

The eight queens problem
- a neural network approach

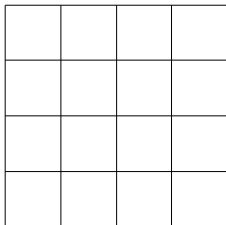
Martin Leslie

MATH577 Project

Fall '09

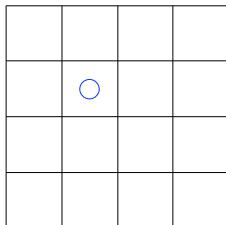
The problem

- ▶ We want to place 8 queens on a standard 8×8 chessboard so that no queen can attack another.
- ▶ Remember queens attack vertically, horizontally and diagonally.
- ▶ An attempt at a 4×4 solution:



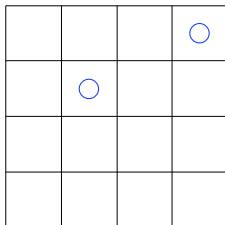
The problem

- ▶ We want to place 8 queens on a standard 8×8 chessboard so that no queen can attack another.
- ▶ Remember queens attack vertically, horizontally and diagonally.
- ▶ An attempt at a 4×4 solution:



The problem

- ▶ We want to place 8 queens on a standard 8×8 chessboard so that no queen can attack another.
- ▶ Remember queens attack vertically, horizontally and diagonally.
- ▶ An attempt at a 4×4 solution:



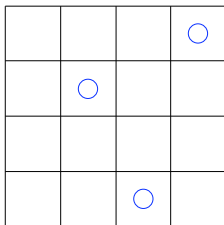
The problem

- ▶ We want to place 8 queens on a standard 8×8 chessboard so that no queen can attack another.
- ▶ Remember queens attack vertically, horizontally and diagonally.
- ▶ An attempt at a 4×4 solution:

			○
	○		
		○	

The problem

- ▶ We want to place 8 queens on a standard 8×8 chessboard so that no queen can attack another.
- ▶ Remember queens attack vertically, horizontally and diagonally.
- ▶ An attempt at a 4×4 solution:



- ▶ And we're stuck!

The neural network

- ▶ We look to Hopfield and Tank's approach to the traveling salesman problem for inspiration.
- ▶ We have N^2 nodes in our network each representing a square of the chessboard: 0 for no queen and 1 for a queen.
- ▶ We find an energy function E that is minimized for a solution to our problem.
- ▶ Then neuron $n_{r,s}$ has activation

$$a_{r,s} = -\frac{\partial E}{\partial n_{r,s}}$$

and updated value

$$n_{r,s} = \frac{1}{1 + e^{-\beta a_{r,s}}}.$$

The energy function

$$\begin{aligned} E = & \text{term for columns} + \text{term for rows} \\ & + \text{term for NW-SE diagonals} \\ & + \text{term for NE-SW diagonals} \\ & + \text{term to have } N \text{ queens} \end{aligned}$$

The energy function

$$E = \sum_{j=1}^N \left(1 - \sum_{i=1}^N n_{ij} \right)^2 + \text{term for rows}$$

- + term for NW-SE diagonals
- + term for NE-SW diagonals
- + term to have N queens

The energy function

$$E = \sum_{j=1}^N \left(1 - \sum_{i=1}^N n_{ij} \right)^2 + \sum_{i=1}^N \left(1 - \sum_{j=1}^N n_{ij} \right)^2$$

- + term for NW-SE diagonals
- + term for NE-SW diagonals
- + term to have N queens

The energy function

$$E = \sum_{j=1}^N \left(1 - \sum_{i=1}^N n_{ij} \right)^2 + \sum_{i=1}^N \left(1 - \sum_{j=1}^N n_{ij} \right)^2$$

+ $\sum_{i=1}^N \sum_{j=1}^N n_{ij}$ (# of other 1's on NW-SE diagonal containing i, j)

+ term for NE-SW diagonals

+ term to have N queens

The energy function

$$E = \sum_{j=1}^N \left(1 - \sum_{i=1}^N n_{ij} \right)^2 + \sum_{i=1}^N \left(1 - \sum_{j=1}^N n_{ij} \right)^2$$
$$+ \sum_{i=1}^N \sum_{j=1}^i n_{ij} \sum_{k=1}^{N-(i-j)} n_{k+i-j,k} + \sum_{i=1}^N \sum_{j=i+1}^N n_{ij} \sum_{k=1-(i-j)}^N n_{k+i-j,k}$$

+ term for NE-SW diagonals

+ term to have N queens

The energy function

$$\begin{aligned} E = & \sum_{j=1}^N \left(1 - \sum_{i=1}^N n_{ij} \right)^2 + \sum_{i=1}^N \left(1 - \sum_{j=1}^N n_{ij} \right)^2 \\ & + \sum_{i=1}^N \sum_{j=1}^i n_{ij} \sum_{k=1}^{N-(i-j)} n_{k+i-j,k} + \sum_{i=1}^N \sum_{j=i+1}^N n_{ij} \sum_{k=1-(i-j)}^N n_{k+i-j,k} \\ & + \sum_{i=1}^N \sum_{j=1}^{N+1-i} n_{ij} \sum_{k=1}^{i+j-1} n_{i+j-k,k} + \sum_{i=1}^N \sum_{j=N+2-i}^N n_{ij} \sum_{k=i+j-N}^N n_{i+j-k,k} \\ & + \text{ term to have } N \text{ queens} \end{aligned}$$

The energy function

$$\begin{aligned}
 E = & \sum_{j=1}^N \left(1 - \sum_{i=1}^N n_{ij} \right)^2 + \sum_{i=1}^N \left(1 - \sum_{j=1}^N n_{ij} \right)^2 \\
 & + \sum_{i=1}^N \sum_{j=1}^i n_{ij} \sum_{k=1}^{N-(i-j)} n_{k+i-j,k} + \sum_{i=1}^N \sum_{j=i+1}^N n_{ij} \sum_{k=1-(i-j)}^N n_{k+i-j,k} \\
 & + \sum_{i=1}^N \sum_{j=1}^{N+1-i} n_{ij} \sum_{k=1}^{i+j-1} n_{i+j-k,k} + \sum_{i=1}^N \sum_{j=N+2-i}^N n_{ij} \sum_{k=i+j-N}^N n_{i+j-k,k} \\
 & + \left(\sum_{i=1}^N \sum_{j=1}^N n_{ij} - N \right)^2
 \end{aligned}$$

The constants

- ▶ Actually, we should have some constants out the front of our energy terms:
 - ▶ $\gamma/2$ for column and row terms;
 - ▶ $\delta/2$ for diagonal terms;
 - ▶ $\epsilon/2$ for N queens term.
- ▶ Also, β is inverse temperature.

Taking derivative of E

- ▶ Taking the derivative isn't too bad. For example

$$\begin{aligned} & \frac{\partial}{\partial n_{rs}} (\text{NW-SE diagonal term}) \\ &= \frac{\partial}{\partial n_{rs}} \sum_{i=1}^N \sum_{j=1}^N n_{ij} (\# \text{ of other 1's on diag. including } i, j) \\ &= (\# \text{ of other 1's on diag. including } r, s) + \\ & \quad \sum (n_{ij} \text{ where } i, j \text{ is on same diagonal as } r, s) \\ &= 2 (\# \text{ of other 1's on diag. including } r, s) \end{aligned}$$

Implementing the network

- ▶ Fix constants: I took $\gamma = \delta = \epsilon = 1$ to start with.
- ▶ Create an initial state $n = (n_{ij})$ with random entries between 0 and 1.
- ▶ Run through the neurons in order (asynchronously) and update them as discussed earlier.

Does it work?

- ▶ No.
- ▶ If we take β fixed and small, say $\beta = 1$ or 0.1 , then n converges to a state where all the entries are approximately equal. This isn't a very good solution.
- ▶ If we take β fixed but larger, say $\beta = 10$, then n quickly settles down from its initial high energy random state but then oscillates wildly.

The fix: ‘continuous’ network + simulated annealing

- ▶ We use the update rule

$$n_{r,s} = n_{r,s} + \left(\frac{1}{1 + e^{-\beta a_{r,s}}} - n_{r,s} \right) \Delta t$$

with $\Delta t = 0.1$ to make the oscillations less wild.

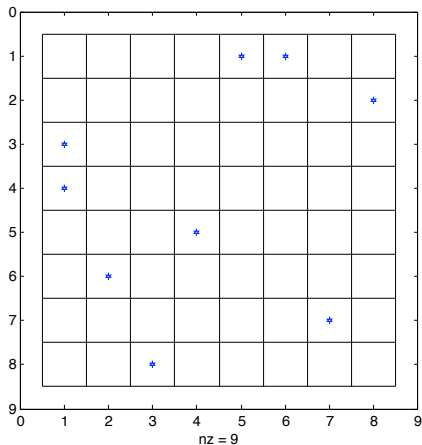
- ▶ Also we cool down the temperature as the network runs by setting the inverse temperature to be

$$\beta = 20 \left(\frac{l}{L} \right)^2$$

when on step l of a total L .

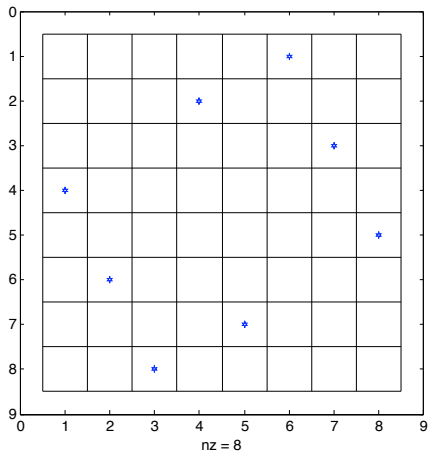
Does it work now?

- ▶ With $\gamma = \delta = \epsilon = 1$ and $L = 10000$ we converge to an almost solution:



One way to fix it

- ▶ If we make $L = 20000$, giving the system more time to bounce around, it converges to the following solution:



Another way

- ▶ We can keep $L = 10000$, but change γ to 1.1. This finds the following solution:

