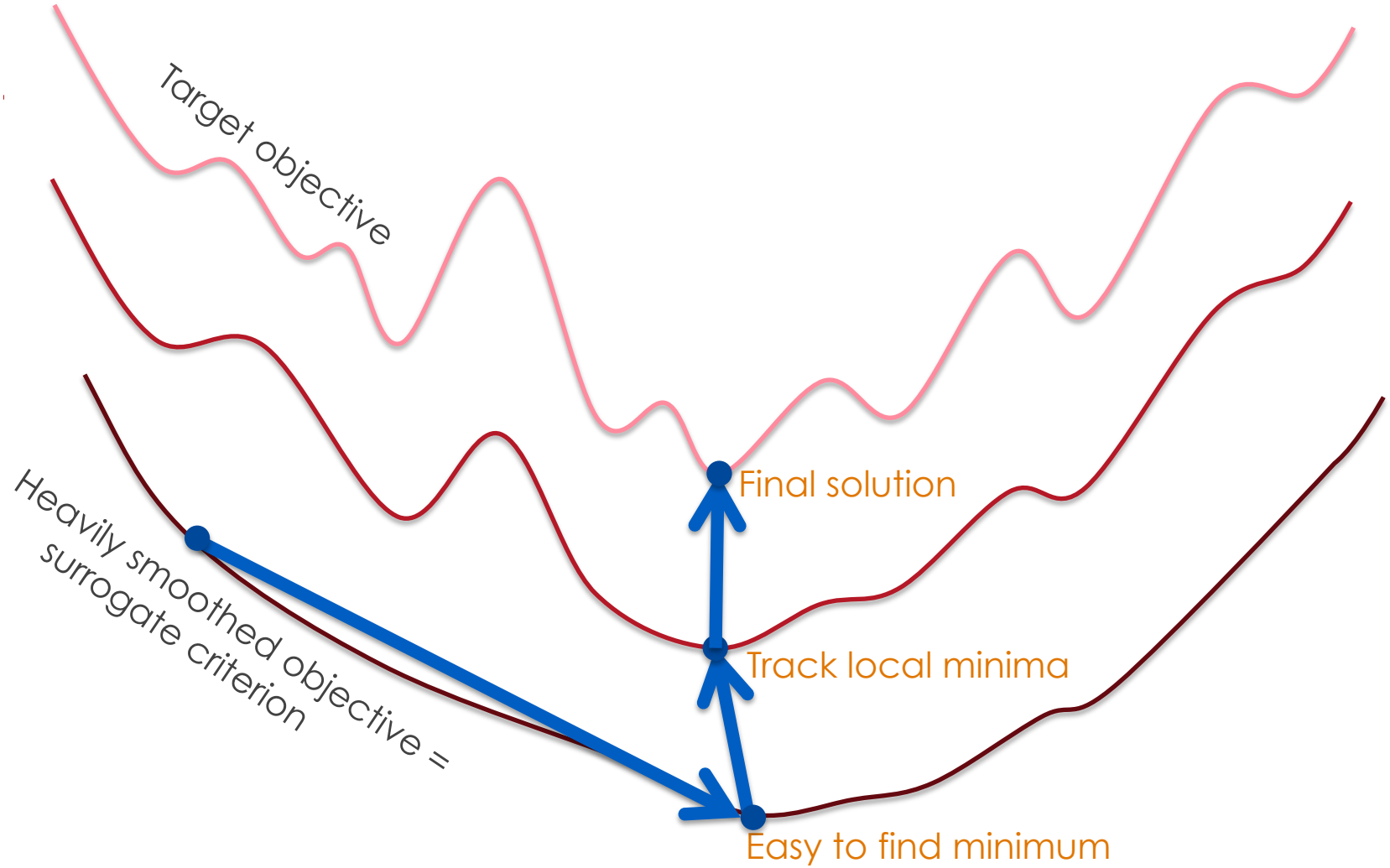
(Bengio, Louradour, Collobert & Weston, ICML'2009)

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





Continuation Methods



Long-Term Dependencies

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

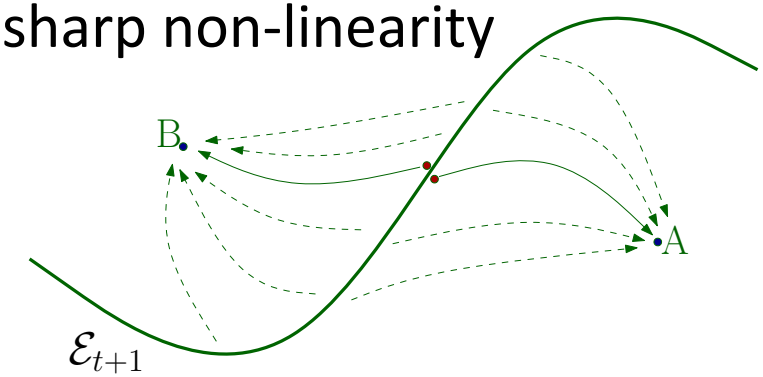
$$L = L(s_T(s_{T-1}(\ldots s_{t+1}(s_t, \ldots))))$$

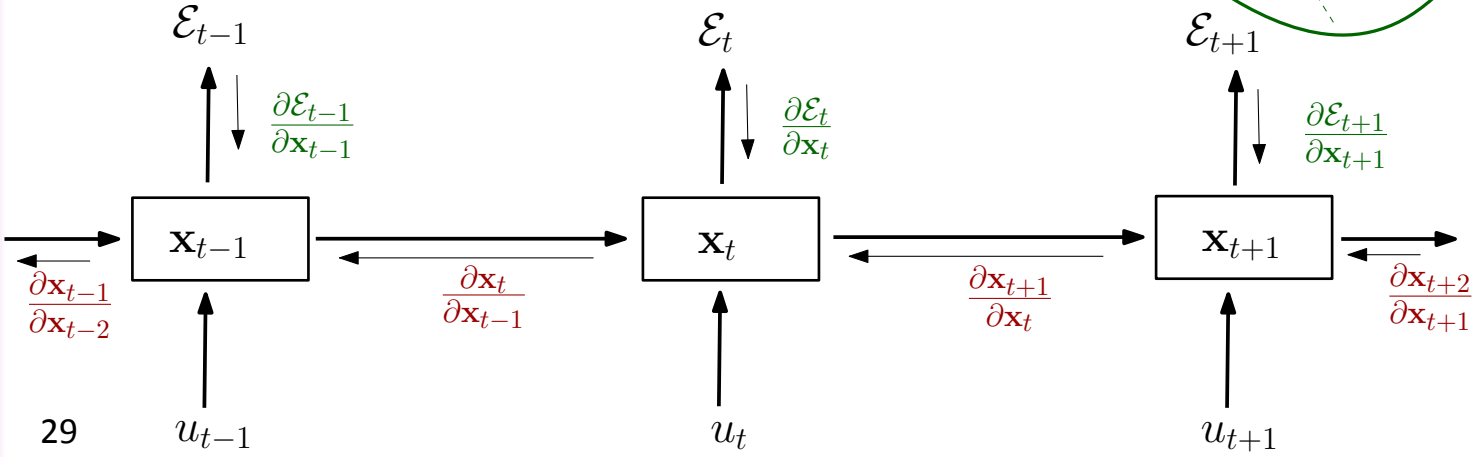
$$\frac{\partial L}{\partial s_t} = \frac{\partial L}{\partial s_T}\frac{\partial s_T}{\partial s_{T-1}}\cdots\frac{\partial s_{t+1}}{\partial s_t}$$

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

The Optimization Challenge in Deep / Recurrent Nets

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

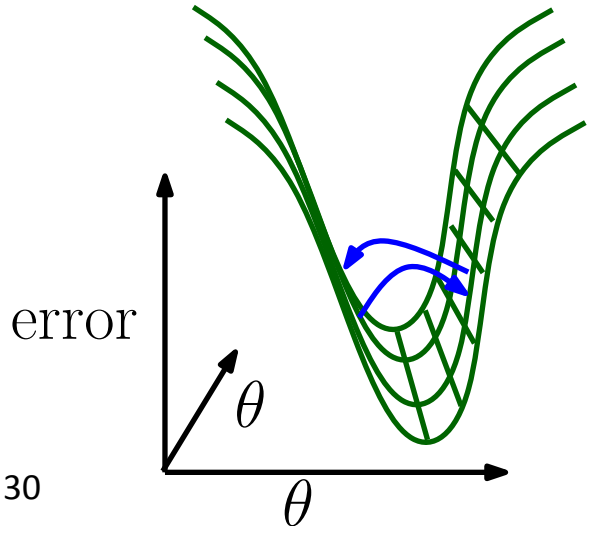


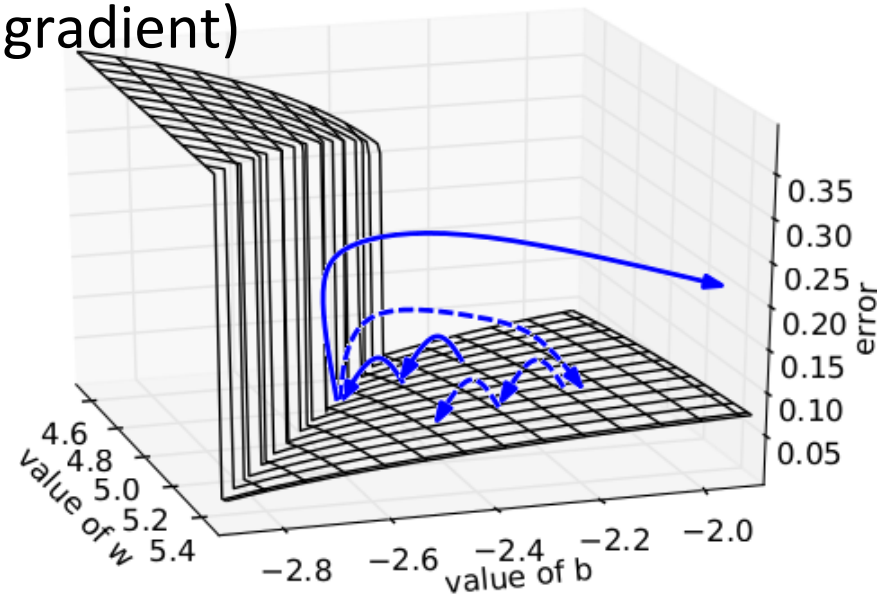


RNN Tricks

(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)





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

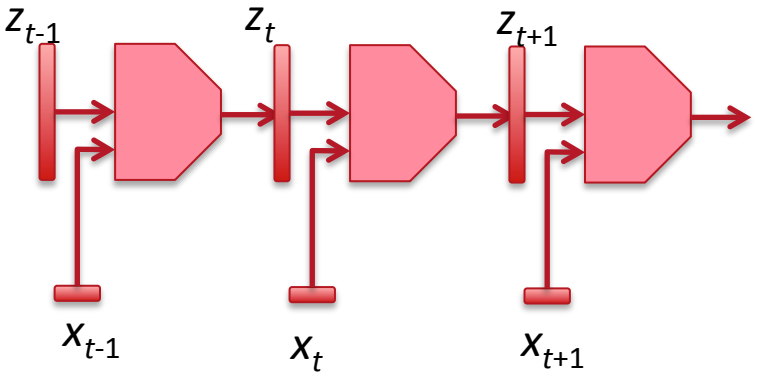
- Exploit those **hints** to **disentangle** better!

  (Becker & Hinton 1993, Wiskott & Sejnowski 2002, Hurri & Hyvarinen 2003, Berkes & Wiskott 2005, Mobahi et al 2009, Bergstra & Bengio 2009)

- RNNs working at different time scales

  (Elhihi & Bengio NIPS'1995), (Koutnik et al ICML 2014)

Long-Term Dependencies and Clipping Trick



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

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

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

Gradient Norm Clipping

$$\hat{\mathbf{g}} \leftarrow \frac{\partial error}{\partial \theta}$$

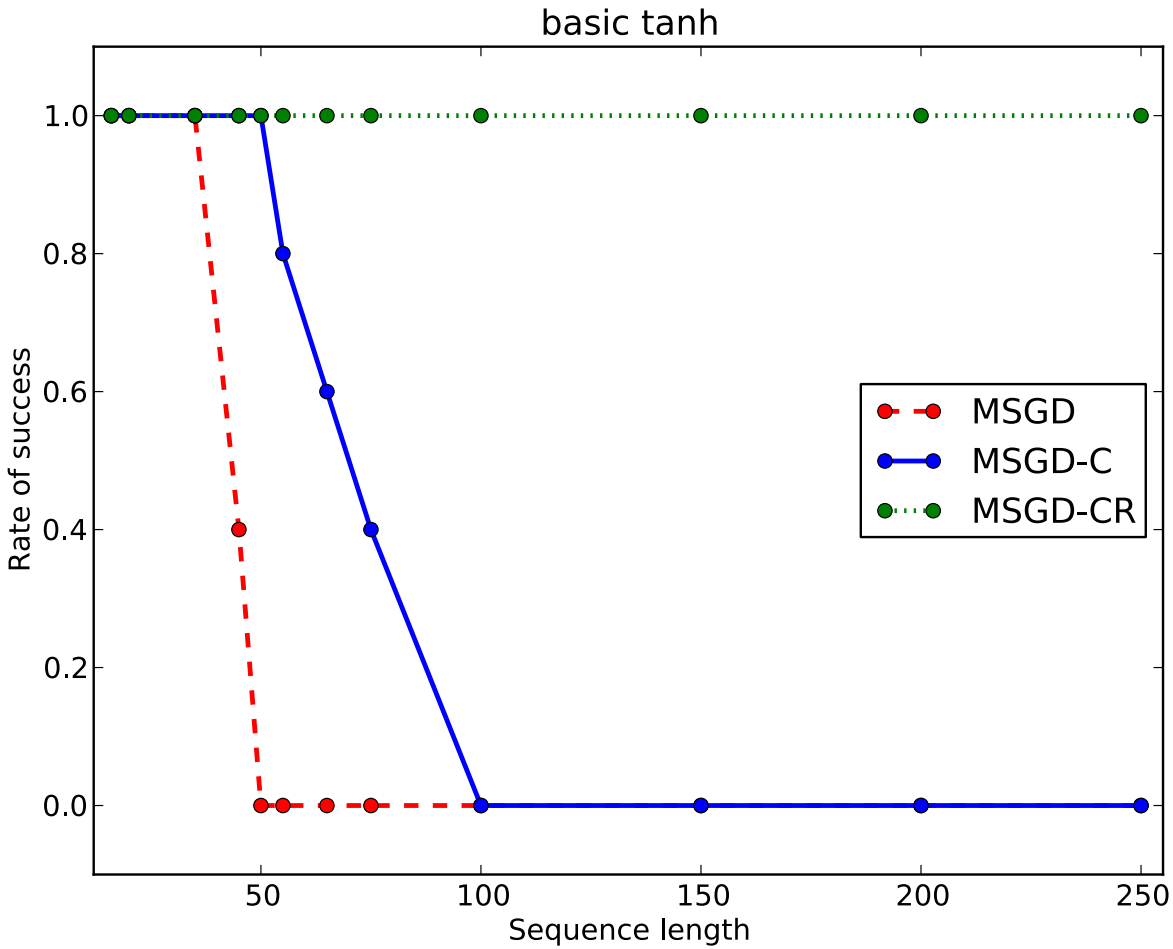
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**

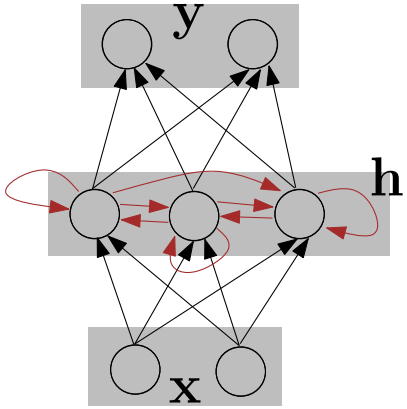
$$\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$$

end if

Combining clipping to avoid gradient explosion and Jacobian regularizer to avoid gradient vanishing

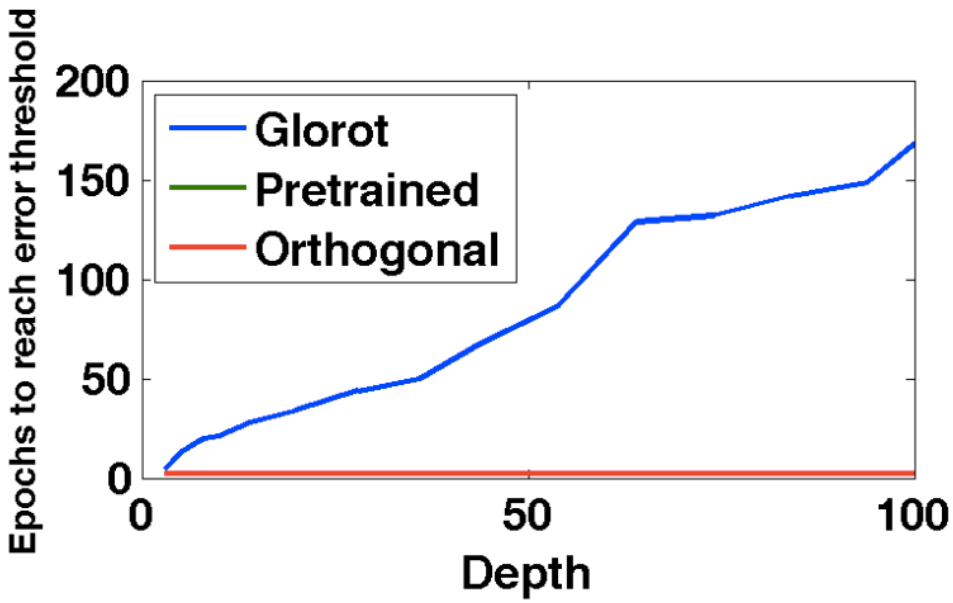
- (Pascanu, Mikolov & Bengio, ICML 2013)





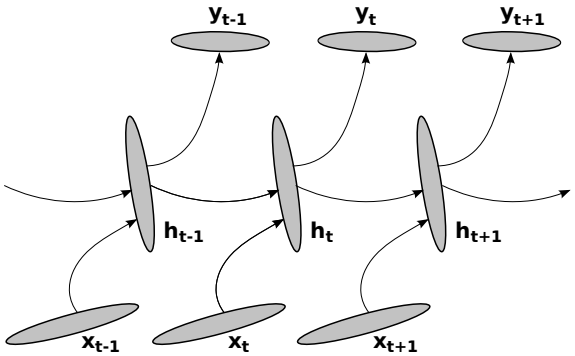
Orthogonal Initialization Works Even Better

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



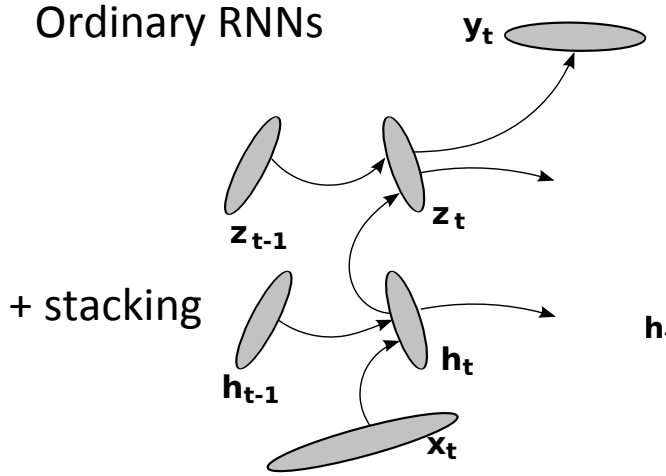
Increasing the Expressive Power of RNNs with more Depth

- ICLR 2014, *How to construct deep recurrent neural networks*

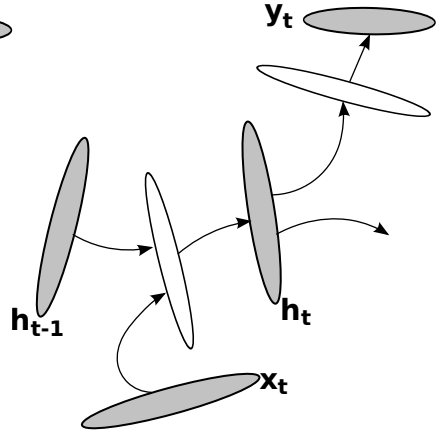


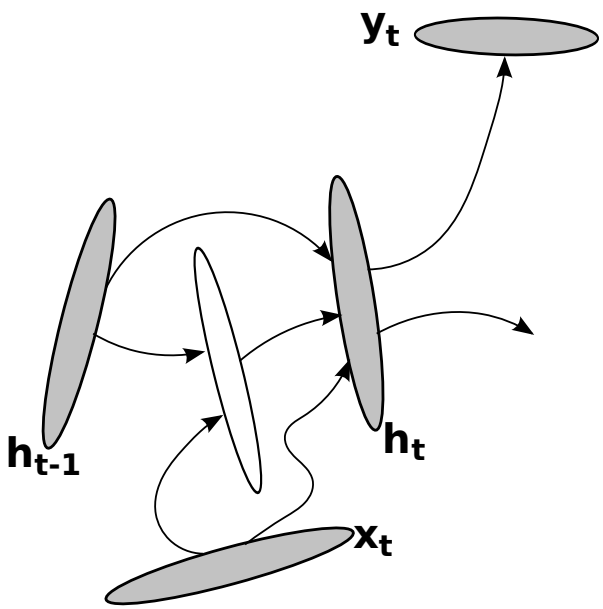
Ordinary RNNs

+ deep hid-to-out
+ deep hid-to-hid
+deep in-to-hid



+ stacking



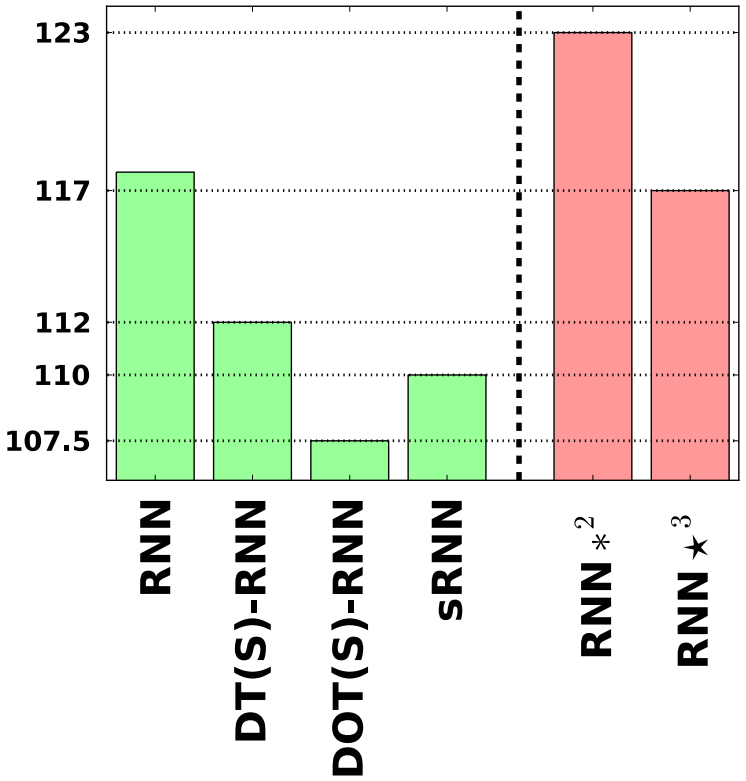


+ skip connections for creating shorter paths

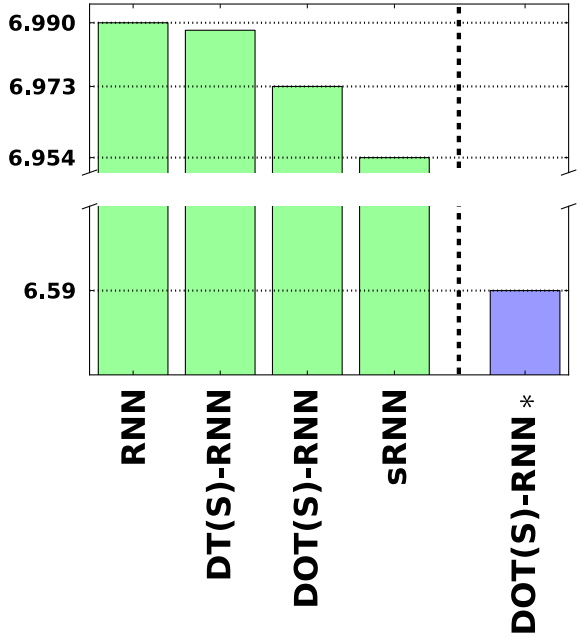
Deep RNN Results

- Language modeling

(Penn Treebank perplexity)



- Music modeling (Muse, NLL)



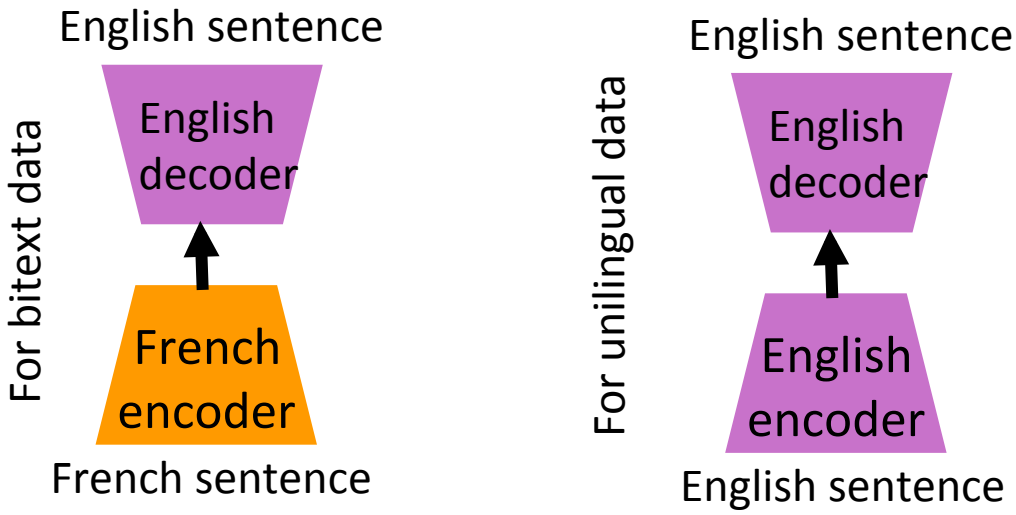
More results in the ICLR 2014 paper

Already Many NLP Applications of DL

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

Encoder-Decoder Framework for Machine Translation

- 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

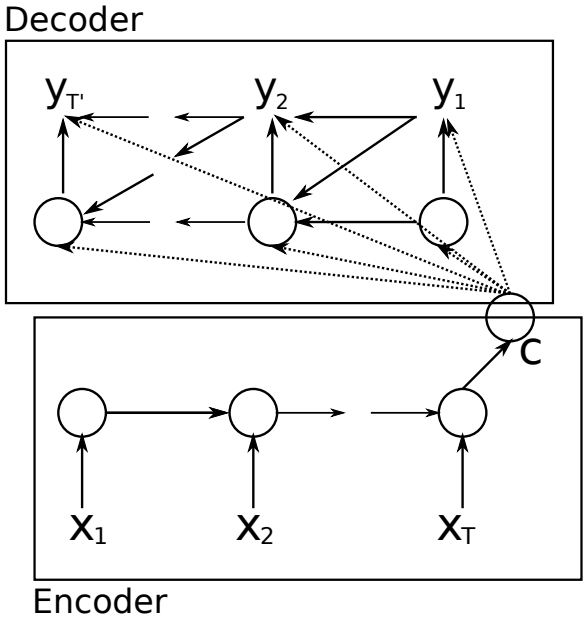


RNNs for Machine Translation

(Cho, Merrienboer, Gulcehre, Bougares, Schwenk, Bengio; arxiv 2014)

Encoder-decoder framework:

- Encoder = 'summarizing' RNN: word sequence → last-state vector = sequence representation $_t$
- Decoder = 'generative' RNN: context C → distribution over word sequences



RNNs for Machine Translation

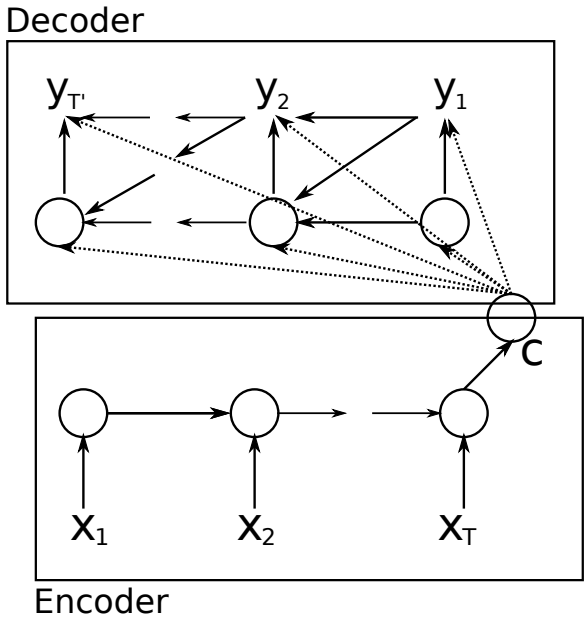
- Decoder = 'generative' RNN: context C → distribution over word sequences
  - $P(Y_1..Y_{T'} \mid C) = \pi P(Y_t \mid H_t, C)$

  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 model: ancestral sampling from Y_1 to $Y_{T'}$.

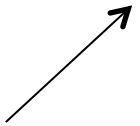
- Output sequence can be of different length T'≠T not necessarily aligned with input sequence



RNNs for MT: Results

- English-French WMT'14 task
- Train on both bilingual (supervised) and unilingual (unsupervised)
- Trained on phrases (phrase table), added into log-linear model of MOSES

Models	BLEU	
	dev	test
Baseline	27.63	29.33
CSLM	28.33	29.58
RNN	28.48	29.96
CSLM + RNN	28.60	30.64
CSLM + RNN + WP	28.93	31.18



+1.85 BLEU points

LISA team: Merci! Questions?