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>)**

Assignment 1

- Read the first section of the following article by Marvin Minsky:
- http://web.media.mit.edu/~minsky/papers/SymbolicVs.Connectionist.html
- down to ".. we need more research on how to combine both types of ideas."

- Submit through BrightSpace a 250 word MAX summary by October 2nd at 23:59, any time zone of your choice (Baker Island?).

- 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.
- Make sure it's gone through!

Gradient descent and associators

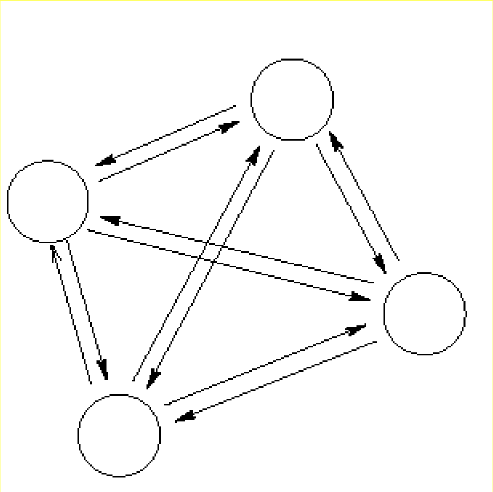
- Complexity of gradient computation is minimal: o(nm)
- BUT, it is unclear how many steps along the gradient need to be taken…

- Much better storage of examples than with one-shot learning, though typically there will be residual error.

$$\Delta w_{ji} = -\eta \sum_{p=1}^{P} (y_j^{(p)'} - y_j^{(p)}) x_i^{(p)} = \eta \sum_{p=1}^{P} (y_j^{(p)} - y_j^{(p)'}) x_i^{(p)}$$

Hopfield Nets

- Networks of <u>binary threshold</u> units.
- Feedback networks: each units has connections to all other units except itself.



The energy function

- **The global energy is the sum of many contributions. Each contribution depends on one connection weight and the binary states of two neurons:**

$$E = -\frac{1}{2}\sum_{i,j} w_{ij} y_i y_j - \frac{1}{2}\sum_i b_i y_i$$

- **The simple energy function makes it easy to compute how the state of one neuron affects the global energy (it is the activation of neuron!):**

$$E(y_i = -1) - E(y_i = 1) = \sum_j w_{ij} y_j + b_i$$

Storing memories (learning)

- If we want to store a set of memories:

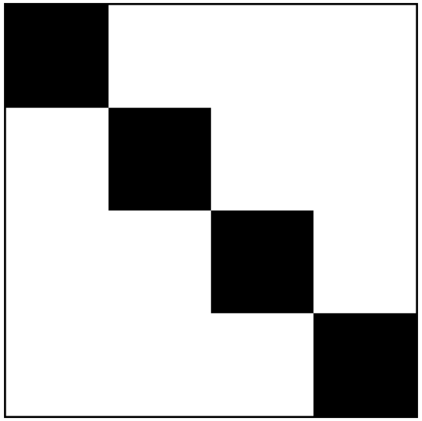
$$y^{(1)}, .., y^{(p)}, .., y^{(P)}$$
$$y^{(p)} = (y_1^{(p)}, .., y_m^{(p)})$$

- if the states are –1 and +1 then we can use the update rule:

$$\Delta w_{ji} = \eta \sum_p y_i^{(p)} y_j^{(p)}$$

Example

- Say we have 4 patterns of size 4:
- (1, -1, -1, -1)
- (-1, 1, -1, -1)
- (-1, -1, 1, -1)
- (-1, -1, -1, 1)



- We build the sigmoid Hopfield net based on the 4 patterns (Matlab nnet toolbox).
- Incidentally, here one-shot learning would not work: try.

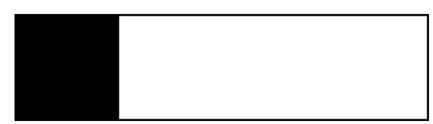
$$\Delta w_{ji} = \eta \sum_p y_i^{(p)} y_j^{(p)}$$

Example

- Let’s now start from some state Y for the neurons, and watch the network evolve.
- Y=(1 0 0 0)

Steps (1 update/neuron):

1: (0.4999 -0.6620 -0.6620 -0.6620)
2: (0.5022 -0.8476 -0.8476 -0.8476)
3: (0.6351 -0.9332 -0.9332 -0.9332)
4: (0.8186 -1.0000 -1.0000 -1.0000)
5: (1 -1 -1 -1) **converged to pattern 1!**



Example (2)

- **Different starting point:**
- **Y=(0 0 0 0)**

Steps:

1: (-0.4273 -0.4273 -0.4273 -0.4273)

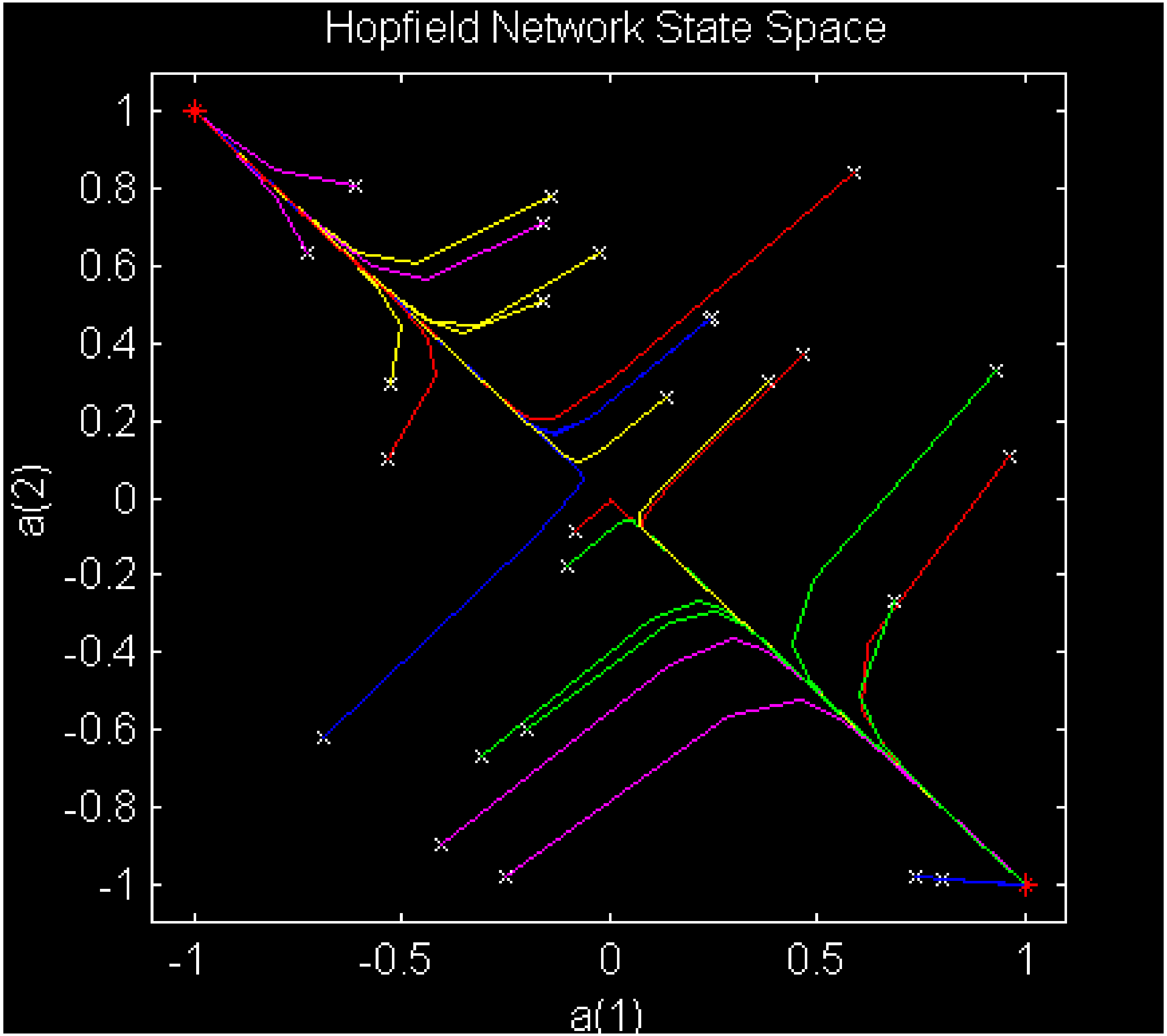
2: (-0.5226 -0.5226 -0.5226 -0.5226)

3: (-0.5439 -0.5439 -0.5439 -0.5439)

4: (-0.5486 -0.5486 -0.5486 -0.5486)

..

7: (-0.55 -0.55 -0.55 -0.55) **stuck in the middle!**



Hopfield Network State Space
a(2)
a(1)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

Example (3)

- Let's now try train a Hopfield net on slightly nastier vectors (the previous ones were orthogonal):

(1.0000   -1.0000   -1.0000   0.3000)
(0.1000   1.0000   -1.0000   -0.1000)
(-1.0000   -1.0000   1.0000   -0.5000)
(-1.0000   -1.0000   -1.0000   1.0000)

Example (3)

- Let's start from some state Y for the neurons, and watch the network evolve.
- Y=(1 0 0 0)

Steps (1 update/neuron):

1:	(0.9957	-0.3693	-0.3693	-0.0028)	
5:	(1.0000	-0.5332	-0.5332	-0.0040)	
50:	(1.0000	-0.5348	-0.5348	-0.0040)	odd plateau
170:	(1.0000	-0.5345	-0.5350	-0.0040)	
5000:	(1.0000	0.2032	-1.0000	1.0000)	spurious min!

From Hopfield nets to Boltzmann machine

- **Hopfield nets (attempt to) minimise an energy function:**

$$E(y) = -\frac{1}{2}\sum_{i,j} w_{ij} y_i y_j = -\frac{1}{2}\mathbf{y}^{\mathbf{T}}\mathbf{W}\mathbf{y}$$

$$\mathbf{y} = (y_1, .., y_m)^T$$

$$\mathbf{W} = \begin{pmatrix} w_{11} & .. & w_{1i} & .. & w_{1n} \\ w_{j1} & .. & w_{ji} & .. & w_{jn} \\ w_{m1} & .. & w_{mi} & .. & w_{mn} \end{pmatrix}$$

Stochastic units

- **Replace the binary threshold units by binary *stochastic* units. A neuron is switched on/off with a certain probability instead of deterministically.**

Stochastic units

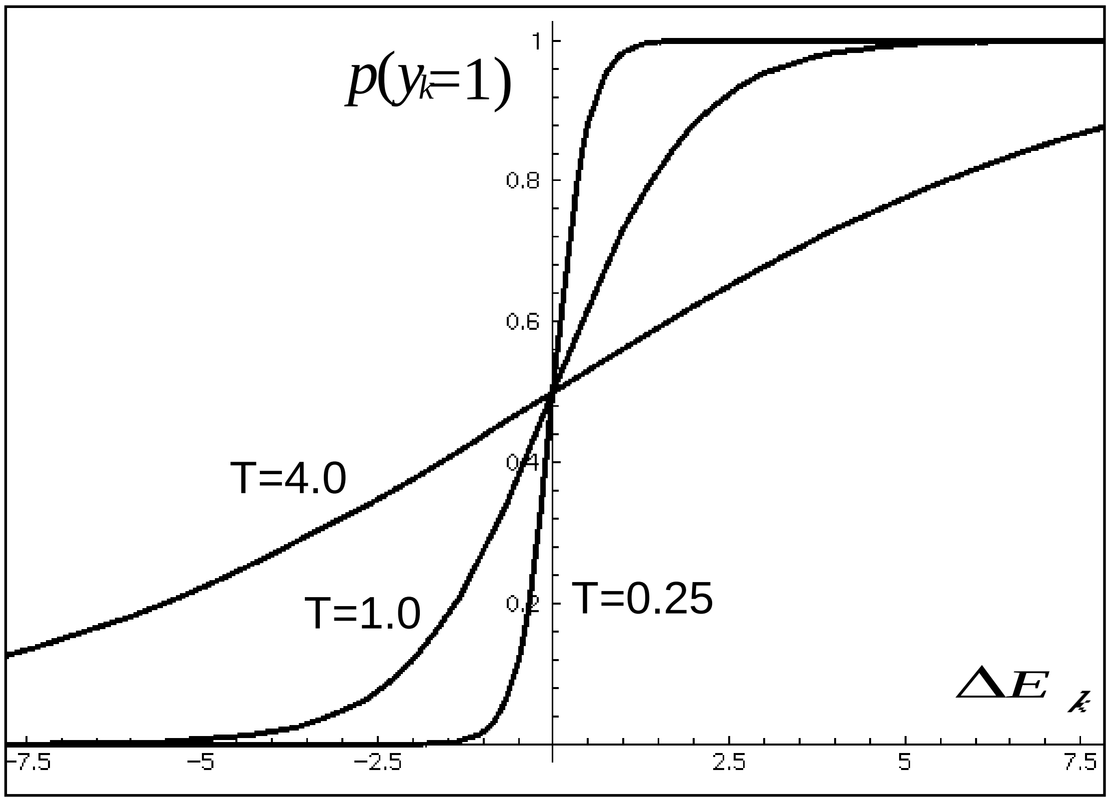
- **Reminder: the energy reduction by switching a neuron on is:**

$$z_i = \sum_j w_{ij} y_j = E(y_i = -1) - E(y_i = 1)$$

Stochastic units

- The probability that a given unit is switched on is a function of the amount of energy that its change of state would contribute to the network's overall energy, and the "temperature" T.

$$\frac{exp(z_i/T)}{exp(z_i/T) + exp(-z_i/T)} = \frac{1}{1 + exp(-2z_i/T)}$$

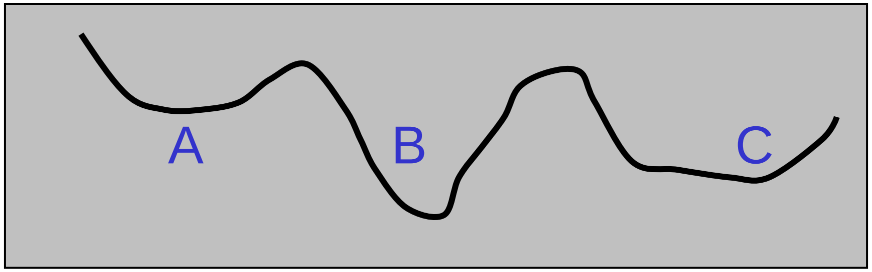

$p(y_k=1)$
1
0.8
0.6
0.4
0.2
T=4.0
T=1.0
T=0.25
ΔE_k
-7.5
-5
-2.5
2.5
5
7.5

Hopfield vs. Boltzmann

- Since in Boltzmann networks the unit update rule (or activation function) has a probabilistic component, if we allow the network to run it will settle into a given equilibrium state only with a certain probability.
- Hopfield networks, on the other hand, are deterministic. Given an initial state their equilibrium state is determined.

Stochastic units

- **Temperature makes it easier to cross energy barriers.**
  - **Start at high temperature where its easy to cross energy barriers.**
  - **Reduce slowly to low temperature where good states are much more probable than bad ones.**



Implementing a probability distribution

- **It can be shown that the Boltzmann machine generates configurations according to the distribution:**

$$P(\mathbf{y}|\mathbf{W}) = \frac{1}{Z(\mathbf{W})} exp(\frac{1}{2}\mathbf{y}^{\mathbf{T}}\mathbf{W}\mathbf{y})$$

- **Where:**

$$Z(\mathbf{W}) = \sum_{\mathbf{y}} exp(\frac{1}{2}\mathbf{y}^{\mathbf{T}}\mathbf{W}\mathbf{y})$$

Learning in the Boltzmann machine

- **Working on the probability distribution it is possible to devise the following learning rule:**

$$\Delta w_{ij} = \eta \sum_{p=1}^{P} \left[y_i^{(p)} y_j^{(p)} - \sum_{\mathbf{y}} y_i y_j P(\mathbf{y}|\mathbf{W}) \right]$$

Learning in the Boltzmann machine

- **First term:**

$$\eta \sum_{p=1}^{P} y_i^{(p)} y_j^{(p)}$$

- **Empirical correlation between neurons i and j, measured from the examples (same as learning rule for Hopfield).**

Learning in the Boltzmann machine

- **Second term:**

$$\eta \sum_{\mathbf{y}} y_i y_j P(\mathbf{y}|\mathbf{W})$$

- **Correlation between neurons i and j, measured on patterns generated according to the probability distribution underlying the model. Notice that we're summing over all possible patterns (2^N)**

Learning in the Boltzmann machine

- The first term is readily evaluated from the examples.
- The second term can be estimated by letting the model evolve until equilibrium, measuring the correlations, and repeating many times: can be computationally tough.
- Monte Carlo.

Interpretation of learning terms

- **First term: the network is awake and measures the correlations in the real world.**
- **Second term: the network sleeps and dreams about the world using the model it has of it.**
- **Once dream and reality coincide learning reaches an end. It is interesting to notice that the network *unlearns* its dreams.**

Weakness of Hopfield nets and Boltzmann machines

- All units are *visible*, i.e. correspond to observable stuff (components of the examples).
- In this situation nothing more than second order interactions can be captured by Hopfield nets and the Boltzmann machine.
- If for instance the examples are bits of images, second order statistics are a poor representation.

Hidden units for Hopfield nets/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.**
- **Higher order correlations can be represented by hidden units.**
- **More powerful model, but even harder to train.**