Connectionist Computing
COMP 30230/41390

Gianluca Pollastri

office: E0.95, Science East.

email: gianluca.pollastri@ucd.ie

Credits

- **Geoffrey Hinton, University of Toronto.**
  - **borrowed some of his slides for "Neural Networks" and "Computation in Neural Networks" courses.**

- **Ronan Reilly, NUI Maynooth.**
  - **slides from his CS4018.**

- **Paolo Frasconi, University of Florence.**
  - **slides from tutorial on Machine Learning for structured domains.**

Lecture notes on Brightspace

- **Strictly confidential...**
- **Slim PDF version will be uploaded later, typically the same day as the lecture.**
- **If there is demand, I can upload onto Brightspace last year's narrated slides.. (should be very similar to this year's material)**

Books

- No book covers large fractions of this course.

- Parts of chapters 4, 6, (7), 13 of Tom Mitchell's "Machine Learning"
- Parts of chapter V of Mackay's "Information Theory, Inference, and Learning Algorithms", available online at:

http://www.inference.phy.cam.ac.uk/mackay/itprnn/book.html

- Chapter 20 of Russell and Norvig's "Artificial Intelligence: A Modern Approach", also available at:

http://aima.cs.berkeley.edu/newchap20.pdf

- More materials later..

Marking

- **3 landmark papers to read, and submit a 10-line summary on Brightspace about: each worth 6-7%**
- **a connectionist model to build and play with on some sets, write a report: 30%**
- **Final Exam in the RDS (50%)**

Programming assignment

- Implement a Multi-Layer Perceptron, test it.
- The description on Brightspace.
- Submit through Brightspace code and test results by Dec the 5th at 23:59, any time zone of your choice (Baker Island?).
- 30% of the overall mark
- One third of a grade down every day late, that is: if you deserve an A and you're 1 day late you get an A-, 2 days late a B+, etc.

Assignment 3

- Read the paper "Reducing the Dimensionality of Data with Neural Networks", by Hinton and Salakhutdinov (2006).
- Don't panic!
- The paper is on Brightspace.
- Submit through Brightspace a 400 word MAX summary by Nov 22nd at 23:59, any time zone of your choice (Baker Island?).
- 7% of the overall mark.
- One third of a grade down every 2 days late, that is: if you deserve an A and you're 1-2 days late you get an A-, 3-4 days late a B+, etc.

Summary: NNs and ML

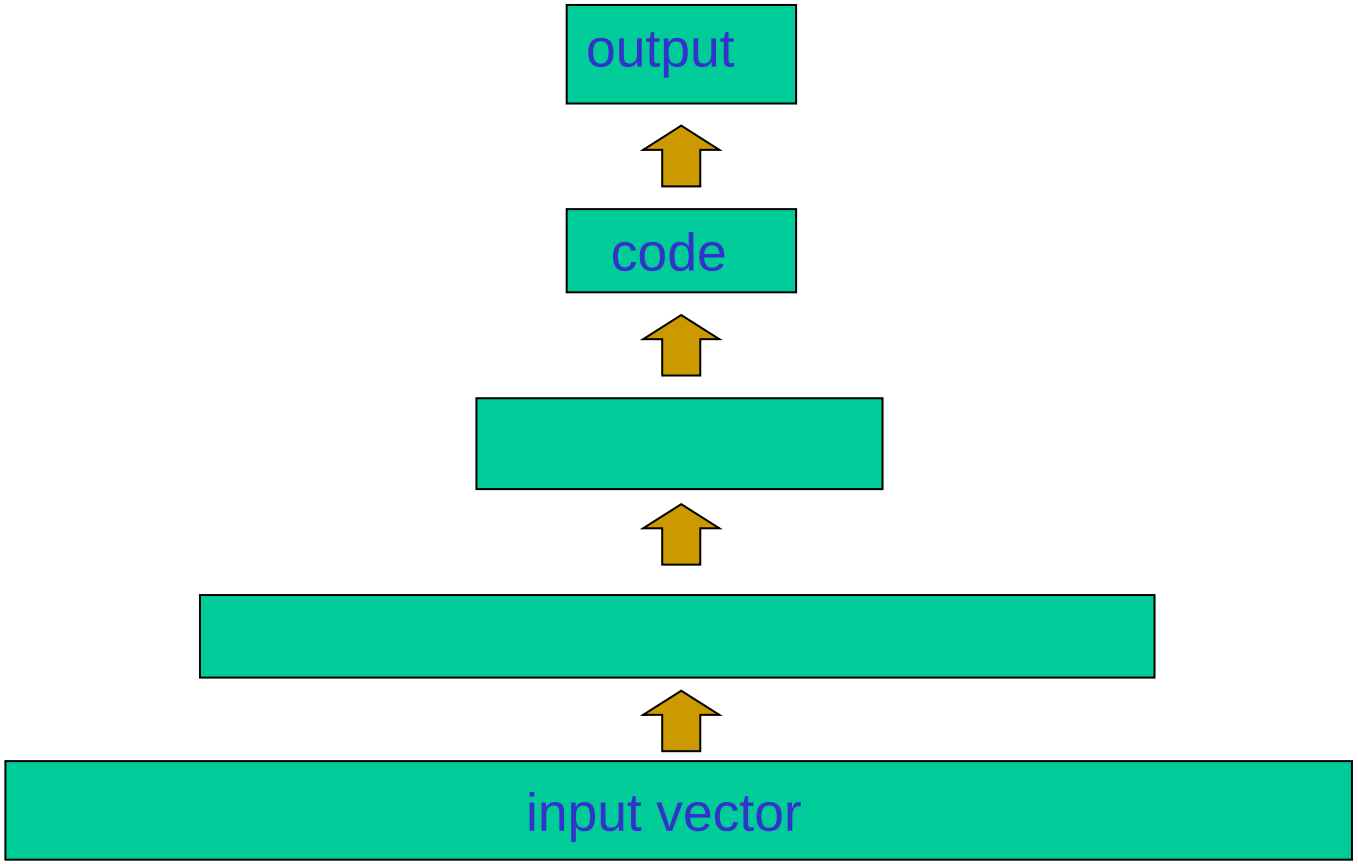
	probabilistic model for P(t|y)	natural output function	error (- log likelihood)
regression	gaussian	linear	squared
binary classification	binomial	sigmoid	relative (mutual) entropy
multi-class classification	multinomial	softmax	relative (cross) entropy

Deep networks

and Deep Learning

Connectionist Computing
COMP 30230

Deep networks?



Deep is hard

- **There are many reasons why deep should work better than shallow.**
- **But anyone who has trained a neural network with many layers knows the more inner layers you add, the longer the initial plateau lasts.**
- **By 5 hidden layers, typically, it is long enough that you can't afford to wait..**

Deep is hard because gradients vanish

- **The problem is that when networks get deep the gradient *vanishes*.**
- **When a network is untrained, the deeper down a hidden unit is, the more subtle its effect is on the outputs.**
- **If it's subtle, it means it doesn't do much to the error if you change it.**

Old solutions

- **Several solutions, often ad hoc, over the years.**
- **Typically they involved tampering with the gradient in a way or another:**

- **The gradient is vanishing? Don't let it..**

Journal of Machine Learning Research 4 (2003) 575-602 Submitted 2/02; Revised 4/03; Published 9/03

The Principled Design of Large-Scale Recursive Neural Network Architectures–DAG-RNNs and the Protein Structure Prediction Problem

Pierre Baldi PFBALDI@ICS.UCI.EDU
Gianluca Pollastri GPOLLAST@ICS.UCI.EDU
School of Information and Computer Science
Institute for Genomics and Bioinformatics
University of California, Irvine
Irvine, CA 92697-3425, USA

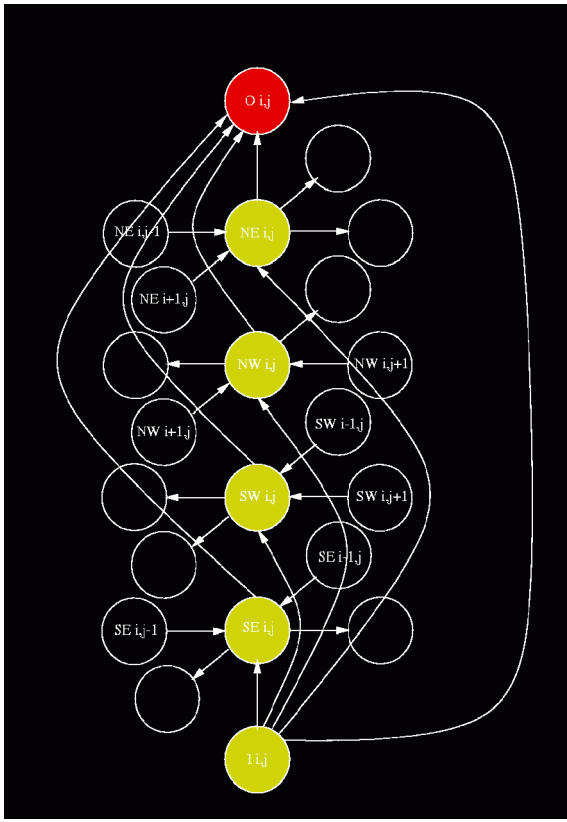
Editor: Michael I. Jordan

Abstract

We describe a general methodology for the design of large-scale recursive neural network architectures (DAG-RNNs) which comprises three fundamental steps: (1) representation of a given domain using suitable directed acyclic graphs (DAGs) to connect visible and hidden node variables; (2) parameterization of the relationship between each variable and its parent variables by feedforward neural networks; and (3) application of weight-sharing within appropriate subsets of DAG connections to capture stationarity and control model complexity. Here we use these principles to derive several *specific* classes of DAG-RNN architectures based on lattices, trees, and other structured graphs. These architectures can process a wide range of data structures with variable sizes and dimensions. While the overall resulting models remain probabilistic, the internal deterministic dynamics allows efficient propagation of information, as well as training by gradient descent, in order to tackle large-scale problems. These methods are used here to derive state-of-the-art predictors for protein structural features such as secondary structure (1D) and both fine- and coarse-grained

2D RNNs

$$\begin{cases} O_{ij} = \mathcal{N}_O(I_{ij}, H^{NW}_{i,j}, H^{NE}_{i,j}, H^{SW}_{i,j}, H^{SE}_{i,j,}) \\ H^{NE}_{i,j} = \mathcal{N}_{NE}(I_{i,j}, H^{NE}_{i-1,j}, H^{NE}_{i,j-1}) \\ H^{NW}_{i,j} = \mathcal{N}_{NW}(I_{i,j}, H^{NW}_{i+1,j}, H^{NW}_{i,j-1}) \\ H^{SW}_{i,j} = \mathcal{N}_{SW}(I_{i,j}, H^{SW}_{i+1,j}, H^{SW}_{i,j+1}) \\ H^{SE}_{i,j} = \mathcal{N}_{SE}(I_{i,j}, H^{SE}_{i-1,j}, H^{SE}_{i,j+1}) \end{cases}$$



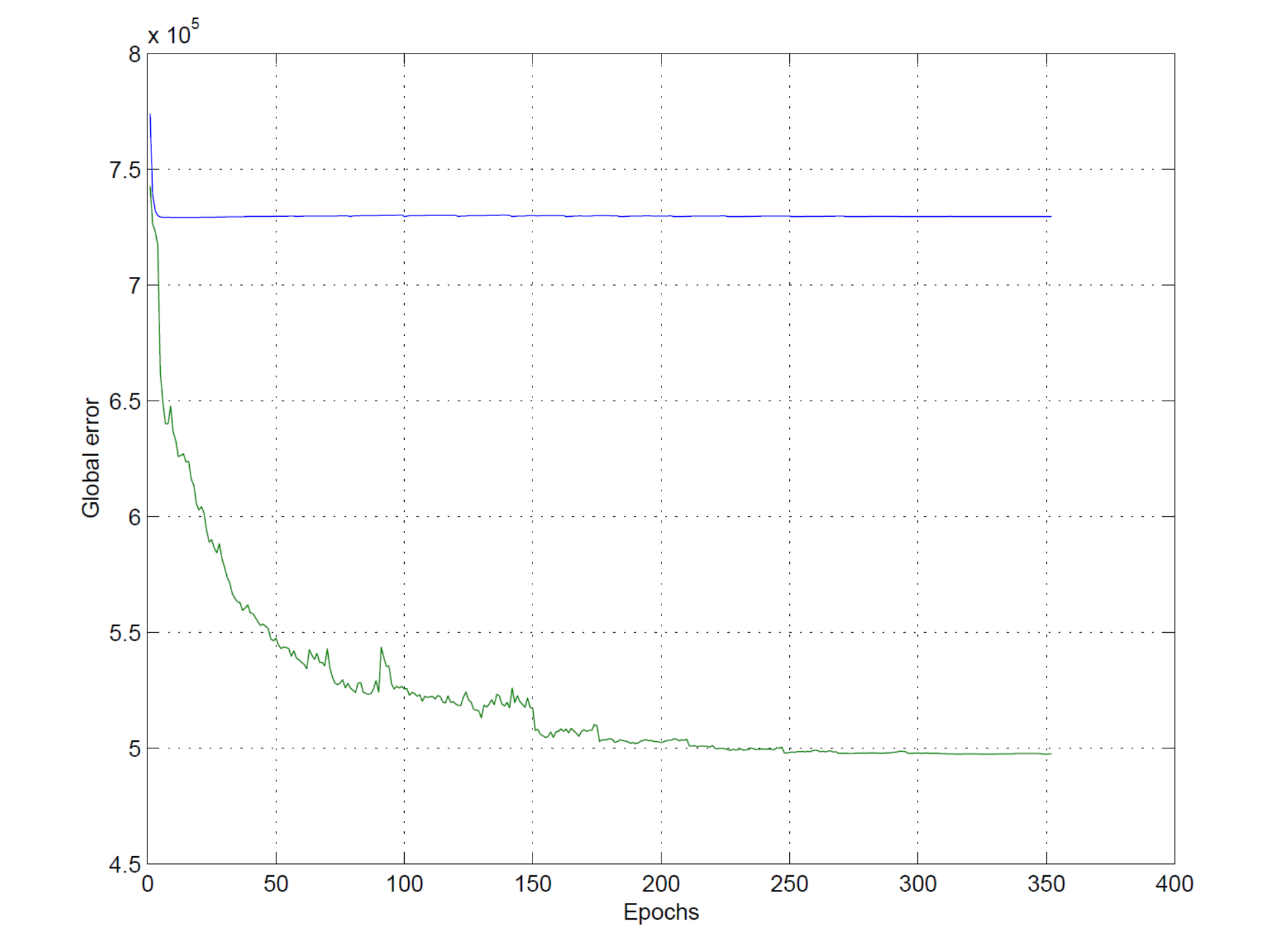
Pollastri & Baldi 2002, *Bioinformatics*
Baldi & Pollastri 2003, *JMLR*

4.1.3 Learning and Initialization

Training is implemented on-line, by adjusting all the weights after the complete presentation of each protein. As shown in Figure 6, plain gradient descent on the error function (relative entropy for contacts or least mean square for distances) seems unable to escape the large initial flat plateau associated with this difficult problem. This effect remains even when multiple random starting points are used. After experimentation with several variants, we use a modified form of gradient descent where the update Δw_{ij} for a weight w_{ij} is piecewise linear in three different ranges. In the case, for instance, of a positive backpropagated gradient δw_{ij}

$$\Delta w_{ij} = \begin{cases} \eta \times 0.1 & : \text{ if } \delta w_{ij} < 0.1 \\ \eta \times \delta w_{ij} & : \text{ if } 0.1 < \delta w_{ij} < 1.0 \\ \eta & : \text{ if } 1.0 < \delta w_{ij} \end{cases}$$

where η is the learning rate, and similarly for negative gradients with the proper sign changes. Figure 6 shows how this heuristic approach is an effective solution to the large-plateau problem. The learning rate η is set to 0.1 divided by the number of protein examples. Prior to learning,


x 10^5
8
7.5
7
6.5
6
5.5
5
4.5
Global error
0
50
100
150
200
250
300
350
400
Epochs

Deep learning solutions

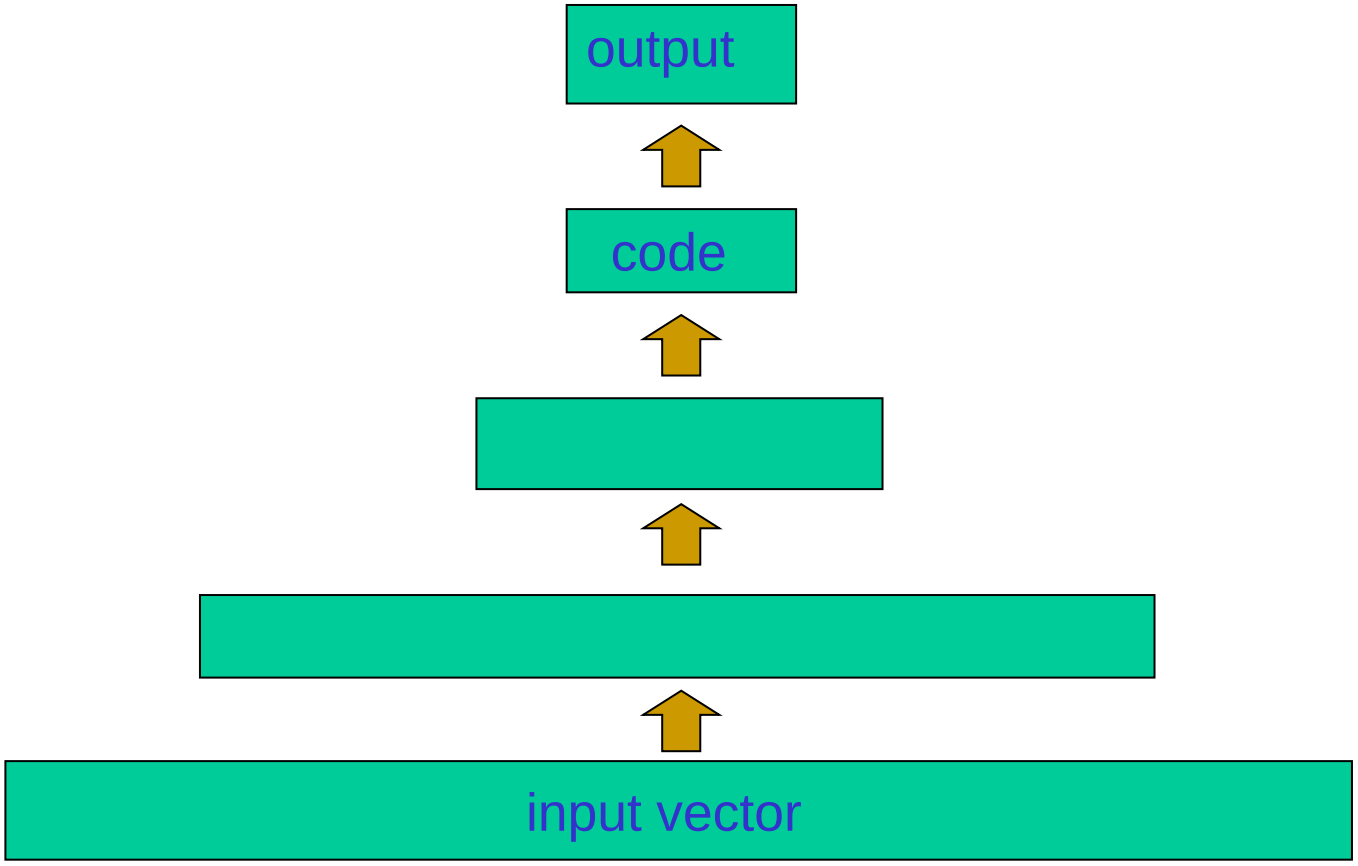
- **These are more principled solutions that were proposed starting circa 2006.**
- **Essentially they fall into two main categories:**
- **pre-train**
- **use artificial inner targets**

The rationale of pre-training

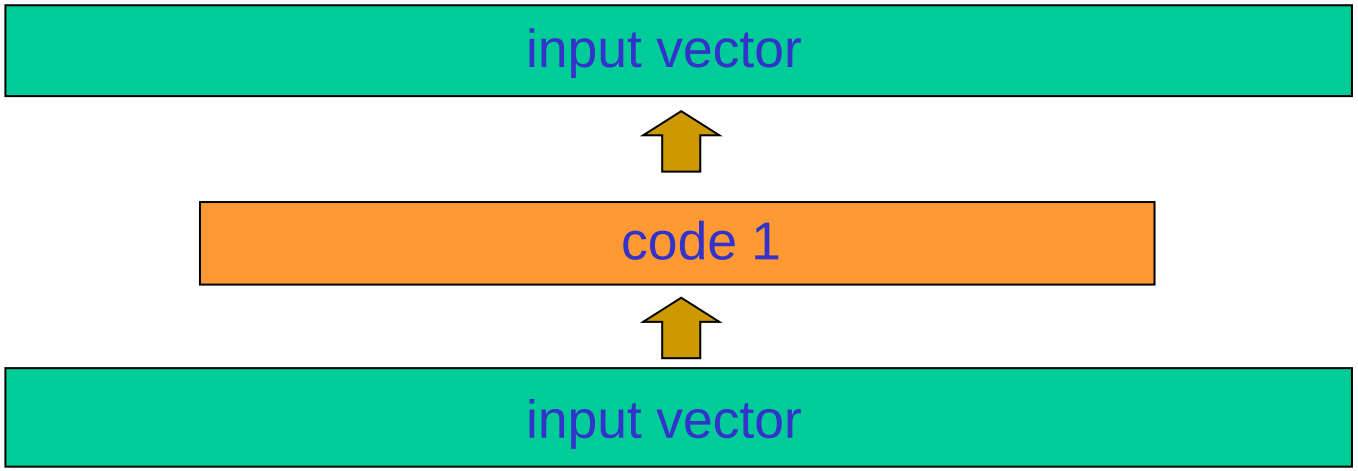
- **Stack deep networks layer by layer.**
- **Make sure your first hidden layer represents the input meaningfully before you add a second layer.**
- **Then make sure your second layer represents the first layer (thus, the input) meaningfully before you add a third layer.**
- **And so on…**

- **This can be done by auto-association**

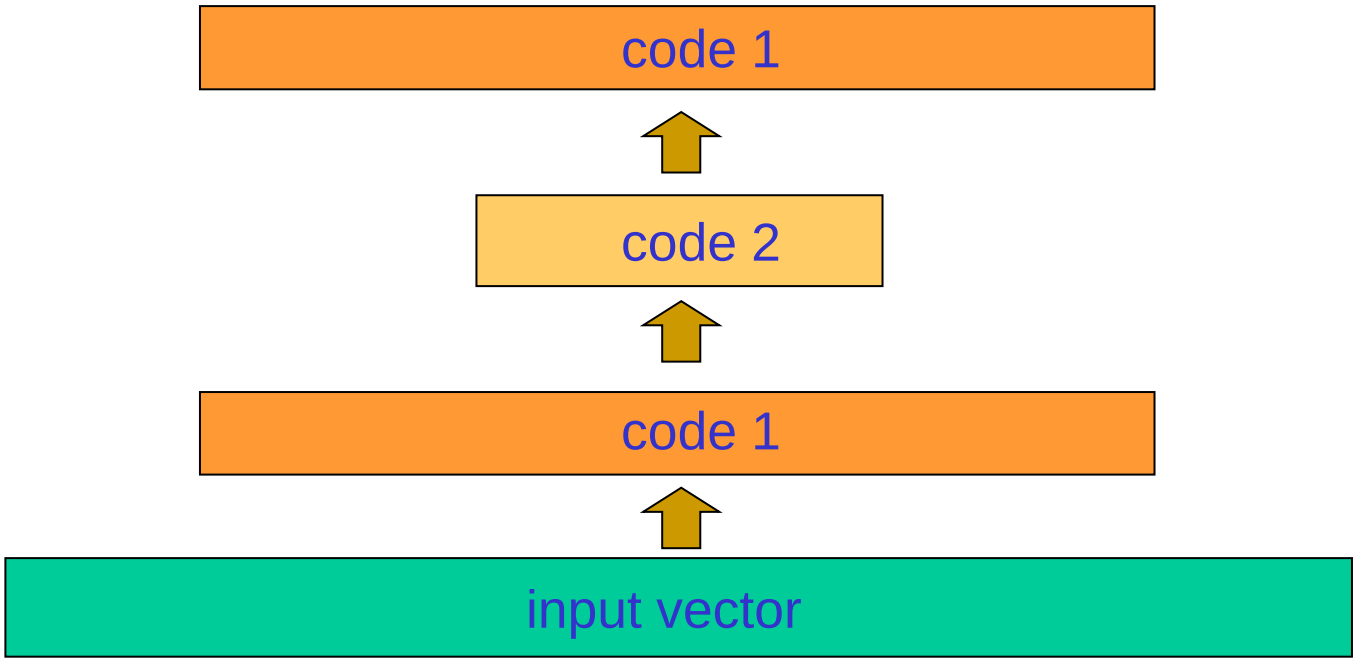
Pre-training



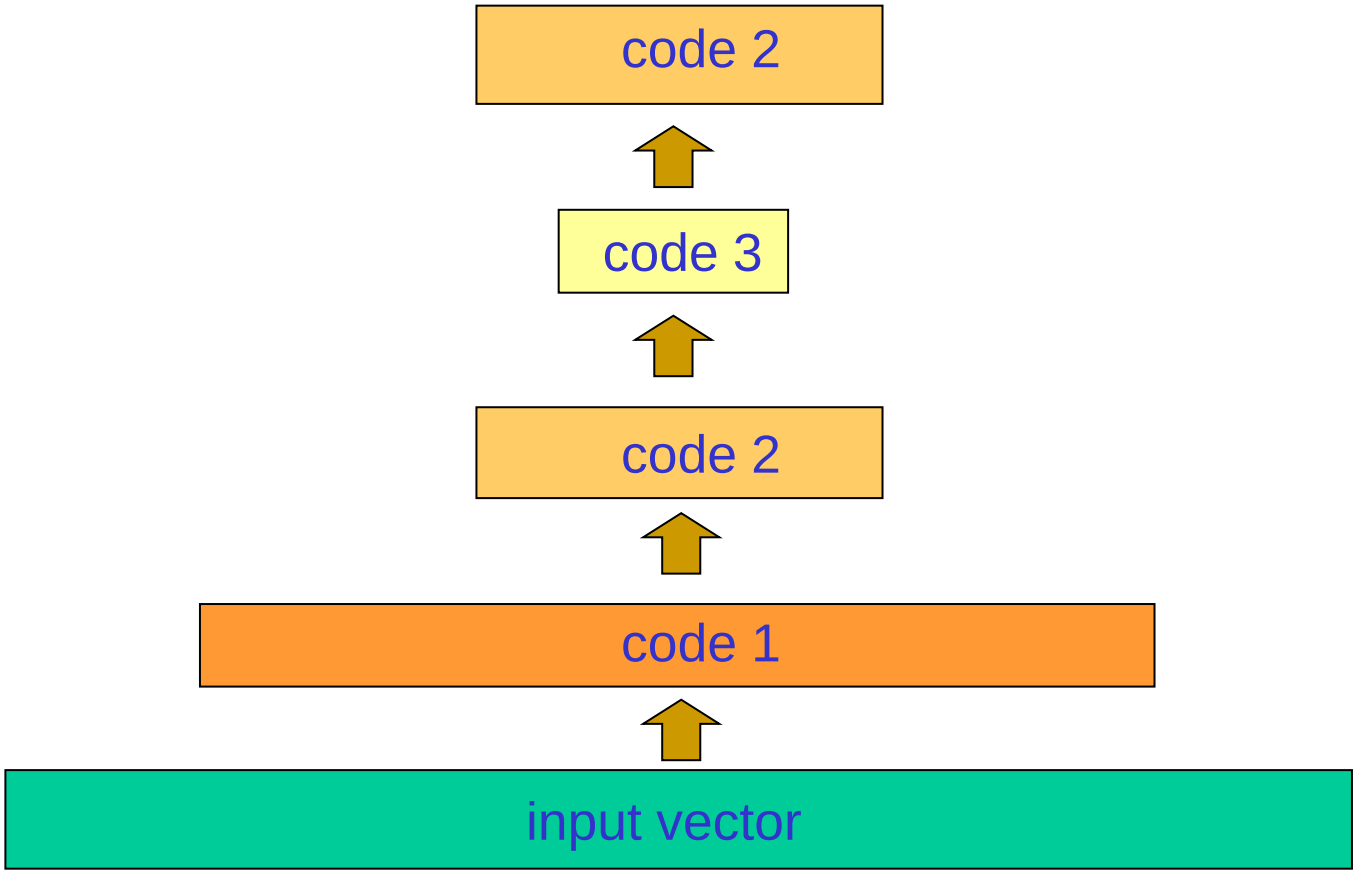
Pre-training (2)



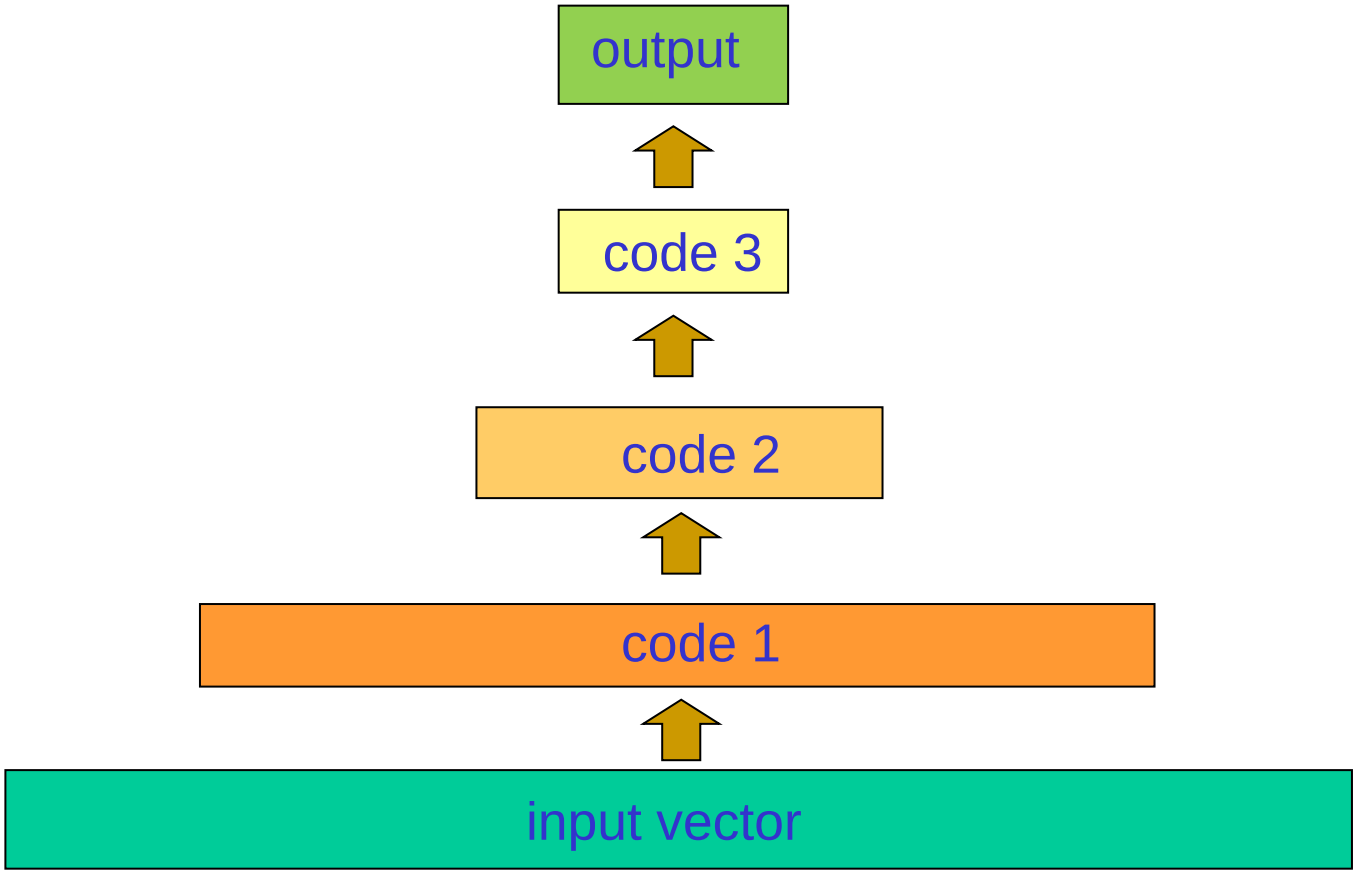
Pre-training (3)



Pre-training (4)

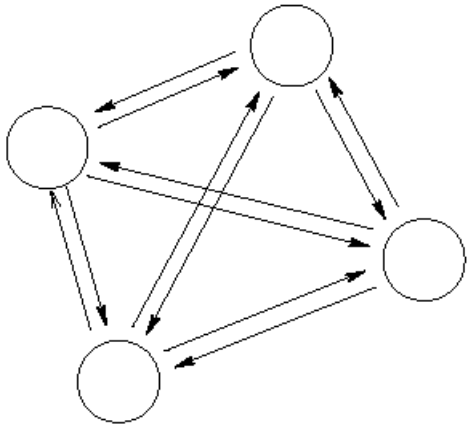


Pre-training (end)



Boltzmann Machines

- **Networks of stochastic units.**
- **Feedback networks: each units has connections to all other units except itself.**



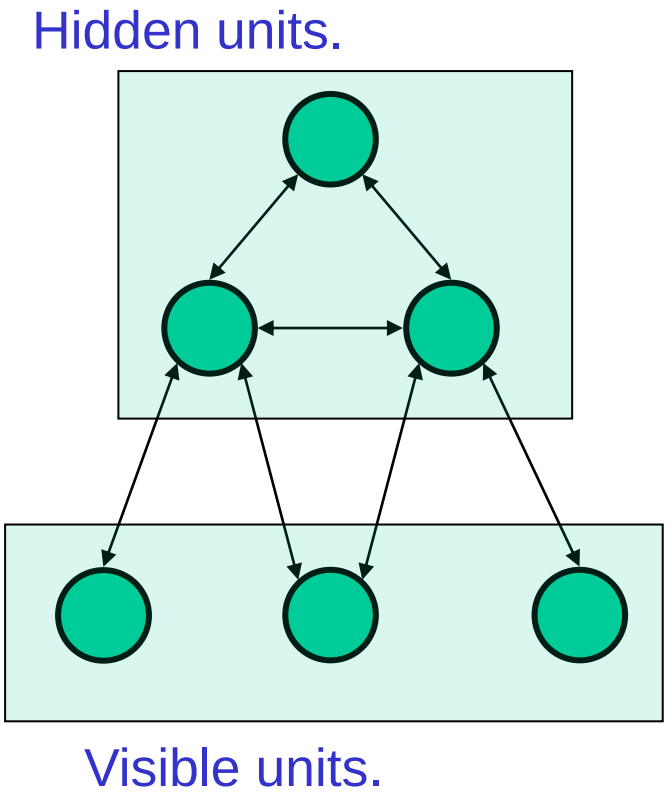
Yes, but what the heck are they for?

- **The idea is training this thing so that we place minima where we want them..**

- **We have a bunch of examples, we tweak the weight matrix by some rule, and at the end of our tweaking the machine will have a penchant for falling into the examples when we let it run.**

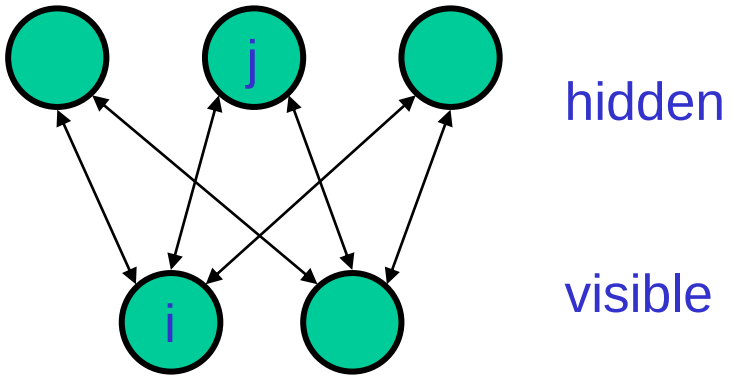
Hidden units for Boltzmann machines

- **Instead of using the net just to store memories, use it to construct *interpretations* of the input.**
  - **The input is represented by the visible units.**
  - **The interpretation is represented by the states of the hidden units.**



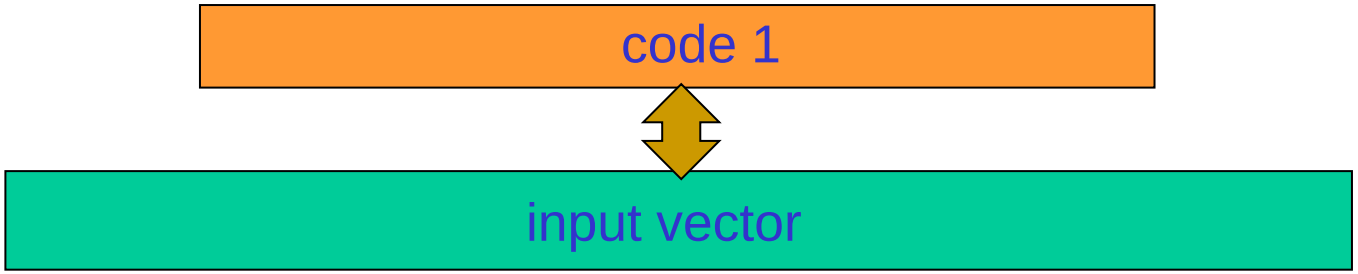
Restricted Boltzmann Machines (RBM)

- **We restrict the connectivity to make inference and learning easier.**
  - **Only one layer of hidden units.**
  - **No connections between hidden units.**

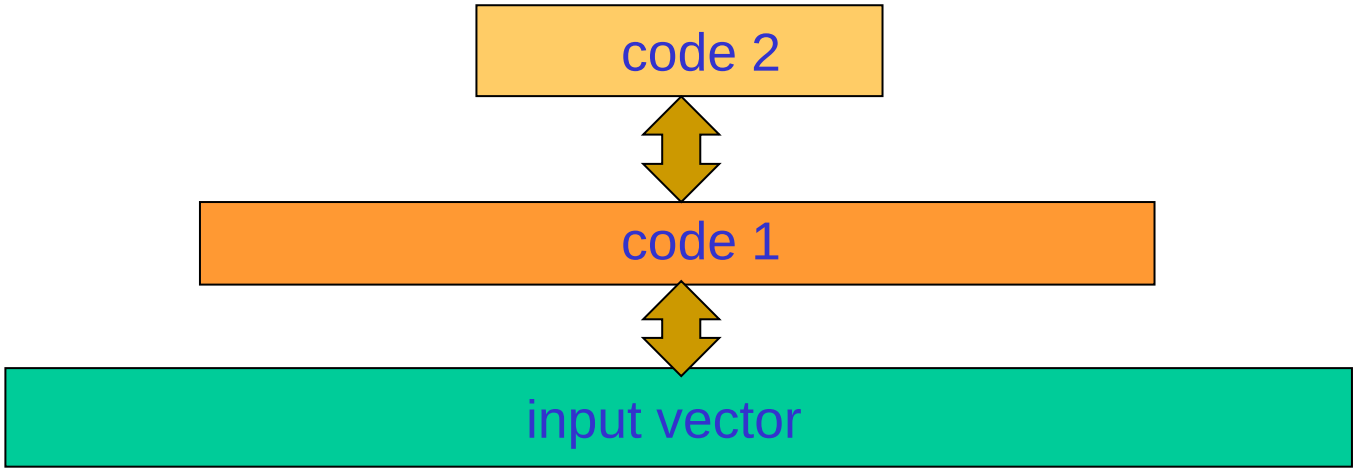


- **If it works, the hidden units are a perfect code for the examples!**

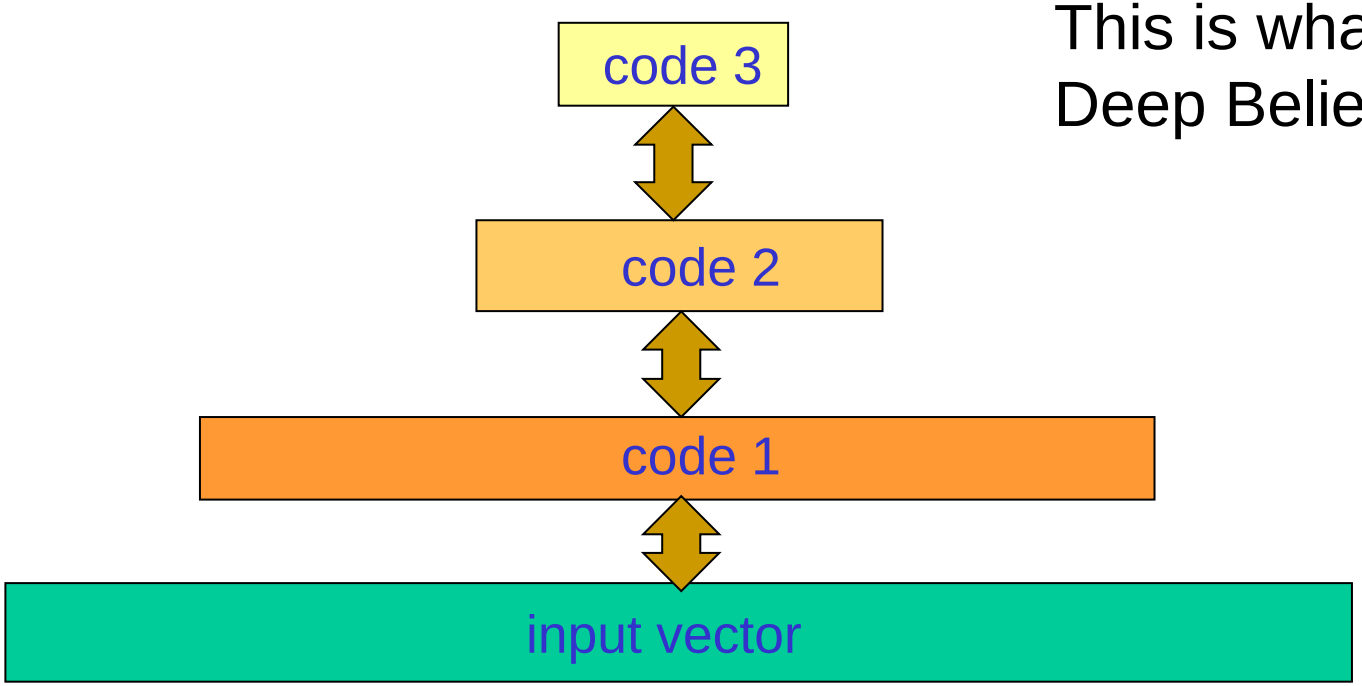
Alternative pre-training: Restricted Boltzmann Machines



RBM pre-training (2)

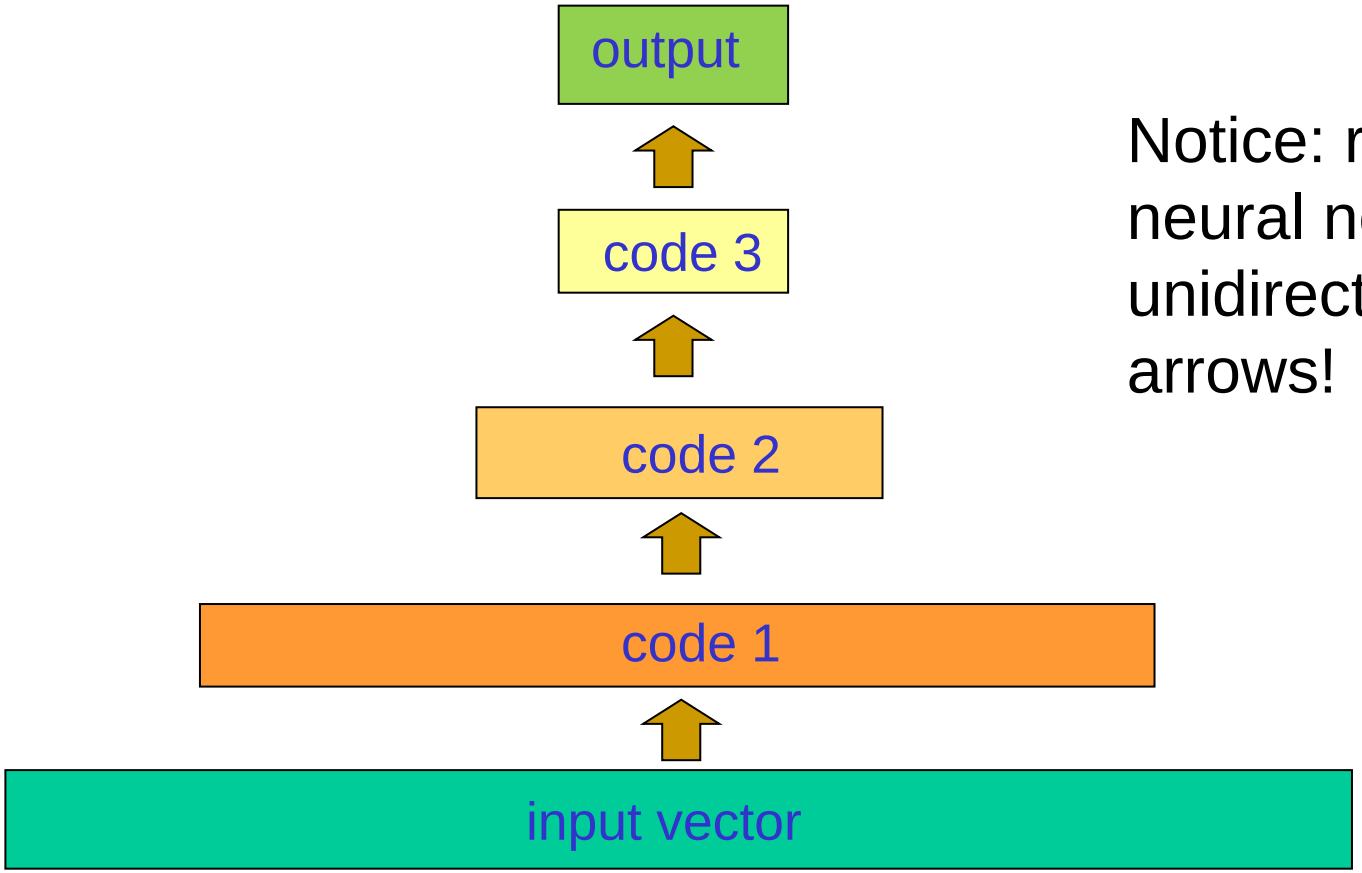


RBM pre-training (3)



This is what they call a Deep Belief Network

RMB (end)



Notice: recast in neural net fashion: unidirectional arrows!