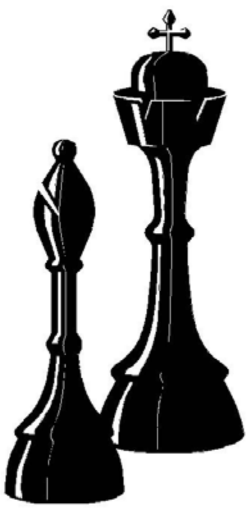
Complex Problem Solving With Neural Networks: Learning Chess

Mr. Jack Sigan

Dr. Aleksander Malinowski, Advisor

Dept. of Electrical and Computer Engineering

BRADLEY UNIVERSITY



November 5, 2004

Outline

- Neural network introduction
- Chess and neural networks
- Chess-specific network architectures
  - Geographical approach
  - Functional approach
- Data representations and input vectors

Neural Networks

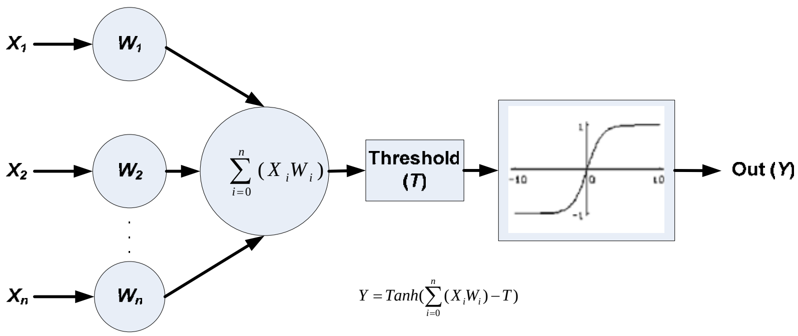


Figure 1: Node structure with hyperbolic tangent activation function

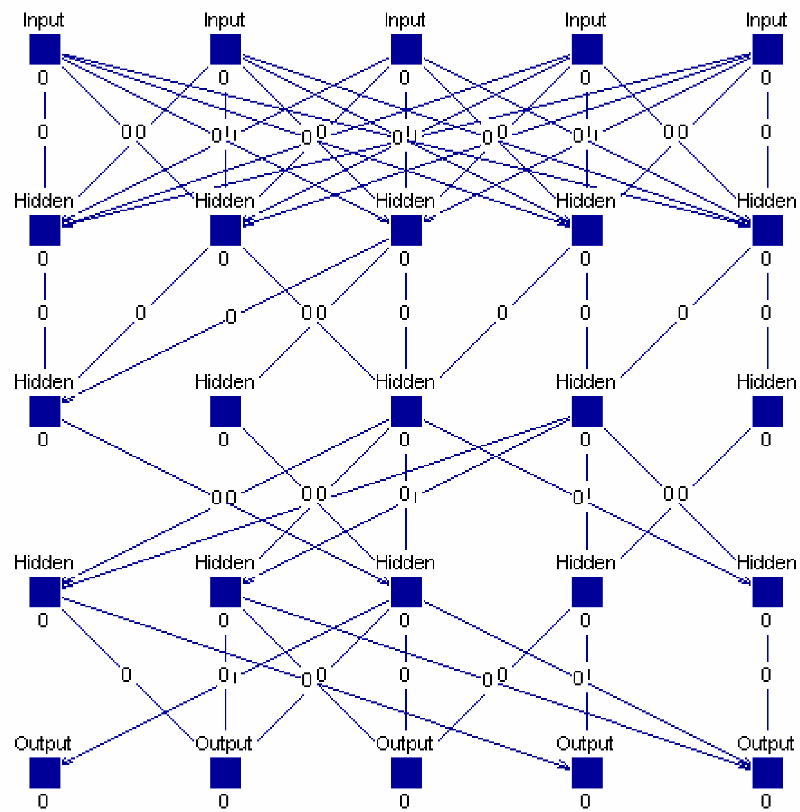


Figure 2: Simple partially connected neural network structure from Stuttgart Neural Network Simulator (SNNS)

Chess and Neural Networks?

- Demonstrates complex decision making
- Highly nonlinear problem
- Schemas
- Widely studied
- Massive amounts of available data
- Success with checkers
- Mixed results with chess in the past

Parallel Network Designs

- Decrease learning cycle time
- Less destructive learning process
- Multiple “suggested” moves may be returned
- Simpler network architectures
- 2 paradigms for deconstructing chess
  - Geographical “move based”
  - Functional “piece based”
- External logic is applied to both paradigms to filter out illegal moves or impossible moves

Parallel Network Designs

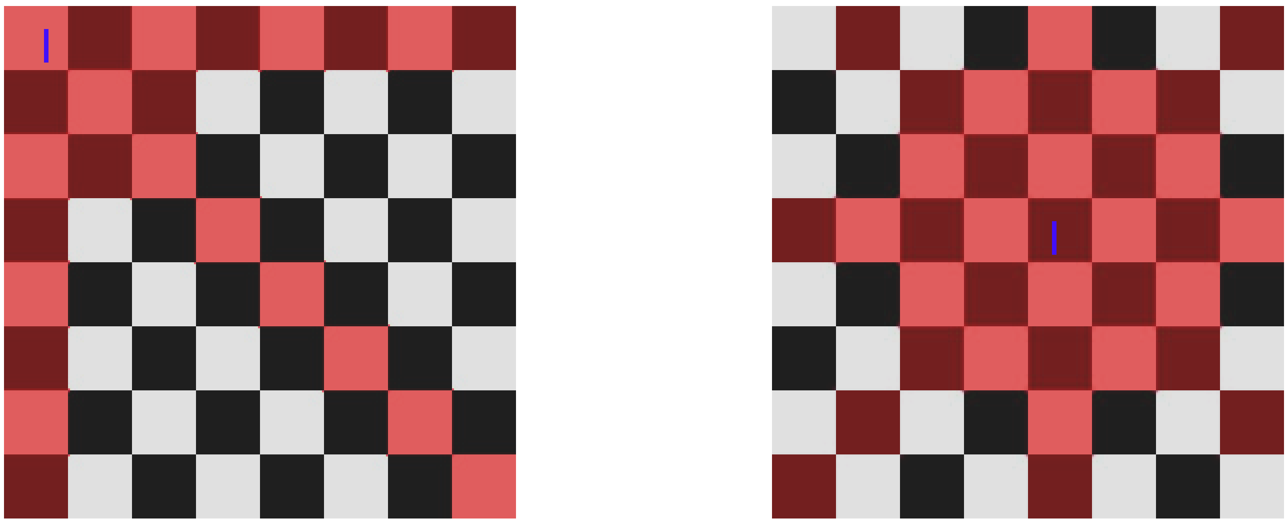


Figure 3: Mapping the legal moves in chess, using overlay consisting of queen + knight moves

Excluding castling, there are 1856 moves possible — ignoring the piece type which is moved

Example: Moving from d3 to d4 is considered ONE possible move, whether the piece is a pawn, queen, king, etc.

Approach A: Geographical

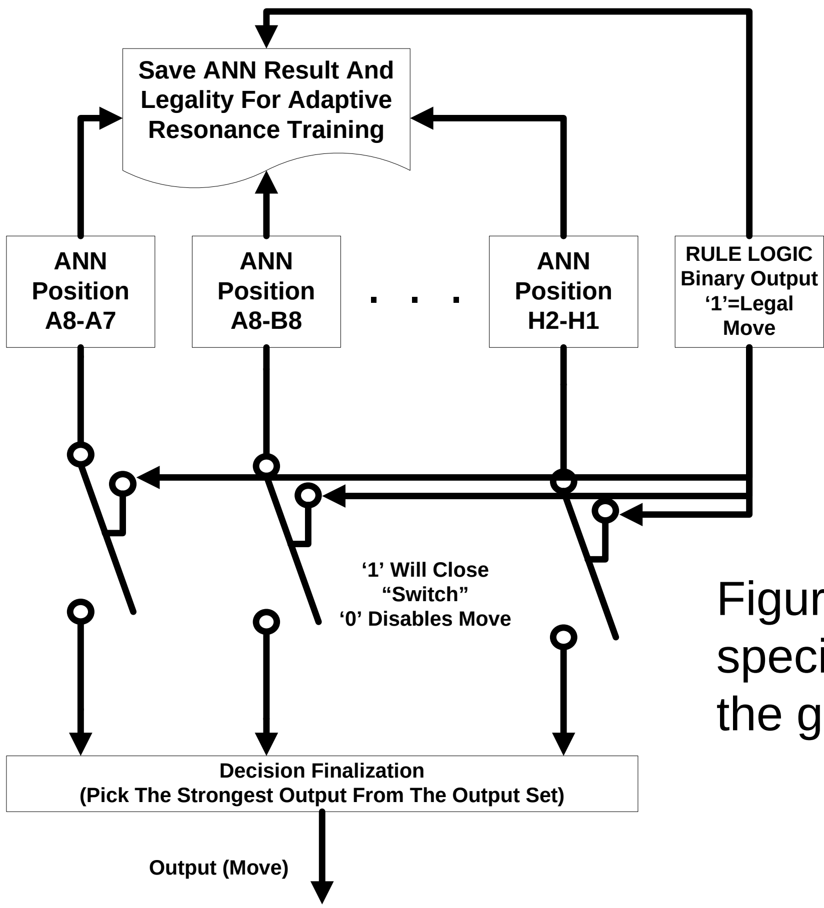


Figure 4: Design with "move specific" neural networks—the geographical approach

Approach B: Functional

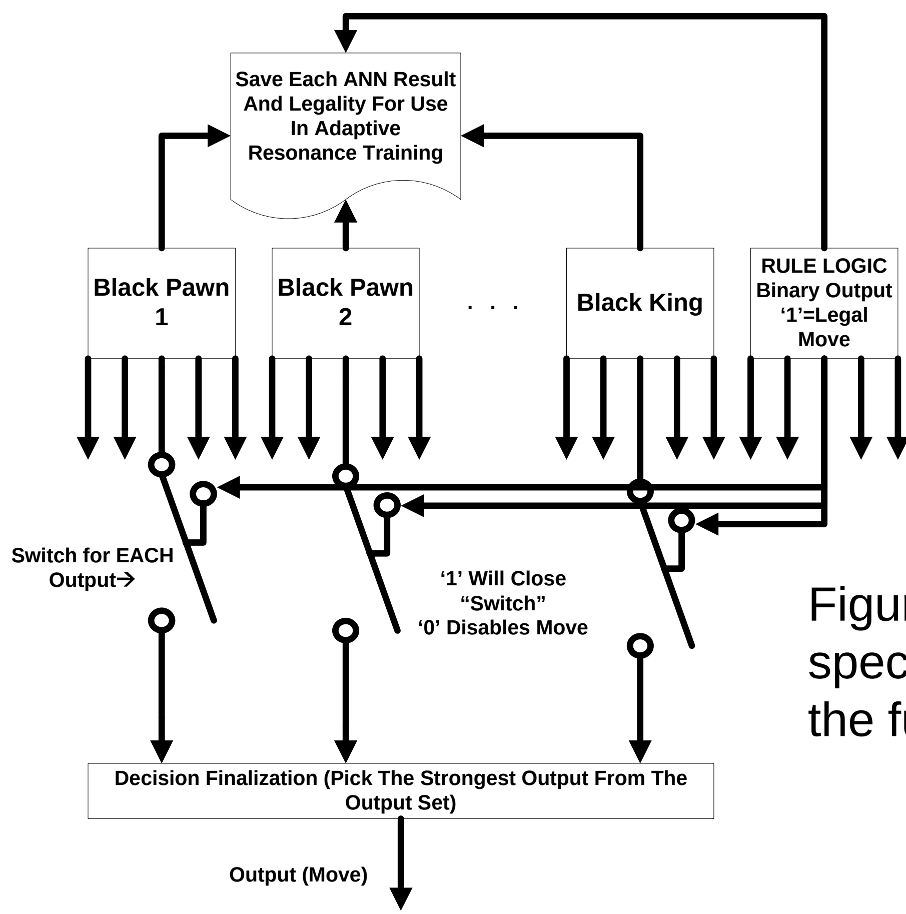


Figure 5: Design with “piece specific” neural networks—the functional approach

Data Representations

```
 1. e4 d6      2. d4 Nf6  3. Nc3 g6    4. Nf3 Bg7    5. Be2 O-O    6. O-O Bg4
 7. Be3 Nc6    8. Qd2 e5  9. d5 Ne7    10. Rad1 Bd7  11. Ne1 Ng4   12. Bxg4 Bxg4
13. f3 Bd7    14. f4Bg4  15. Rb1 c6    16. fxe5 dxe5 17. Bc5 cxd5  18. Qg5 dxe4
19. Bxe7 Qd4+ 20. Kh1f5  21. Bxf8 Rxf8 22. h3 Bf6    23. Qh6 Bh5   24. Rxf5 gxf5
25. Qxh5 Qf2  26. Rd1e3  27. Nd5 Bd8   28. Nd3 Qg3   29. Qf3 Qxf3  30. gxf3 e4
31. Rg1+ Kh8  32. fxe4 fxe4 33. N3f4 Bh4 34. Rg4 Bf2 35. Kg2 Rf5   36. Ne7 1-0
```

Figure 6: Example of the PGN (algebraic) standard

```
rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - pm d4;
rnbqkbnr/pppppppp/8/8/3P4/8/PPP1PPPP/RNBQKBNR b KQkq d3 pm Nf6;
rnbqkb1r/pppppppp/5n2/8/3P4/8/PPP1PPPP/RNBQKBNR w KQkq - pm Nf3;
rnbqkb1r/pppppppp/5n2/8/3P4/5N2/PPP1PPPP/RNBQKB1R b KQkq - pm b6;
```

Figure 7: Example of the EPD (string) format

Preprocessing

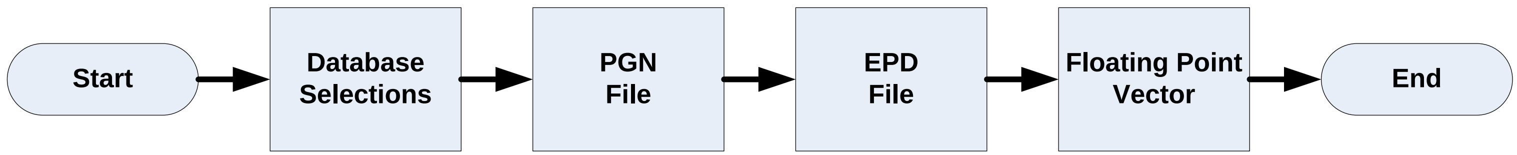


Figure 8: The game data preprocessing procedure

•Board positional data, plus a ‘yes’ or ‘no’ decision in regards to making the specific move must be in the floating point vector.

•A 50/50 mix of ‘yes’ and ‘no’ samples will be used in all training sets

•The geographical approach does not differentiate between pieces to be moved, only the initial and final positions are important

Input Vector Creation

Piece	EPD Character Black, white	Weight Black, white
King	k,K	1.0, -1.0
Queen	q,Q	0.9, -0.9
Rook	r,R	0.5, -0.5
Knight	n,N	0.4, -0.4
Bishop	b.B	0.3, -0.3
Pawn	p,P	0.1, -0.1

Figure 9: “Standard” values for pieces used in the floating point input vector creation

```
# Input pattern 1:
0.5 0.4 0.3 0.9 1 0.3 0.4 0.5
0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
-0.1 -0.1 -0.1 -0.1 -0.1 -0.1 -0.1 -0.1
-0.5 -0.4 -0.3 -0.9 -1 -0.3 -0.4 -0.5
# output pattern 1:
1
```

Figure 10: Floating point input vector for the initial board position

Questions?

Additional questions and comments are invited. Please contact:
Jack Sigan, jsigan@bradley.edu

Parallel Network Designs

- starting position *i*
- final position *f*
- $m_{if}=f(b_i)$ represents all legal moves for a board position b_i
- game *g* of *n* moves may be expressed as a set of board positions *bi, biЄg,* where *i* is the position number 0 to *n*.

$$L = \sum_{i=0}^{n} f(b_i)$$ Legal moves for a game of *n* positions

$$M = \sum_{i=0}^{\infty} (L_i)$$ Legal moves for chess (all games)

•In this design, it is required to create an individual ANN structure for all moves *t*, *tЄM. M*=1856, ignoring castling

Parallel Network Designs

- starting position *i*
- final position *f*
- piece *p*
- $m_{if}=f(p,b_i)$ represents all legal moves for a piece in b_i
- game *g* of *n* moves may be expressed as a set of board positions *bi, biЄg,* where *i* is the move number 0 to *n*.

$$L = \sum_{i=0}^{n} f(p, b_i)$$ Legal moves for a game of *n* moves

$$M = \sum_{i=0}^{\infty} (L_i)$$ Legal moves for a piece (all games)

•In this design, it is required to create an individual ANN structure for all pieces *p*, *p*=16