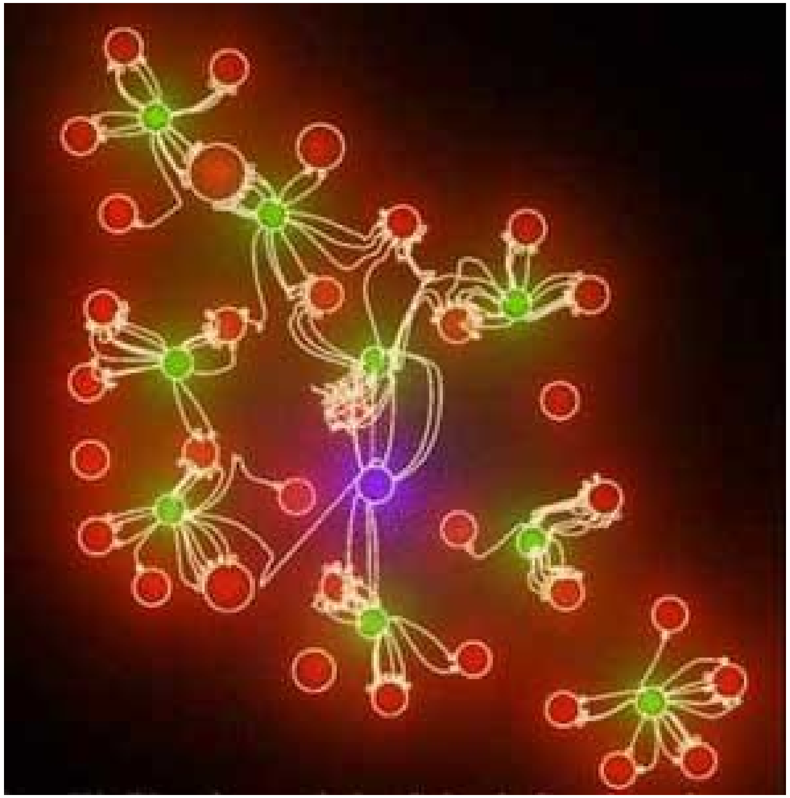
COMPSCI 367 Tutorial 6



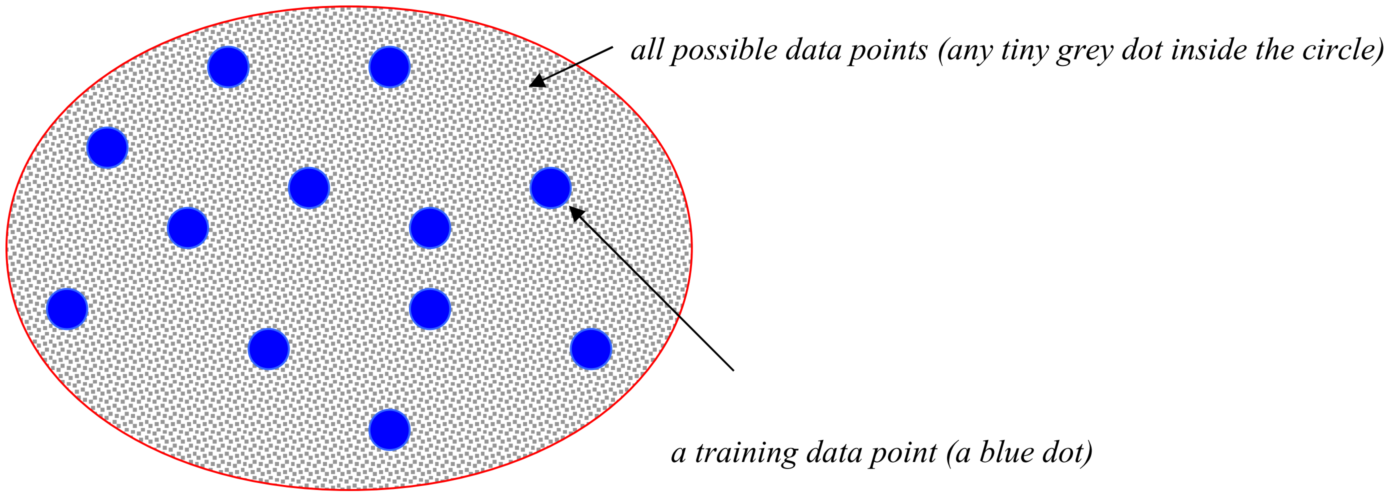
Overview

- Error, Bias & Variance
- Hopfield Nets

Learning Task

- Learn a target concept *'f'* (e.g. given the weather of a day, the function tells you if Aldo will play sport on that day).
- Consider hypothesis space, *H*
- Training examples are given to the learner – trainer chooses example independently
  - Uses a probability distribution *D* over all possible days – e.g. maybe some days are more likely to be chosen than other days (and maybe not)

Learning Task



Sample Error

- Fraction of examples in a particular sample of examples that it misclassifies
  - e.g. out of the 12 blue dots above, 3 are misclassified
  - Sample error would be 3/12 = 0.25

$$error_s(h) \equiv \frac{1}{n} \sum_{x \in S} \delta(f(x), h(x))$$

True Error

- Probability that, if I randomly pick an example (out of all possible examples) according to distribution D, it misclassifies it
- True error is what we want – but sample error is the best we can get
- So how good is sample error at estimating true error?
- Problems with sample error
  - Bias
  - Variance

Bias (similar to *accuracy*)

- Is the error between the “true value” f(x) and the average of a collection of outputs f′(x) (each from a different hypothesis) from your learner

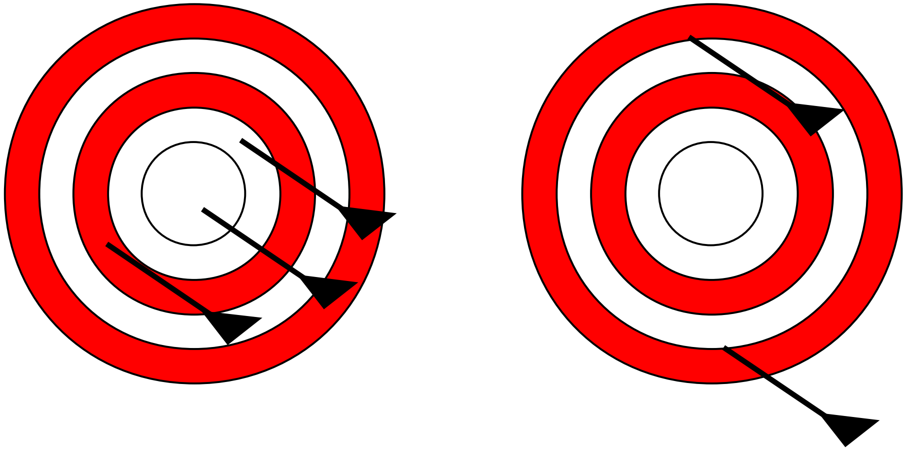
$$\text{StatBias}(A,m,x) = f'(x) - f(x)$$

Bias *(cont...)*

- Archery analogy – bull's eye is the "true value" we are aiming for and an arrow is the output from one learnt hypothesis produced by our learner

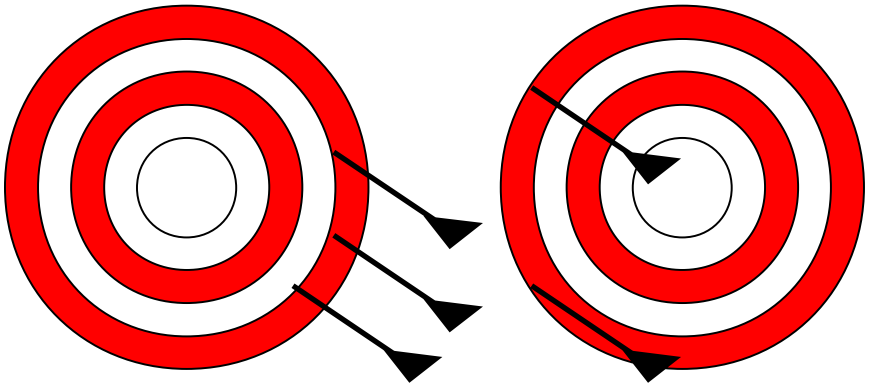
Bias *(cont...)*

- Both boards below have low bias because the error between the average of the arrows and the bull's eye is low



Bias *(cont...)*

- Whereas the following boards have high bias



Bias *(cont...)*

- we'll look at f′(x) closer – this is the average value from input *x* given by the different hypotheses that your learner might produce, depending on the training set

Bias *(cont...)*

- f_{si} is a hypothesis learnt on training set "*si*" (of size *m*)
- Just take the average of all possible "f_{si}"

$$f'(x) = \lim_{l \to \infty} \frac{1}{l} \sum_{i=1}^{l} f_{s_i}(x)$$

Where does this bias come from?

- Machine learning bias
- Systematic error bias
- Straight-statistical bias

Machine Learning Bias

- Inductive bias (we've seen this before, e.g. candidate-elimination and ID3)
- This allows the learner to generalise beyond the training data
- …but also makes assumptions that might not hold

Systematic Error Bias

- Errors that deviate from the true value in a consistent way
  - e.g. thermometer that reads 2 degrees higher than the real temperature
  - e.g. reading clock that is running fast
- Very difficult to distinguish between systematic error and real patterns in the data
- Often requires independent source of information (e.g. instrument calibration)

Straight-Statistical Bias

- States that as training set size gets smaller, error will increase

Statistical Bias

- Puts all the biases together into one formula (below)
- i.e. is the error between the “true value” f(x) and the average of a collection of hypotheses outputs f′(x) where the hypotheses are learnt from different training sets of size m from your learner

$$\mathrm{StatBias}(A,m,x) = f'(x) - f(x)$$

Variance *(similar to spread or precision)*

- This is the average (squared) error between f′(x) (the mean value of different hypotheses, each learnt from one of the training sets s_1, s_2,…s_i) and the value from each hypothesis f_{si}

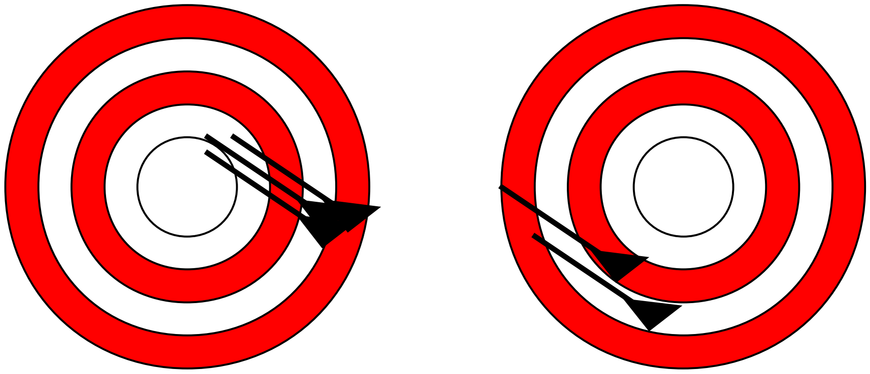
$$\text{Variance}(A,m,x) = E[(f_S(x) - f'(x))^2]$$

Variance *(cont...)*

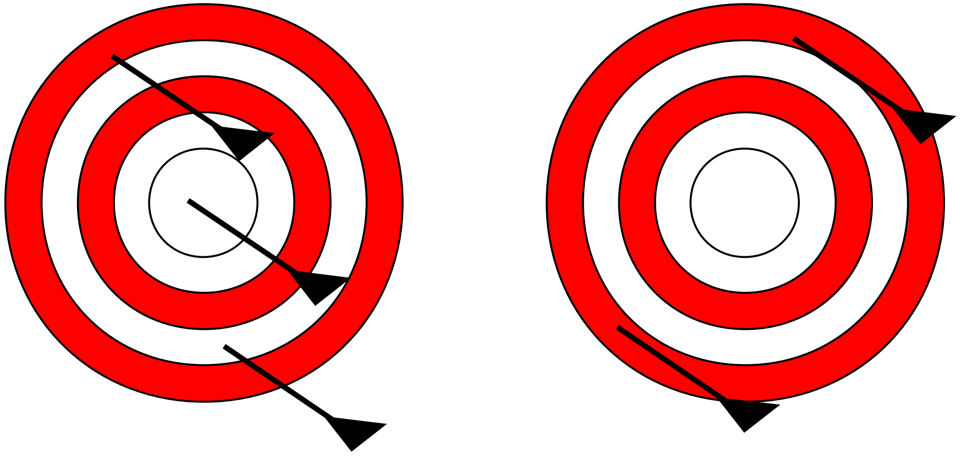
- Archery analogy – bull's eye is the "true value" we are aiming for and an arrow is the output from one learnt hypothesis produced by our learner

Variance *(cont...)*

- Both boards below have low variance because the error between (a) the average of the arrows and (b) each particular arrow is low (i.e. all arrows are near the average)



- Whereas the following boards have high variance:

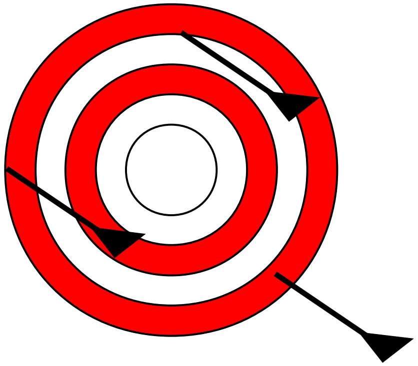


Variance *(cont...)*

- *high variance* = arrows are spread out
- *low variance* = arrows are near each other

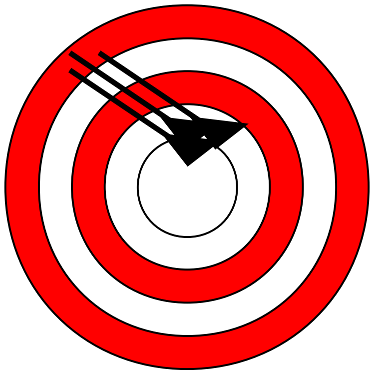
Bias + Variance

- note: variance can be high while bias is low (low precision but high accuracy – remember bias is the average of the outputs from different hypotheses)



Bias + Variance

- note: variance can be low while bias is high (high precision but low accuracy)



Error

- Error (mean squared error) = $bias^2$ + variance

$$\mathrm{Error}(A,m,x)=\mathrm{Bias}(A,m,x)^2+\mathrm{Variance}(A,m,x)$$

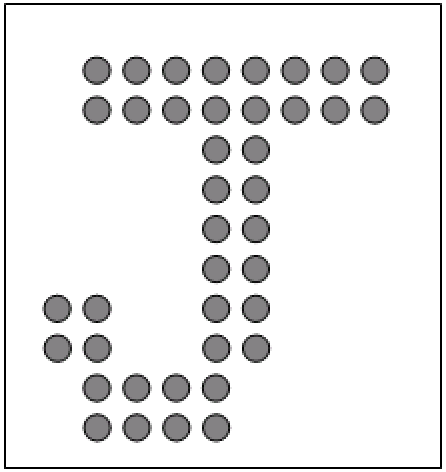
Hopfield Nets

- Kevin Gurney, "Associative Memories – the Hopfield Net" available at:
- http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.40.8237

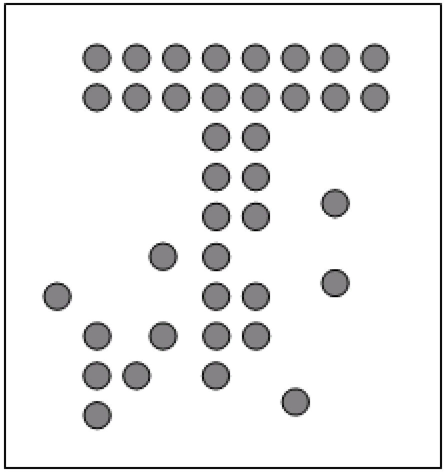
Hopfield Nets

- Content-Addressable Memory
- Can learn some patterns (e.g. image of letter "J") and given a partial pattern (slightly scrambled image of "J") will reproduce the nearest learnt pattern
  - *see next slide*

Hopfield Nets



original image “J”

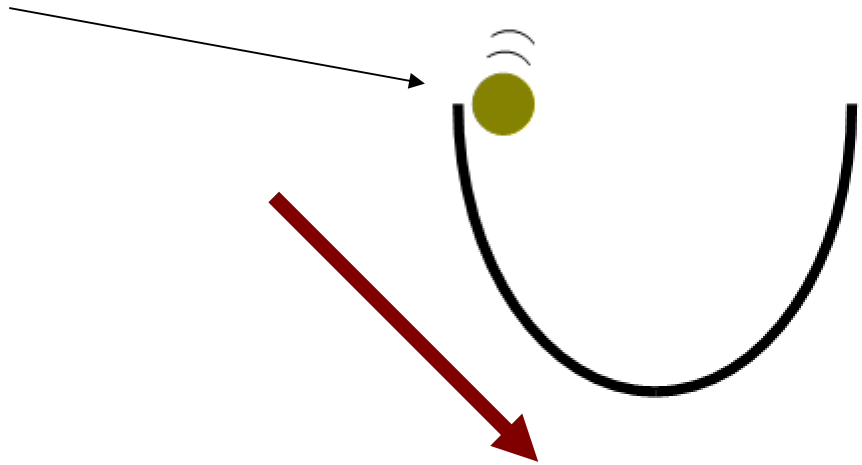


partially corrupt image

A Physical Analogy with Memory

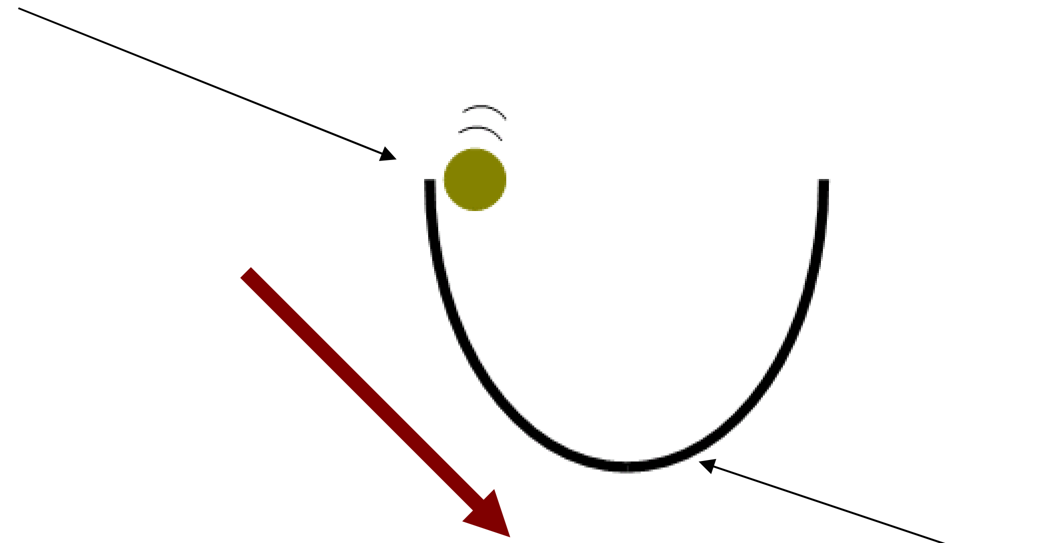
- We have a boulder rolling down a valley

Push the ball from some initial state



A Physical Analogy with Memory

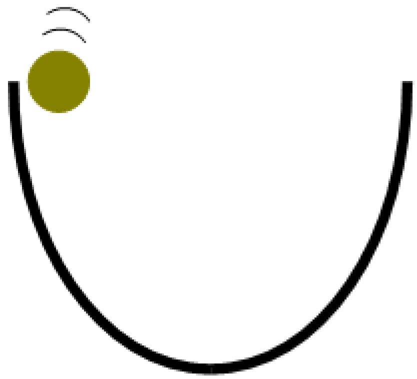
Potential energy => *Kinetic* energy



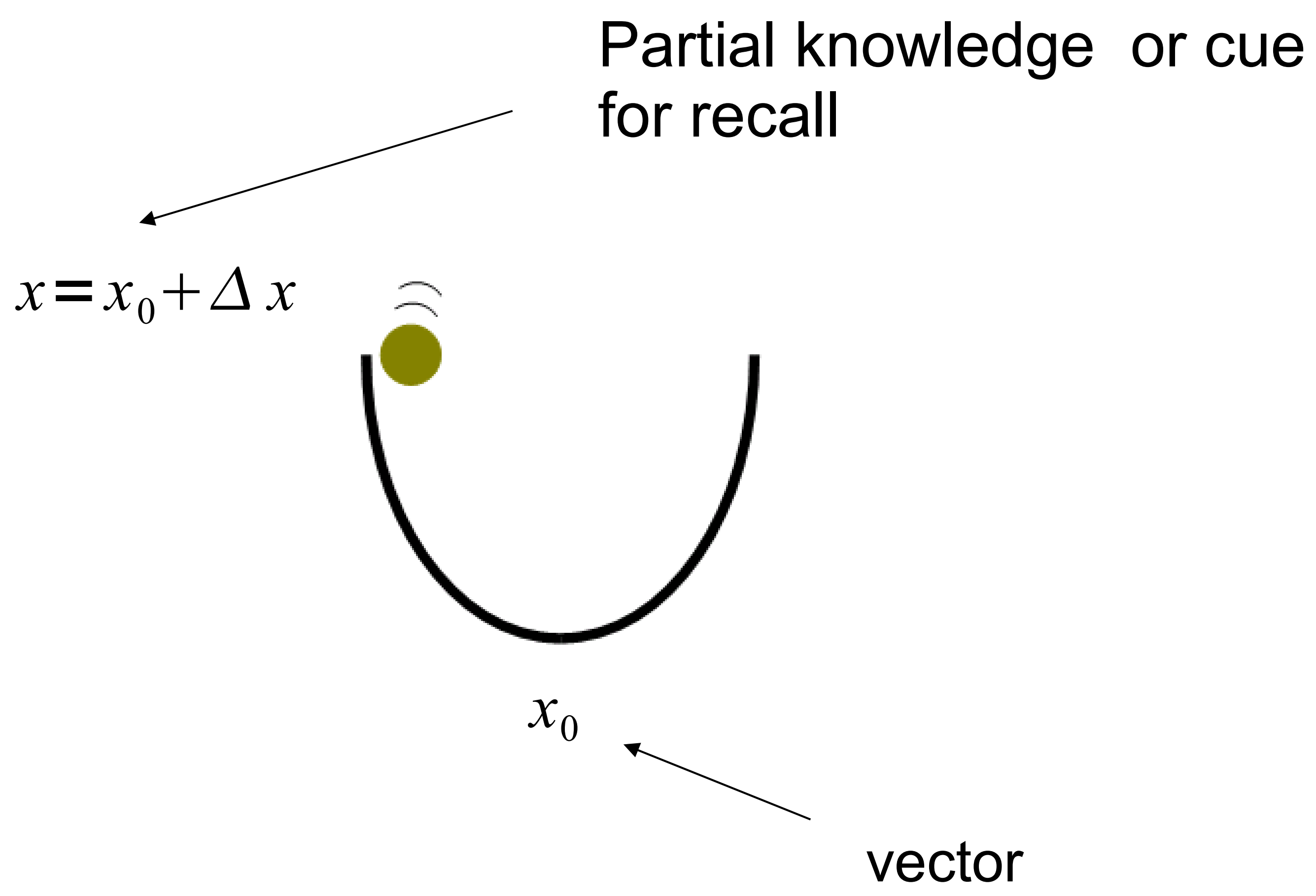
Ball comes to rest at the energy minimum of the system

A Physical Analogy with Memory

- The resting state is said to be *stable* because the system remains there after it has been reached.
- Or (another way of thinking), the ball "remembers" where the bottom of the bowl is.



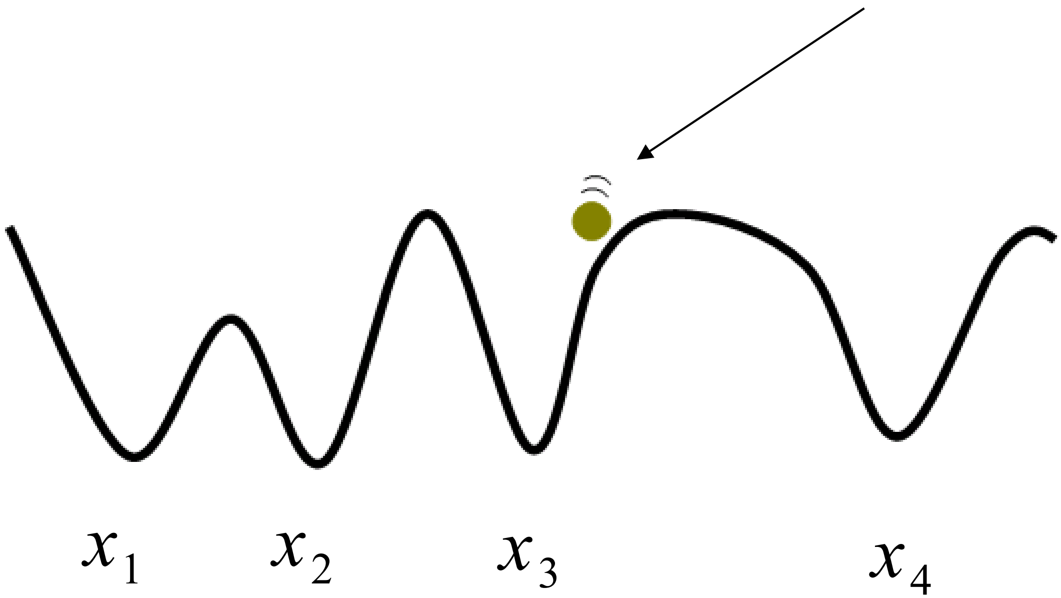
A Physical Analogy with Memory



A Physical Analogy with Memory

- Corrugated surface

Depending on where the ball is placed it evokes the closest stored memory



Stored sets of patterns

Summary of physical system

- If we were to build a network which behaves like this we must include the following:
  - 1) It is completely described by a state vector

    $v = (v_1, v_2, \ldots, v_n)$
  - 2) There are a set of stable states $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$. These will correspond to the stored patterns.
  - 3) The system evolves in time from any arbitrary starting state, *v*, to one of the stable states and this may be described as the system decreasing its energy *E*. The corresponds to the process of memory recall.

Hopfield Nets

- Hopfield network has units that are either active or passive
- Weighted, symmetric connections between units
- A pattern can be described as some particular combination of active units in a state (each unit has a unique name or identifier)

Hopfield Nets

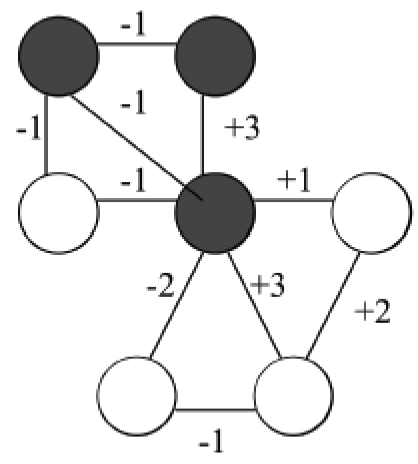
- The bottom of the valley represents the pattern that the Hopfield net has learnt
- Wherever the boulder is initially placed, it will roll towards the nearest local minimum – this represents the Hopfield net iteratively processing the next network state
- The boulder will eventually stop rolling at the bottom of the valley – this represents the stable state of the Hopfield network

Hopfield Nets

- Depending on where the boulder is initially placed it will roll towards the nearest valley – given some partial pattern, the Hopfield network (using the parallel relaxation algorithm) will eventually stabilise at the closest matching pattern

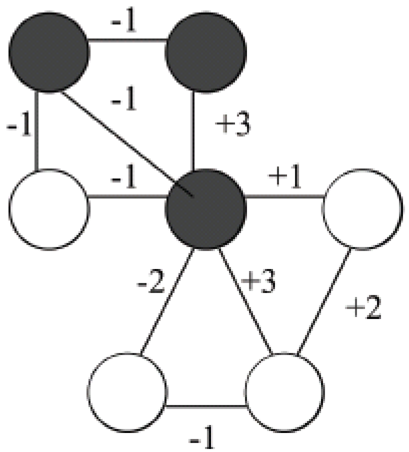
Hopfield Nets

- 1) State vector (each units state).
- 2) Stable states (states where no units change from active to passive and vice versa)
- 3) Parallel relaxation algorithm



Hopfield Nets

- Learning a pattern = adjusting the weights
- Recalling a pattern = parallel relaxation algorithm



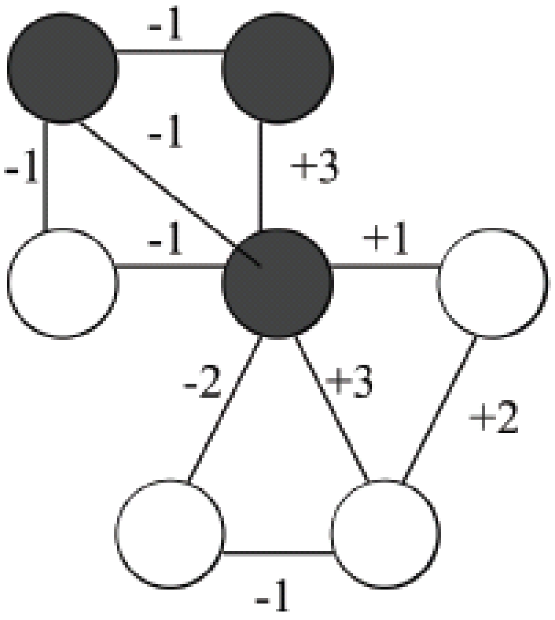
Hopfield Nets

Parallel Relaxation in Hopfield Networks

- A random unit is chosen.
- If any of its neighbors are active, the unit computes the sum of the weights on the connections to those active neighbors.
- If the sum is positive, the unit becomes active; otherwise it becomes inactive.
- The process (parallel relaxation) is repeated until the network reaches a stable state.

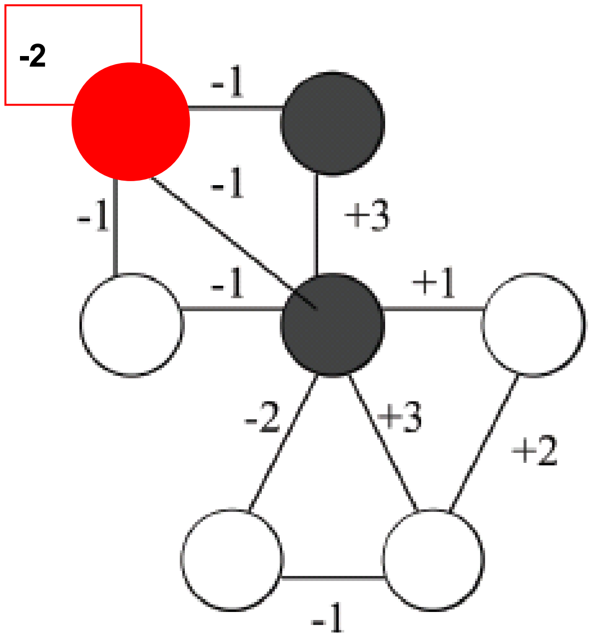
Hopfield Nets *(example)*

initial



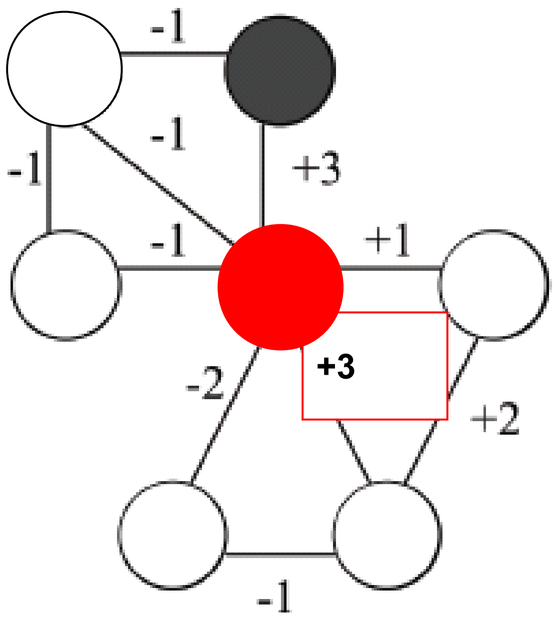
Hopfield Nets *(example)*

step 1



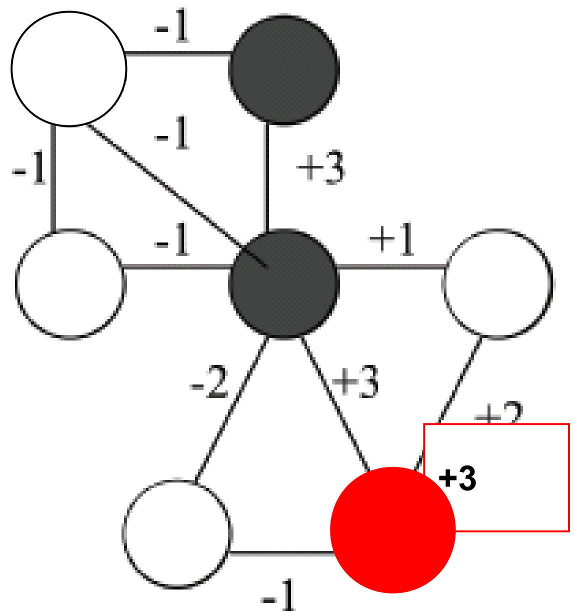
Hopfield Nets *(example)*

step 2



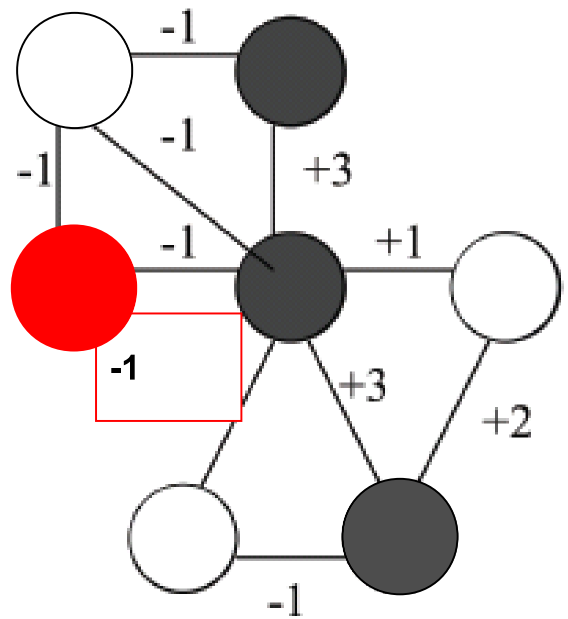
Hopfield Nets *(example)*

step 3



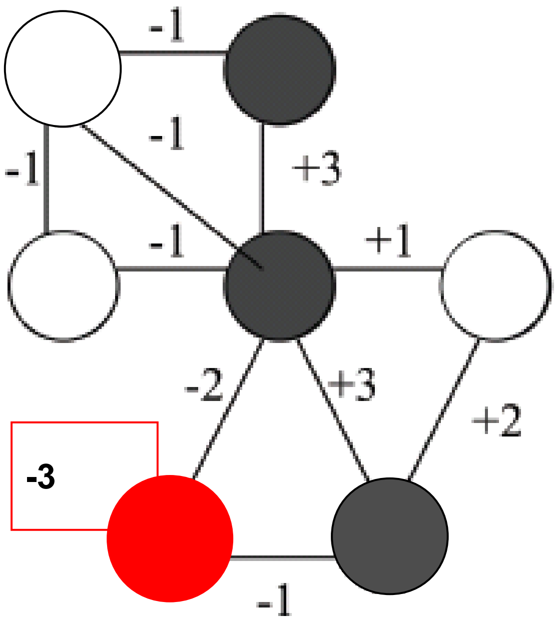
Hopfield Nets *(example)*

step 4

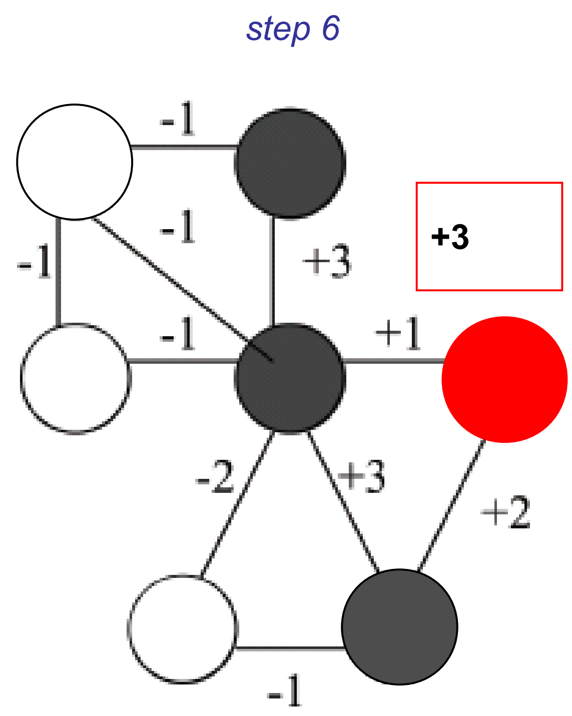


Hopfield Nets *(example)*

step 5



Hopfield Nets *(example)*



Hopfield Nets *(example)*

(continue until stable)