Weightless Neural Networks

The standard MLP type network has various drawbacks, one of which is the time it takes to learn.

An alternative type of network, almost unique to the UK, is the Weightless Neural Network – these are also called n-tuple networks or RAM based networks.

These have a very different model of a neuron – a memory

These neurons have no weights – hence 'weightless' nets

Learning is also different and much simpler

Being based on memories, are implementable in hardware – WISARD was the first commercial neural network system

We will investigate the standard system and generalise

The n-tuple Neuron

Weightless networks arose out of Bledsoe and Browning's work (1959) on n-tuples

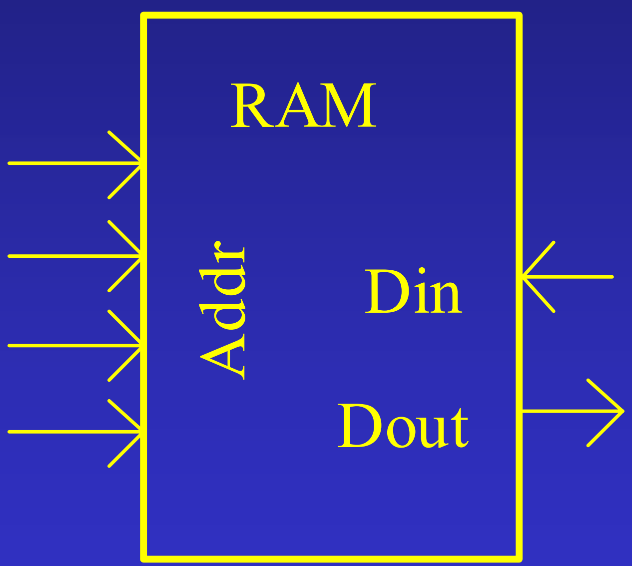
n-tuples are n bits (binary digits) sampled from input data.

The n-tuple neuron is basically a standard RAM.

A n-tuple is put on the input address lines of the RAM.

To learn, a value is written into the specified address.

To analyse, read from the addressed location.



Simple (Impractical) Use

Suppose we want to recognise images of faces

Initially, clear all locations in RAM

Take a digitised picture

- Connect each pixel to an input of the RAM
- This addresses one location – write a ‘1’ : learns image

Other images could be learnt also.

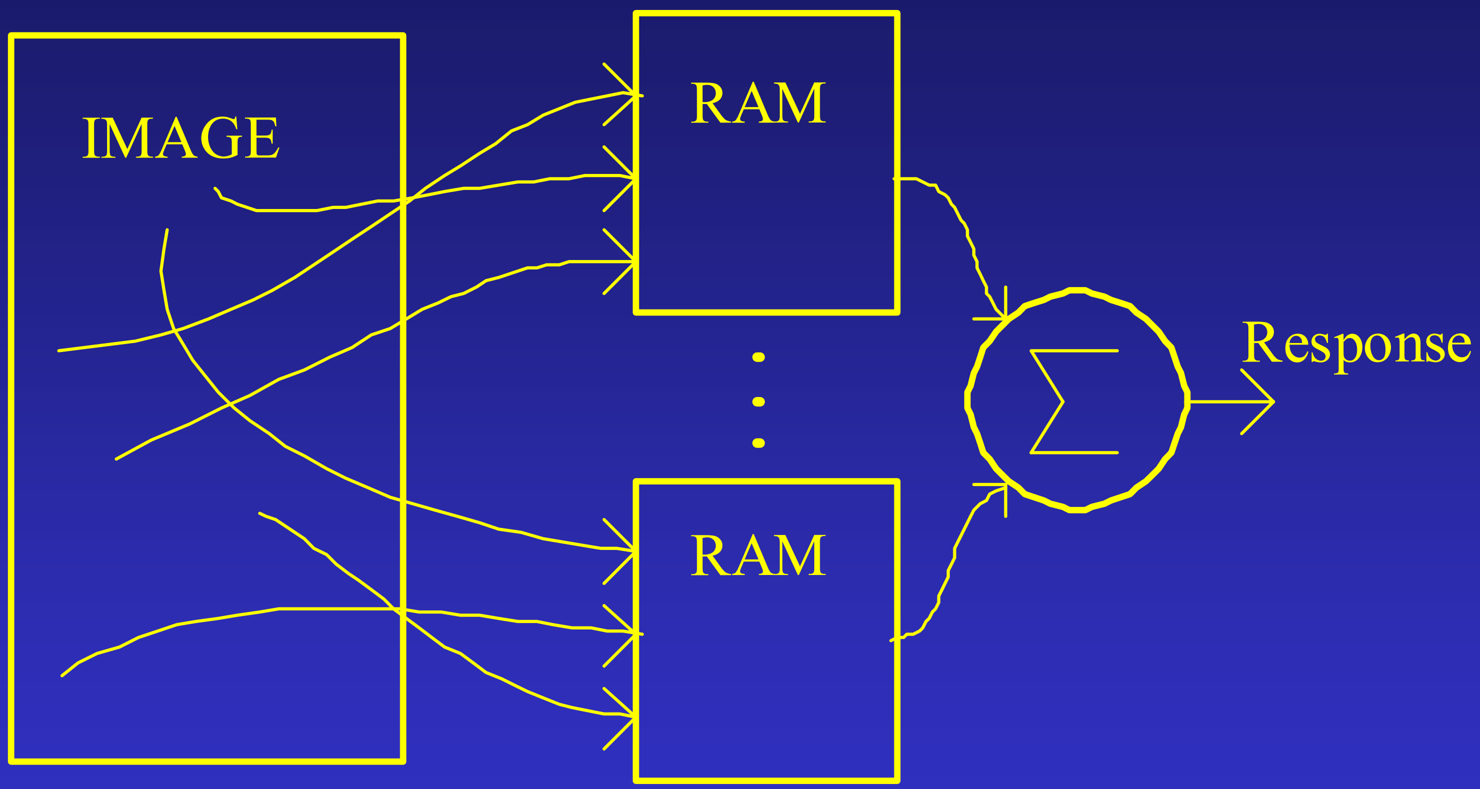
Present new image: recognised if ‘1’ at addressed location

For, say, a 256*256 binary image, we would need a RAM with 2^{16} ie 65536 address lines = 2^{65536} locations

Solution – have many such RAMs, address for each found by sampling different bits from input …

Tuples best sampled randomly but consistently

Practical Configuration



Learn: write '1' into approp location in each RAM

Analyse, count how many RAM neurons 'fire' (have '1')

Class Discriminators

A group of RAM Neurons is called a Class Discriminator

Typically many (similar) examples of one pattern class (eg various images on one person's face) are taught to system

Then the Disciminator is able to recognise

an image it has been taught AND

an image similar to, but not identical to one already taught.

e.g. RAM 1 might recognise a tuple from image 5

RAM 2 might recognise a tuple from image 8, etc.

An image is 'recognised' if percentage of RAMs which output 1 is greater than some threshold (say 90%) but depends on amount of noise, and how accurate system needed.

Multiple Discriminators

Note can ‘teach’ different classes in one discriminator
- eg images of two faces can be taught

Then system can say if it recognises an input
- but it will not be able to say which face it is

To discriminate between classes, have one discriminator for each
- Teach each class into its own discriminator ONLY
- When analyse, count firings of ALL discriminators
- Image belongs to discriminator with largest number of fires

Memory Requirements

In a Discriminator employing:

Input vector of size ***R***.

k-times over-sampling.

Using a tuple size of ***n***.

Memory of each RAM is defined as Z, where $\mathbf{Z} = 2^{\mathbf{n}}$

M RAMs are needed.

Where $\boldsymbol{M = k \times R / n}$.

The memory requirement of the entire network is thus simply:

MEMORY = M (rams) x Z (bits per ram)

e.g. 256*256 image, 8 bit tuples, no over-sampling ($k = 1$)

M = 1 * 256 * 256 / 8 = **8192** and **Z** = 2^8 = **256**

MEMORY = 8192 * 256 = 2097152 bits

Generalisation in a n-Tuple Network

How many input patterns will result in maximum output of four RAMs firing?

There are four training images:

TRAINING set size: T = [4]

There are eight patterns that will result in maximum FOUR RAMs firing.

The four training images plus:

[2+3], [2+4], [3+4], [2+3+4]

Hence GENERALISATION set is of size [8]

1	2	1	3
4	3	2	2
2	1	3	3
4	4	4	1

Tuple Maps for 4 RAMS

Image 1

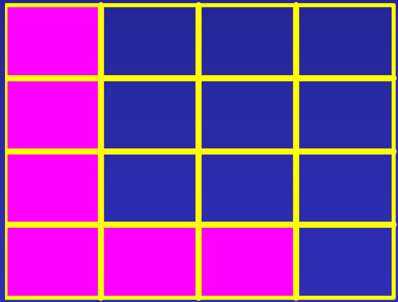


Image 2

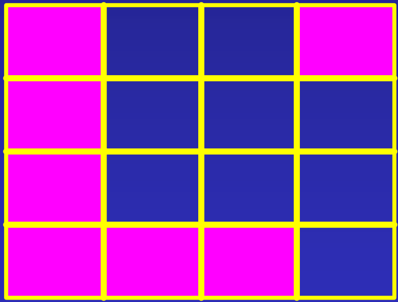


Image 3

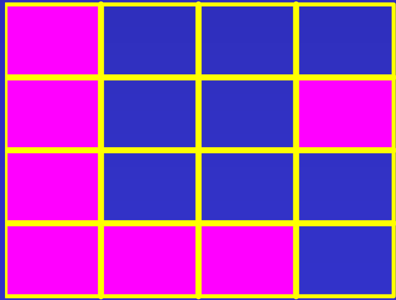
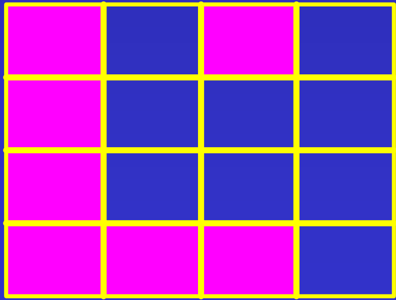


Image 4



Algorithm

First, set all locations in each RAM to 0

Learn (one image)

- For all RAMS
  - Sample n-bits (to form a tuple)
    - Write ‘1’ in location with address tuple in given RAM

Analyse (one image)

- NumFire := 0;
- For all RAMs
  - Sample n-bits (to form a tuple)
  - If location with address tuple in RAM = 1,
- INC(NumFire)
- IF NumFire > Threshold, Image is Recognised

Let us show some experiments on character data

MATLAB m-file

```
function ans = wnn_simple (data, tuplemap, tsize, discrim)
% Learn data into discriminator or see if one 'recognises' data
% To learn invoke by
%   DISCRIM = WNN_SIMPLE(DATA, TUPLEMAP, TSIZE)
%       DATA is cell array of data (each typically 8*8 chars)
%       TUPLEMAP specifies the order in which it is sampled
%       TSIZE is tuplesize
%       This routine returns in DISCRIM the taught image(s)
% or analyse by
%   Ct = WNN_SIMPLE(DATA, TUPLEMAP, TSIZE, DISCRIM)
%       DATA,  TUPLEMAP, TSIZE as above
%       DISCRIM is the taught network
%       Routine returns the percentage of firing neurons
% Dr Richard Mitchell 29.7.03
```


Cybernetics

Tuple Mapping Optimisation Method

From Bishop, Crowe, Minchinton & Mitchell, 1990 (IEE colloq)
Evolutionary Learning to Optimise Mapping in n-tuple networks
Aim : choose mapping to maximise discrimination of classes.
So select at random some mapping
Get measure of discrimination
REPEAT
    Select other mapping by mutation (changing) of bits in mapping
    Get measure of discrimination
        % learn classes A and B, how different number of 'fires'
    IF better THEN adopt this mapping and its measure
UNTIL done   *% NB mutation rate decays*

For Character Recog: much better discrim between c & e; i & l.

Handling Non-Binary Input Data

What if images comprise grey levels not just black white

Want to be able to cope with small changes in lighting

Could turn grey level into series of 0s and 1s (say 8 bits)

Then data to process is 256*256*8 bits

But – need more RAMs

a change of grey from 3 to 4 involves 3 bit changes

thus generalisation will be poor

One solution is to use gray code

0 to 7 is: 000 001 011 010 110 111 101 100

So changing 3 to 4 is 010 to 110 just one bit change

But too many RAMs needed still

Threshold and Thermometer Coding

Simple solution – choose a **Threshold**

- if gray value ≥ threshold, tuple bit = 1 else tuple bit = 0
- Image now in effect 256*256*1 so same number of RAMs
- But if lighting changes a little and many values near threshold there can be quite a large tuple change.

More advanced – multiple thresholds – or **Thermometer Code**

Replace (say) 0..255 by (say) 5 patterns

v < 50	v < 100	v<150	v<200	rest
0000	0001	0011	0111	1111

Image now 256*256*5, so need 5 times as many RAMs

But system less susceptible to lighting changes

Minchinton Cells

These are simple processing elements placed between input data and the tuple forming elements

Easy to think of as being between input & RAM address inputs

Let I(x) be value at position x in input data I

Simplest cell I(x) > constant   This is thresholding

Type 1 cell $I(x_1) > I(x_2)$ Compares two random points

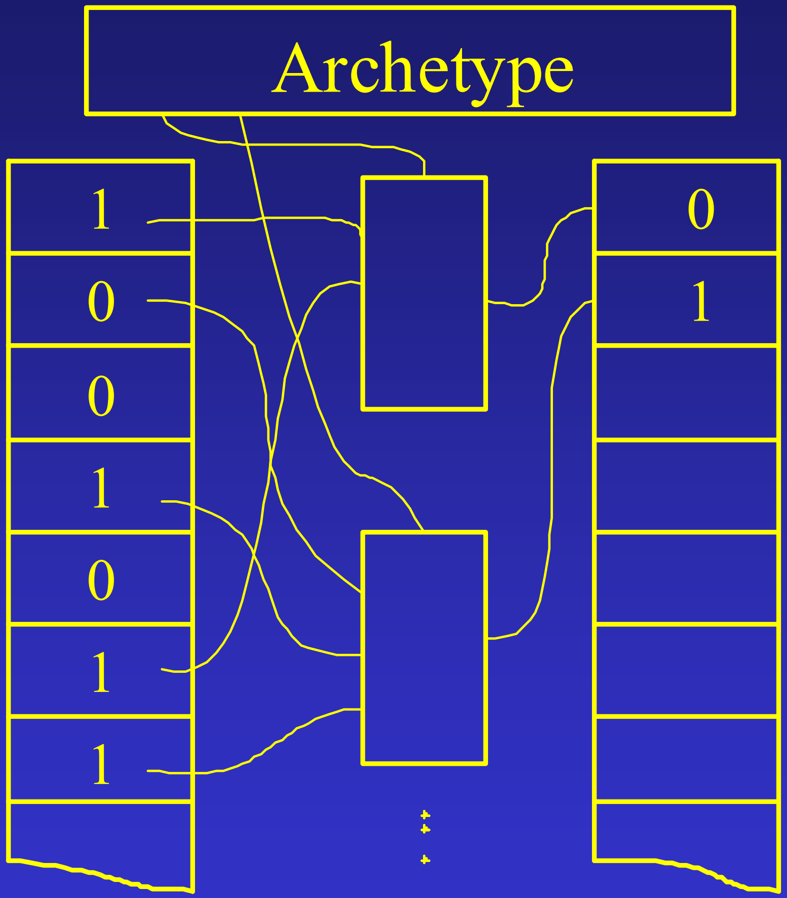
If lighting changes, for much of image, grey value increased by constant amount. Thus this difference unchanged. So type 1 cell makes system more tolerant of lighting changes.

Does not increase number of RAMs. Seems best method. See:

S Lauria, R.J.Mitchell: "*Pre-Processing Grey Level Data for Weightless Neural Networks*", Proc CESA '98, Tunisia, 671-675

Weightless Associative Networks

- It is possible to use n-tuple RAMs directly to map an input to an output vector.
- By over-sampling by **n** (tuple size) the output vector has the same size as the input.
- When learning into r^th^ RAM, store in RAM the value of rth pixel in 'archetype' image.
- When analyse, values output from RAMs should be archetype (or close to it).



Associative n-tuple network

Pattern Separation – Binary Images

D Aitken J.M.Bishop R.J.Mitchell S Pepper : *Pattern Separation in Digital Learning Nets*", Elec Lett, 25:11, pp: 685-686 (1989)

For storing archetypes of many classes in one discriminator

As one class may want 1 in a RAM, another class a 0.

So define RAMs have four states for each location

- GROUND – not learnt – can be overwritten to '0' or '1'
- State 0 – equivalent to a '0'
- State 1 – equivalent to a '1'
- CLASH – used where tried to override '1' with '0' or vv

Initially all RAMs are in GROUND state.

On analyse, if find GROUND or CLASH in rth neuron, best guess is to output rth value in INPUT

Put in Feedback Loop

Train network by presenting various images of each class, each time storing in network the archetype of the class.

For reading out

- Take input; pass to Associative Network; get output
- This output should be close to 'archetype'
  - Key point is output is closer to archetype than input
- So use output as input to Associative Network and analyse
- The new output should be closer still to the archetype
- Do again – after a few iterations, systems stabilises.

Has been extended to grey level data: states 0..n; as well as GROUND and CLASH, but basically operation same.

Other Types of Weightless Networks

Note, other types exist

P-RAMs and PLNs - Probabilistic Neurons

neuron stores the probability that it will fire.

These are used in feedback circuits

G-RAMs - Generalised Neurons (Igor Aleksander)

ADAM - a different form of associative network

- this is the work of Jim Austin at York.

- it is a two stage network

and more …

still today there is some research into these networks