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 <u>6-7%</u>
- a connectionist model to build and play with on some sets, write a report: <u>30%</u>
- Final Exam in the RDS (<u>50%</u>)

Backpropagation as message passing

- **Backpropagation is a schedule for computing weight updates (according to gradient descent) in *layered* networks of neurons of any depth.**
- **Any layer can be seen as an independent processor passing messages forward and backwards.**

Forward, backwards

- Forward: digest the input through the weights and produce an output.
- Backwards: digest deltas from the layer above, generate new deltas and pass them to the layer below.

- During backwards weight updates are also computed.

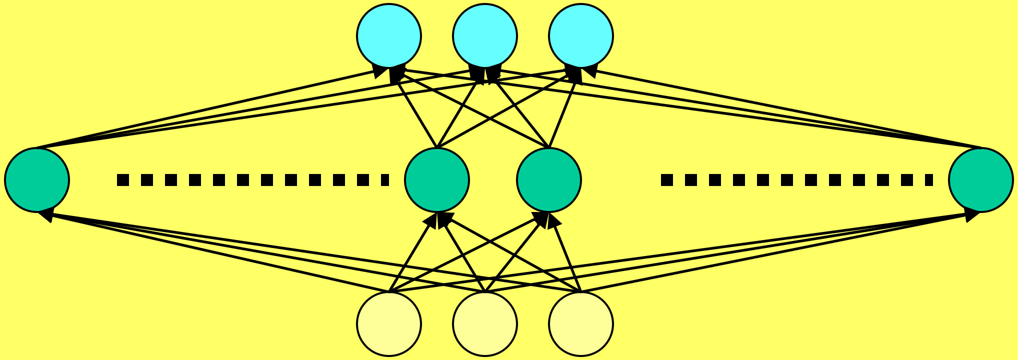
- A layer doesn't need to know anything about the network topology to do this. Excellent object.

Backpropagation

- Works on any Direct Acyclic Graph of continuous units: no binary-threshold (can't compute f'()).
- Loops acceptable only with time delays, we'll see this later.
- Very efficient:
  - $O(|w|)$
  - Large networks possible ($\sim 10^4$-10^6 weights reported in many real world applications)

Expressive power

- **Shortly:**
  - **A single hidden-layer network can approximate every input-output mapping (provided enough units in the hidden layer)**



VC dimension: Single and Multi-Layer Perceptrons

- **SLP performs linear separation. If there are n inputs**
  - **VC(SLP) = n+1.**
- **MLP is more powerful:**
  - **VC(MLP) = 2(n+1) M (1+logM)**

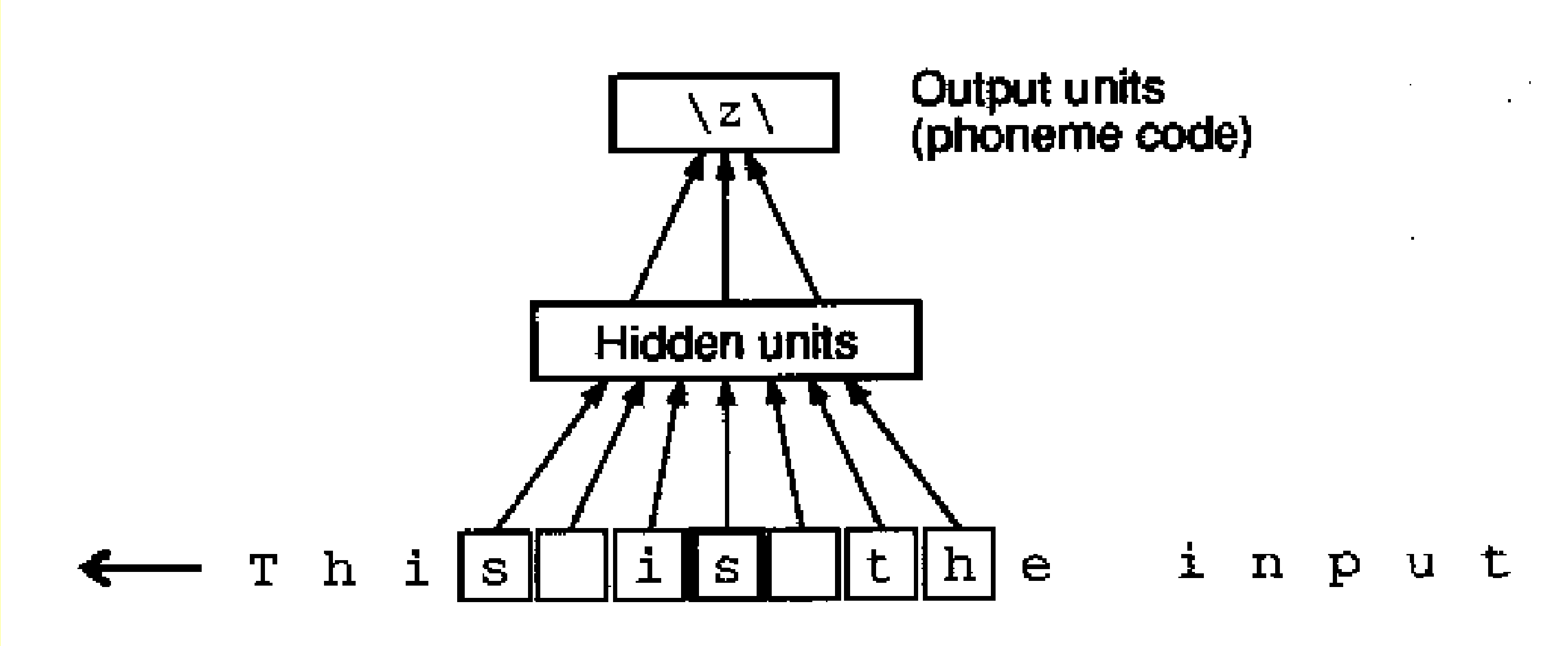
MLP applications: matching words and sounds

- Sejnowski and Rosenberg, "NETtalk, a parallel network that learns to read aloud", Cognitive Science, 14, 179-211 (1986)

- Teaching an MLP how to pronounce English by backprop.
- The network was given a stream of words, with the corresponding phonemes.
- Once the network had learned, it was possible to make it read.

The network

- 203x80x26 network
- input is sliding sequence of 7 characters
- 80 hidden units



MLP applications: protein secondary structure prediction

- **Proteins are strings:**

FEFHGYARSGVIMNDSGASTKS

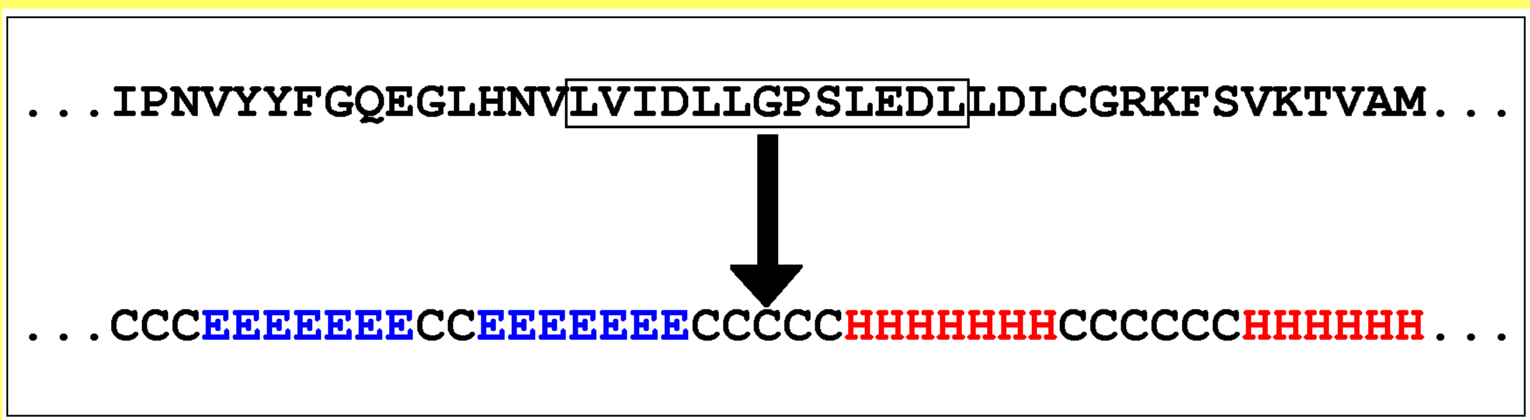
GAYITPAGETGGAIGRLGNQAD

TYVEMNLEHKQTLDNG

- **Structures too:**

by NETtalk

- Qian and Sejnowski 1988.
- They used NETtalk to predict it
- Sliding window, stacked networks.



MLP applications: handwritten digit recognition

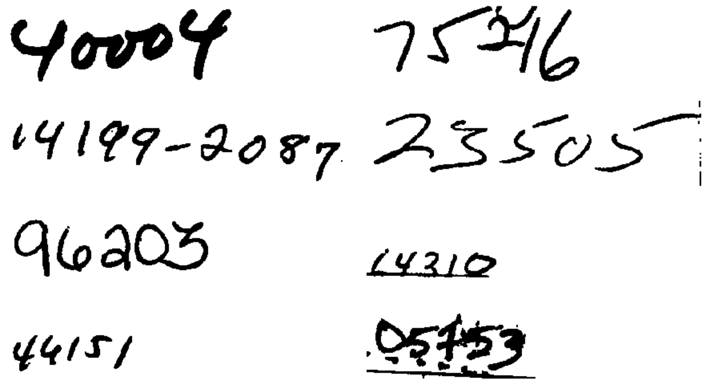
- "Hand-written digit recognition with a back-propagation network", Le Cun et al. 1990

- Multi-layer perceptron applied to handwritten digits.
- Relatively little non-connectionist preprocessing: digits split, centred, normalised.

The sets

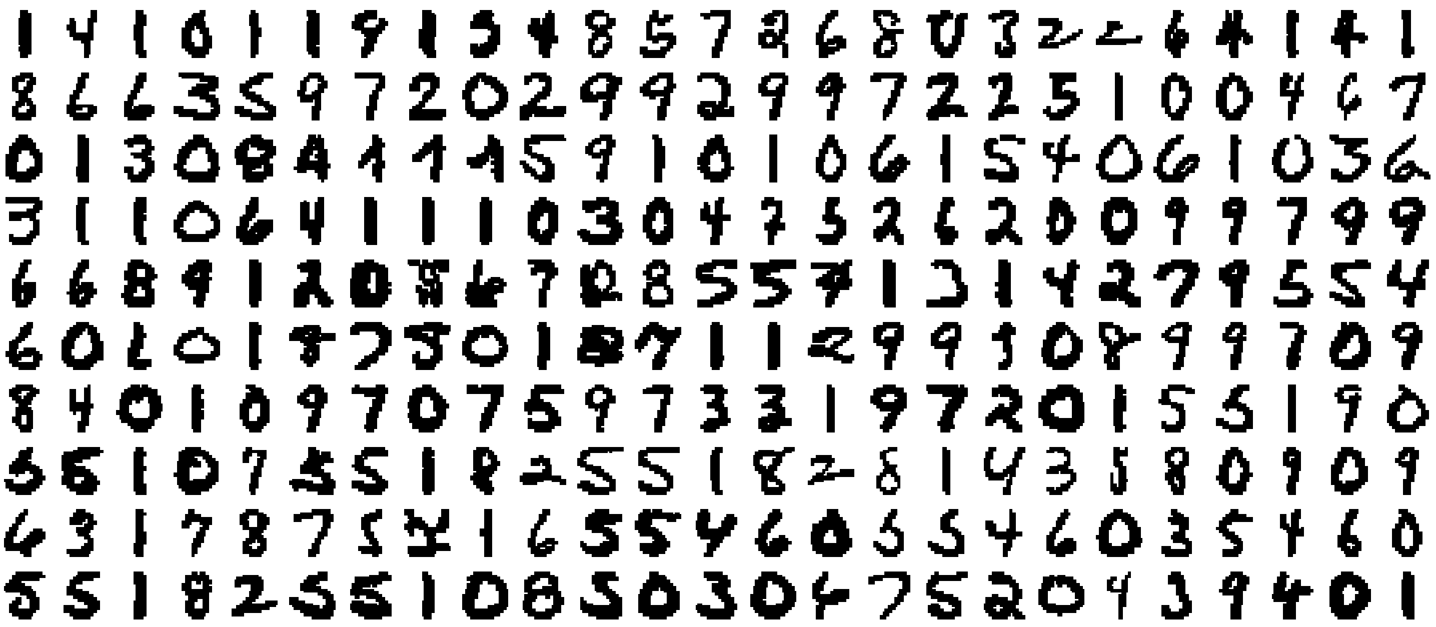
- **9298 digits from letters passing through the Buffalo office of the US PS + 3349 printed digits from 35 fonts.**

- **Training set: 7291+2549**
- **Test set: 2007+700**



preprocessing

- **Enough to obtain digits in this format, each one is a 16x16 greyscale image:**



The network

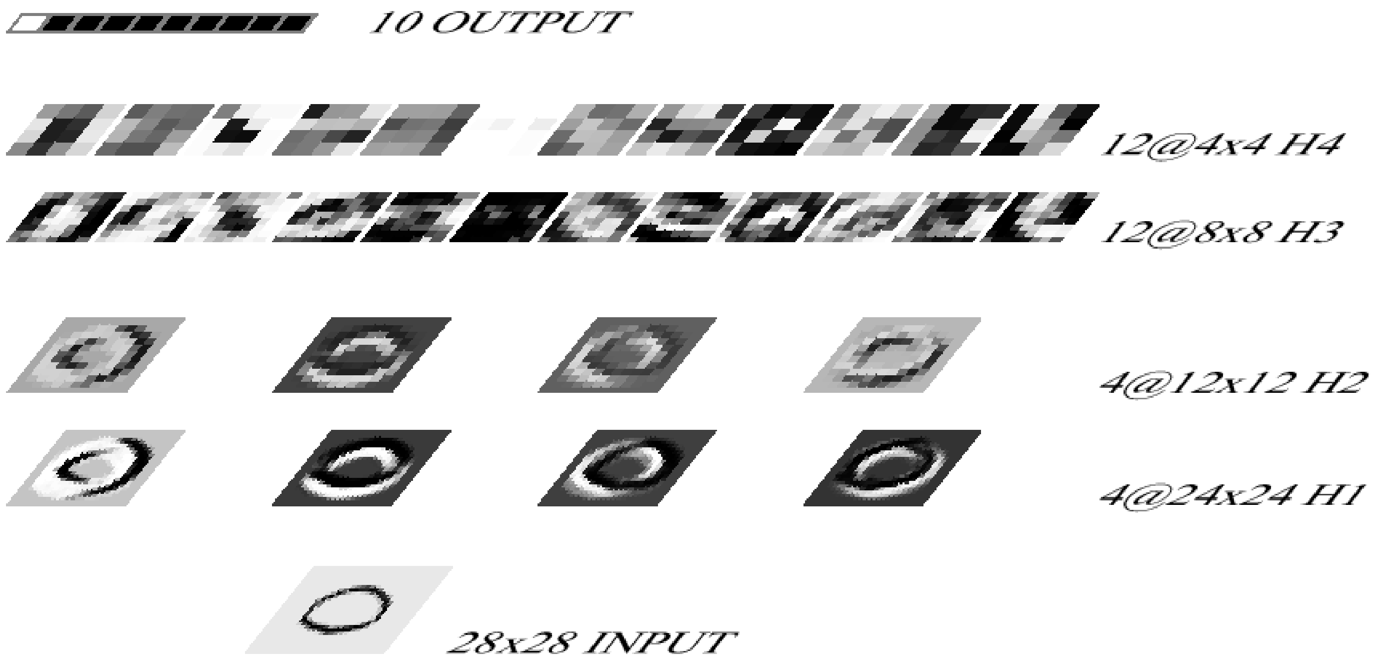
- In theory one could simply feed the image into a Multi-layer perceptron with M hidden units.
- The input would be an array of 16x16=256 real values, the output an array of ten values, one for each class (0,1,..9).
- This means (256+1)M+(M+1)10=267M+10 weights. If M=50 -> 13360 weights, fairly large for 10k examples (and difficult to train in 1990).

The network

- Le Cun et al. design a network with:
- partial connectivity, i.e. not all the units in layer i are connected to all the units in layer i+1.
- weight sharing, i.e. different parts of the networks are forced to use the same weights.

deep network

- **4 hidden layers:**



Feature maps

- **Hidden layers 1 and 3 implement *feature maps*.**
- **<u>Layer 1</u> Input is the 16x16 image, with borders added for technical reasons -> 28x28. Output is composed by 4 maps of 24x24 units. This is really implemented with 4 neurons, each taking 5x5 inputs, each replicated in each possible position on the input map.**
- **This sounds complicated but is fairly easy: instead of a (28x28)->(24x24x4) full connectivity only 5x5 inputs are connected to each output unit. Not only, but there are just 4 neurons/sets of weights (*weight sharing*).**
- **So, a (5x5)->4 full connectivity that sweeps the whole input. Only 104 weights including biases!**

Averaging/subsampling layer

- Hidden layers 2 and 4 implement averaging/subsampling stages.
- <u>Layer 2</u> : 24x24x4 -> 12x12x4
- This is performed using 4 units, each one doing a 2x2->1 mapping. Weights are constrained to be all the same.

layers 3, 4 and 5

- **Layer 3 : more feature maps. 12 8x8 maps. Each map is composed by a neuron (always the same) mapping a 5x5 area into a unit.**
- **Layer 4: Same as layer 2. (8x8x12) -> (4x4x12)**
- **Layer 5: 10 output units fully connected to layer 4. This is where most weights are.**

overall

- 5 layers, position invariance encoded in the architecture, a lot of weights shared.
- ~100k connections -> 2k independent parameters. every weight is shared on average by 50 connections.
- Training complexity is still o(100k) though.

Training the network

- **Training is by gradient descent, using backpropagation.**
- **For each copy j of a shared weight there will be a Δw_j. They are simply added together.**

Results

- After 30 epochs the error on the training set was 1.1% and the squared error 0.017.
- On the test set: 3.4% and 0.024

- To get 1% error: 5.7% rejection (9% on just handwritten)
- A lot of these were actually caused by preprocessing. Some of those that weren't, were ambiguous even to humans.