Solving the Quasigroup problem using Simulated Annealing

Samuel Amin

Quasigroup Problem Definition

- Given a partial assignment of colors, can the partial quaisgroup be completed to obtain a full quasigroup?
- No color should be repeated in any row or column
- 10 by 10 Grid with 10 possible colors for each square

Simulated Annealing

- An approach that resembles simple hill climbing, but occasionally a non optimal step is taken to avoid local minima.
- The probability of taking a non optimal step decreases over time.

Algorithm

- Function SIMULATED-ANNEALING(problem, schedule) returns a solution state

```
current<- initial state of problem
for t <- 1 to infinity do
    T<- schedule[t]
    if T = 0 then return current
    next<- randomly selected successor of current
    E <- VALUE[next] – VALUE [current ]
    if E > 0 then current<- next
    else current<- next only with probability e^(E/T)
```

Adjusting Quasigroup problem for Simulated Annealing

- Initial State
  - Set the predefined values to the grid, and mark them as predefined. These squares will not be altered
  - Randomly fill out remaining squares on grid while ensuring that there are exactly 10 instances of each color.
- To get the next state, randomly swap two squares on grid that are not predefined
- Value of Grid is 100 – Number of repeated squares

Progress and Problems faced

- Tweaking schedule of T
- Local Minima

Handwritten Character Recognition using Neural Networks

CSE 592 Project

Samer Arafeh

System Architecture

- Image (bitmap) Object
  - 16x16 bitmap scaling
  - I/O
- Neural network object
  - Training and learning
  - Recognition
- User interface
  - Hand-write characters
  - Controls learning rate
  - Save learned data

Neural Network

- Multi-layer: 3 Layers neural network

  - 256 Input nodes (node for each for each input pixel)

  - variable number of hidden nodes (currently set to 25)

  - 36 output nodes (0-9 and 'A' to 'Z')

Network nodes evaluation

- 256 input nodes: 0.5 if pixel is on, otherwise -0.5.
- Hidden nodes and output nodes are calculated using the sigmoid threshold unit as:

$$o = 1/(1+e^{-net}) \text{ where}$$

$$net = \sum w_i x_i \quad \text{(over all incoming edges)}$$

Backpropagation

- Hidden and Output weights are initialized to random values between [-0.5,0.5]
- For each output node, calculate the error term δ_k as:

$$\delta_k = (t_k - o_k)$$

- Back propagate the error term to the hidden nodes such that, for each hidden node, calculate the error term δ_h as:

$$\delta_h = \sum w_{kh}\delta_k \quad \text{(over all hidden node edges)}$$

Training

- For each hidden node, re-evaluate each of the output node weight edges (w_{newo}) as:

$$w_{newo} = w_{oldo} + (\eta\ \delta_k h) \ ; \quad \text{h is the hidden node value, } \eta \text{ is the learning rate}$$

- For each input node, re-evaluate each of the hidden node weight edges (w_{newh}) as:

$$w_{newh} = w_{oldh} + (\eta\ \delta_h x) \ ; \quad \text{x is the input node value, } \eta \text{ is the learning rate}$$

Recognition

- Run the re-evalaution algorithm again with the new set of weighted edges and find the output node with the largest which would correspond to the recognized character.

Demo

Robocode an AI Playground

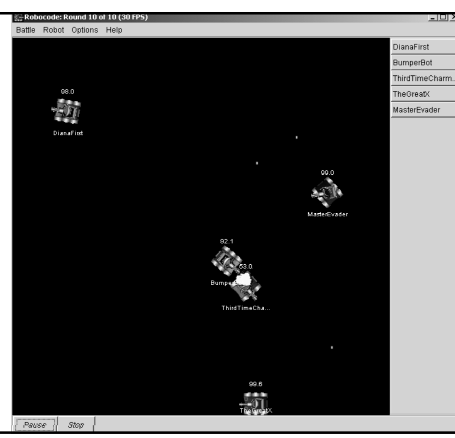
by Diana Bullion

IBM’s Robocode

- IBM’s RoboCode
- Virtual platform to test AI concepts
- Little tanks battle each other
- Each tank has a gun and radar
- Each tank is allotted the same resources (energy, ammunition)

Robots

- Built 5 Robots with different strategies
  - Diana’s First …simple tutorial-like robot
  - BumperBot …brute force tank
  - ThirdTimeCharmer …focused attack
  - TheGreatX …stays out of the way
  - MasterEvader … predicts aiming point
- Implement multiple robots with varying levels of intelligence
- Wanted to prove intelligence and strategy wins over brute force



BumperBot

- Basic robot scans for other robots
- Bumps into them and repeatedly shoots
- Brute force - low intelligence
  - Does not predict where robot will be
  - Does not stay focused on closest robot when different robot is scanned
- Results were surprising - original objective was for the more intelligent robots to win against BumperBot

MasterEvader

- Advanced Robot
- Evasive Movements … random figure-eightish pattern
- Predicts best path to fire bullet … taking into account future speed and location of both target and source robots, time to turn gun, time for bullet to travel
- Fire power relative to target distance

The Rest

- ThirdTimeCharmer
  - Advanced Robot
  - Maintains a focused attack
  - Standard movement pattern
- TheGreatX
  - Travels great distances
  - Rarely shoots
  - Lets others run out of energy
- Diana's First
  - My first robot … modified tutorial

Results

Results for 50 rounds

Robot Name	Total Score	Survival	Last Survivor Bonus	Bullet Dmg	Bonus	Ram Dmg * 2	Bonus	Survival 1sts	Survival 2nds	Survival 3rds
1st: dnb.MasterEvader	18376	7650	1240	6538	796	1870	280	31	3	6
2nd: dnb.BumperBot	10791	4500	240	5069	687	285	10	6	8	12
3rd: dnb.TheGreatX	8210	6850	440	742	15	152	10	11	22	12
4th: dnb.ThirdTimeCharmer	7255	4000	0	309	3	2730	213	0	12	15
5th: dnb.DianaFirst	3587	1950	80	1434	70	52	0	2	6	4

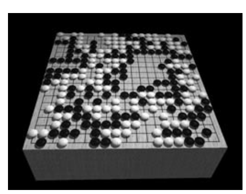
OK

- Survival – 50 pts for everyone that died before it
- Last Survivor – 10 pts for every robot in battle
- Bullet Damage – 1 pt for each pt of inflicted damage
- Bullet Damage Bonus – 20% kill bonus of all the damage it did
- Ram Damage – 2 pts for every pt of ram damage
- Ram Damage Bonus – 30% kill bonus of all ram damage it did

Robocode Rules

- Evironment loop
  - Robot code executed, time incremented, bullets move, robots move, robots scan
- Bullets
  - Bullet damage = 4*firepower (plus 2*(firepower-1) if firepower > 1)
  - Bullet speed = 20 – 3*firepower
  - Energy returned on hit = 3 * firepower
- Robot Collision = .6 damage each
- Advanced Robots take Wall Collision penalty

Learning Go with TD(λ)



Todd Detwiler
CSE 592
Winter 2003

What is GO?

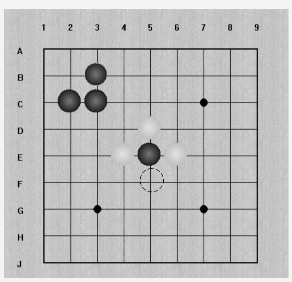
- One of the oldest and most popular board games in the world (around 4000 yrs old)
- A game of territory acquisition
- Deterministic, perfect-information, zero-sum, 2 player strategy game
- A “grand challenge” in AI (Rivest 1993)

The Rules

- Players alternate placing stones on open intersections of the board (a 19x19 grid)
- Adjacent stones form groups
- Empty intersections adjacent to groups form its liberties
- A group is captured when all of its liberties are removed
- 2 passes signify the end of the game
- Ko

Captures

If white plays at the location indicated by the red circle, they will capture the black stone by removing its last liberty.



Why is Go so Hard?

- Pspace-complete
  - Average branching factor of game tree around 200
  - Size of game tree on the order of 10^{170} (compared to around 10^{50} for Chess)
  - Too large for look-ahead evaluation
- No good evaluation function for game states

TD(λ) Approach

- Learn an evaluation function
  - Use neural network as a function estimator
- Temporal credit assignment

$$w_{t+1} - w_t = \alpha(Y_{t+1} - Y_t)\sum_{k=1}^{t} \lambda^{t-k} \nabla_w Y_k$$

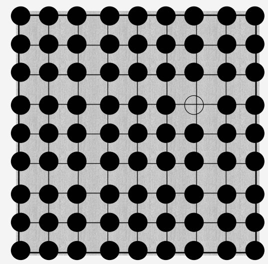
The Pieces that I Started With

- OpenGo 5.1 beta
  - A set of pre-written Go objects as well as an environment for playing in
    - Very buggy, not as useful as I initially suspected
- Nonlinear TD/Backprop pseudo C-code
  - Allen Bonde Jr. and Richard Sutton
  - I have extended this to be an actual C++ object

Player Design

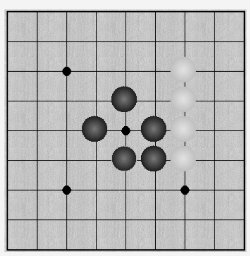
- Like TD-Gammon, games (state sequences) are generated by pitting my Go player against itself
- Unlike TD-Gammon, I am using off-line learning
- Initially give player rules only, no strategy
- Later augmented with one rudimentary extension to reduce plies/game

One Problem



The Extension

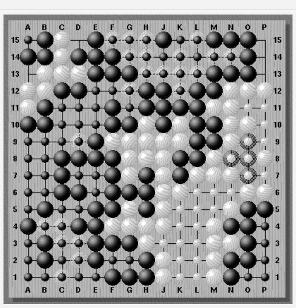
- Don’t fill in simple, size 1 eyes
- Super Ko



Current Status

- Player
  - Identifies all legal moves
  - Plays against itself
  - Detects win
  - Black tracks game states for learning
- TD(λ) network is implemented, but not fully tested
  - Currently testing load/save functionality
- Learning has not yet been achieved

Questions?



Letter Recognition by Using Multi-Layer Neural Network

Meng Tat Fong
03/13/2003

Problem Domain

- Create a classifier to identity the 26 capital letters in the English Alphabet
- Extensible
- Create electronic document from scanned documents, newspapers, etc.

Data Set

- David Slate donated to UCI machine learning repository
- 20,000 samples
- letter images from black-and-white displays
- 20 different fonts
- randomly distorted (all unique samples)

Data Set

- 16 integer attributes
- Normalize to 0.0 – 1.0
- 26 output classes (A-Z)
- 750-800 samples each

- 2,4,4,3,2,7,8,2,9,11,7,7,1,8,5,6,Z
- 4,7,5,5,5,5,9,6,4,8,7,9,2,9,7,10,P

Backgrounds

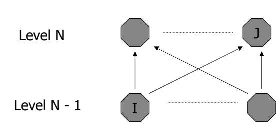
- Not using any existing Machine Learning libraries
- Java

Algorithms

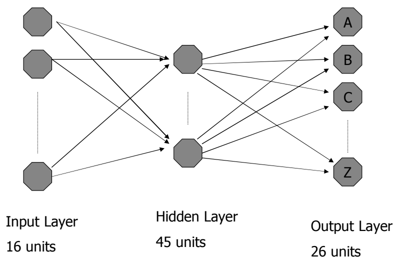
- Separate the sample data set into two sets (~16,000 and ~4,000)
- Network is trained and then verified
- Stochastic gradient descent version of the BackPropagation algorithm
- Unit weight is updated after each sample
- Sigmoid Units to learn non-linear functions

Algorithms

- $W_{ji} = W_{ji} + \Delta W_{ji}$
- $\Delta W_{ji} = \mu E_j X_{ji}$
- Based on the idea that each unit is partially responsible for the error of its parent.



Network Topology



Improvements

- Momentum -- nth weight update partially depending on the previous update
  - $\Delta W_{ji}(N) = \mu E_j X_{ji} + \alpha \Delta W_{ji}(N-1)$
  - Help to escape local minima
  - Move along flat region during the search
  - Increase my network accuracy by 2.2%
  - Momentum 0.58 (75.1% to 77.3%)

Improvements

- Learn from mistakes
  - Train the network with all the training samples once
  - Feed the same samples to train the network, but only use incorrectly classify samples
  - Give the network chances to correct its mistakes
  - Accuracy improved from 72.0% to 77.3%

Improvements

- Ensemble
  - Use multiple networks to perform classification
  - Each network will predict an outcome and the majority will win
  - Improved the accuracy to >80%

Results

- Slate’s Adaptive Classifiers (1990)-- ~80%
- Weka’s J48 Decision Tree -- 87.75%
- Weka’s Naive network -- 64.23%
- Weka’s neural network – no result after 10 hours
- My network – up to 85%, alpha 0.60, momentum 0.58, hidden layer 1, 45 hidden units, >300,000 training examples

Results

- Start small
- Build a small network to solve a simple problem. (no hidden unit, one output class, trivial problem domain)
- Add more output classes
- Add more hidden layers

Results

- Hard to create a generic neural network
- Need to adjust the network topology, learning rates, momentum, etc
- Once you have a working network, it will perform very well

Thank You!

Random Sampling in Mixtures of Bayes Nets

Manish Goyal

Basic Idea

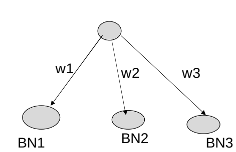
- Bayesian networks serve as compact representations of data
- The data is represented in terms of conditional distributions
- Draw random samples from these conditional distributions to generate data which can then be used for a variety of purposes

Base system

- Random sampling has been applied to a problem relating to recognition of single characters
- The base system consists of a model for each character

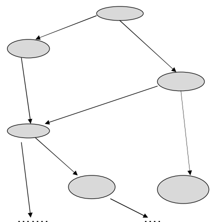
Explanation of Base System

- The model for each character consists of a mixture of Bayes nets(BN_1,….BN_n) with weighting factors w1,….wn
- Models have been trained for each of the 99 supported characters
- The training set consists of approximately 200 samples of each character



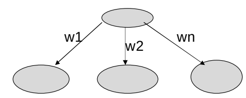
Explanation of Bayesian Nets within each model

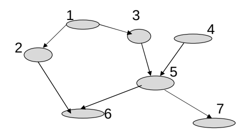
- For each handwritten character we extract 64 features
- These features are a mixture of
  - Fourier Transforms
  - OCR features
  - Contour Features
- For the purpose of this talk the exact nature of these features is not important
- Each of these features is represented as a node in a graph. Hence given that there are 64 features, there are 64 nodes in each Bayesian Network
- Each node is represented in terms of the conditional distribution. ie. P(node/all the parents of the node)



Method of sampling

- First randomly select which Bayesian Network you will select. The bayesian networks are selected with Probability(w1,….wn)
- Once the Bayesian network is selected we now need to generate observations from the network
- For this we need to traverse the graph in order. For ex. In the figure ,the correct order of traversal would be 1,2,…n
- Each node is specified in terms of its conditional mean and covariance given by Mean=M=$\mu_c+\sum_p(X_p-\mu_p)$
- Covariance=C=σ_c
- As you traverse the graph, generate the observation for the particular node by sampling from a Gaussian distribution with Mean=M and Covariance=C
- Once the observations of all the parents are known, the conditional mean can be computed for that node and hence an observation can be made for that node
- Do this for all the nodes
- Iterate through this generating as many samples as are required.





Verification

- Use the generated data to train a feed forward neural network (fully connected,1 hidden layer)
- Compare the error rate using the generated data to a net trained using original data
- See if these two error rates are comparable

Results

- Original training set contains approx 200 samples per code point
- Generated 200 and 500 samples for each code point using the random sampling method
- Test set used consists of 17000 samples

	Error on test set
NN trained used original data	20.32 %
NN trained using generated data(200 samp/code pt)	24.30 %
NN trained using generated data(500 samp/code pt)	23.97 %

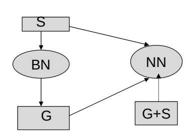
Results contd.

- The previous results were all when we were sampling from a distribution with mean=M and covariance=C
- We can increase or decrease the randomness of the generated data by using a covariance given by h*C where h is a heuristic
- Different nets have been trained for different values of the heuristic factor
- As can be seen h=1 gives the best result (as would be expected theoretically).
- Samples generated per code point=200

	Error on test set
NN trained using h=1	24.30 %
NN trained using h=0.1	41.96 %
NN trained using h=2	24.45 %

Pipe dream

- Rather than using the generated data separately, could we use it to supplement the original training data ? If used in this manner will we be able to improve the base accuracy of the neural network ?



	Error on test set
NN trained using original data	20.32 %
NN trained using 200 generated samples per code pt +original data	20.8 %
NN trained using 200 generated samples per code pt +original data	22.26 %

Conclusions

- Random sampling can be used to generate the original data
- Classifiers trained on this synthesized data have accuracy close to that obtained by using the original data

Possible uses

- Font generation
- Compact representation of data
- Other uses ?

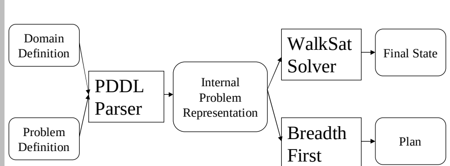
MultiSat – A PDDL Problem Solver

CS 592 Project
Rui Jiang

What is MultiSat

- A SAT solver that accepts PDDL files as input
- Supports STRIPS and part of numeric (multivalued) functions
- Currently has WalkSat and Breadth First Search implemented
- Output plan steps and final state

WorkFlow



PDDL Revisited

- Domain Definition:
  - Predicates
  - Functions (Multivalued)
  - Actions
    - Precondition
    - Effects
- Problem Definition:
  - Objects
  - Initial State
  - Goal

Function in PDDL

- Actually represents a value of an object (or objects).
- Predicate can be viewed as a function that has only true/false value.

Example of Function

Queens Problem Domain

```
(define (domain queens)
   (:requirements :strips :equality :fluents :negative-preconditions)
   (:functions (position ?row) (positionmax) )
   …
   (:action moveright
      :parameters ( ?row)
      :precondition (< (position ?row) (positionmax))
      :effect (and (increase (position ?row) 1))
   )
)
```

Example of Function

- Queens Problem

```
(define (problem queensprob4)
 (:domain queens)
 (:objects  q0 q1 q2 q3)
 (:init
   (= (position q0) 0)
   (= (position q1) 1)
   (= (position q2) 2)
   (= (position q3) 3)
   (= positionmax 3)
 )
 (:goal (and
     (not (= (position q0) (position q1)))
     (not (= (position q0) (position q2)))
     (not (= (position q0) (position q3)))
     (not (= (+ (position q0) 1) (position q1)))
     (not (= (- (position q0) 1) (position q1)))
     (not (= (+ (position q0) 2) (position q2))) …)) )
```

State in MultiSat

- Collection of predicate and function values
- A state in Queens problem:

```
position q0  4
position q1  2
position q2  0
position q3  5
position q4  7
position q5  1
position q6  3
position q7  6
positionmax  7
```

How does WalkSat work?

- Start with the problem initial state.
- While not solved
  - Create an empty queue
  - For each action, generate all possible combination of parameters
    - Evaluate precondition against the current state. If true, do the action and evaluate how many propositions in the goal valid. If all propositions in the goal are valid, the problem is solved and we exit. Otherwise put this action and its result state into the queue.
  - Select the action and parameters that will have the largest number of valid propositions in the goal. With a small probability, randomly select any action from the queue. Replace the current state.

Breadth First Search in MultiSat

- Just the usual breadth first search
- With Dynamic Programming – exclude similar state in the search

Example output – Sokoban

..\bin\multisat -o sokoban\sokoban.pddl -f sokoban\sokoban2.pddl -bf -notree
…
Actions:
1: push_left p33 p32 p31 ==> ValidCount 0
2: push_down p21 p31 p41 ==> ValidCount 0
3: move_up p31 p21 ==> ValidCount 0
4: move_up p21 p11 ==> ValidCount 0
5: push_up p31 p21 p11 ==> ValidCount 0
6: push_up p21 p11 p01 ==> ValidCount 2

Time used: 2.824 seconds

Example output - Queens

F:\CS592\project\test>..\bin\multisat -o queens\queensdomain4.pddl -f queens\queens4.pddl -notree
…
Problem is not solved yet. Let me try try...
Goal's maximum propositions: 18
1: moveleft1 q2 ==> ValidCount 14
2: moveright1 q1 ==> ValidCount 16
3: moveright2 q0 ==> ValidCount 16
4: moveleft2 q1 ==> ValidCount 17
5: moveright2 q2 ==> ValidCount 17
6: moveleft2 q3 ==> ValidCount 18
Haha, we have solved the problem! Final state:
position q0 2
position q1 0
position q2 3
position q3 1
positionmax 3
Time used: 0.000 seconds
Search steps: 6

Performance

- Examples are run on a Dell Inspiron 4150 with 1.8 GHz CPU, 512 MB
- Sokoban problem is run using breadth first search
- Queens problem and quasigroup problem are run with WalkSat

Performance - Sokoban

Here are the 3 sokoban problems from homework 1

	Time used (in seconds)	
	MultiSat	BlackBox
Problem 1 (33 steps)	7.801	17.78
Problem 2 (6 steps)	2.794	0.05
Problem 3 (16 steps)	1.932	0.17

Performance - Queens

- Time Used (seconds)

Random Factor	0.01	0.05	0.1
20 queens	17	20	28
25 queens	48	60	73
30 queens	121	139	180

Performance – Quasigroup with Holes

Problem	Time Used(seconds)
9X9, 20 holes	2.2
10X10, 30 holes	12.7
11X11, 40 holes	55
12X12, 40 holes	90

Hierarchical Text Classification and the Open Directory Project

Will Kallander
CSE592

Series of directories and flat files:

Project Goal

- Use automated methods of hierarchical text classification to facilitate editing.

Use Cases

- Editor is not knowledgeable WRT the placement of a site that has been incorrectly submitted to a category.
- Automated QC – Alleviation of the “Bait and Switch” attack.
- E.g.: As in the case of Adult content in Kids_and_Teens

Approach

- Recreate hierarchical structure at every node.
- Classifiers for all internal nodes.
- Cascade classifications in RT
- N-ary classifications
- Binary classifier:
  - Adult or Non-Adult content?
  - Use same data as hierarchical approach

Feature Selection

- Use data from ODP itself as definition for classifiers:
  - Human generated – contains intelligence about ontology
  - Not as noisy as web data
  - Much smaller than web data
  - Faster – crawling is …S L O W…
  - RDF (type of XML) is easy to parse

The Guts

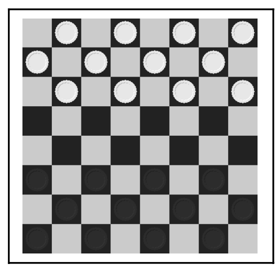
- Perl approach:
  - Rolled my own.
  - Ken Williams' AI::Categorizer module
  - CGI wrapper around C command-line front end to libbow
- C command line:
  - Andrew McCallum's libbow
  - Rainbow – front end interface for indexing
  - Use undocumented (and/or unstable) features

Reinforcement Learning

Playing Checkers

- Machine plays against itself.
- No prior knowledge on strategy.
- Uses a neural network with a hidden layer.
- Reward wins and back-propagate weights.
- Uses TD-λ propagation.

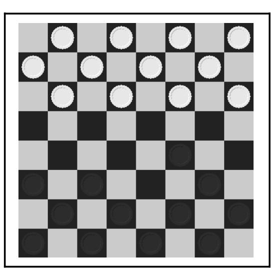
The Game



Moves diagonally forward

Red moves first

The Game

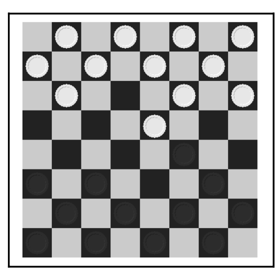


Moves diagonally forward

Red moves first

Followed by white

The Game



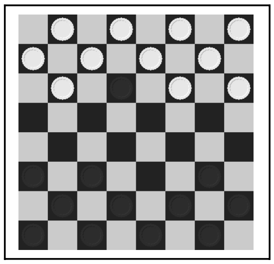
Moves diagonally forward

Red moves first

Followed by white

Captures by jumping over to empty

The Game



Moves diagonally forward

Red moves first

Followed by white

Captures by jumping over to empty

Main components

- Trainer - trains using TD-λ
  - The weights represent knowledge
  - Weights can be serialized
  - The trained net is used as player
- Player – plays with opponent algorithm

Trainer

- A neural network
- Initially randomized weights
- $\Delta w_t = \alpha(P^{t+1} - P^t) \sum_{k=1}^{t} \lambda^{t-k} \blacktriangledown_w P_k$
- Inputs – state of squares, number of discs
- Chooses move that maximizes net output
- Updates weights using change in output

Input representation

- Boolean inputs preferred vs Multivariate for reinforcement learning
- Total of 154 inputs
  - 4 inputs per square (2 – color, 2 – type of piece)
  - 8 inputs per player representing piece advantage
  - 2 inputs for who started the game
  - 2 inputs for who the current player is
  - 6 inputs for the number of moves

Strategies

- Randomization to avoid local minima
  - Randomly pick among the best moves
  - With a low probability pick a completely random move
  - Increase above probability with the number of moves
- Evaluate the next move using lookahead

Strategies …

- Breaking ties based on piece advantage
  - 3 * Man = = 2 * King
  - Punishing the player with considerable piece advantage
- Training with end games to speed up learning

Player

- GUI that accepts 2 player engines
- Play smart Vs trained player
- Smart player uses mini-max algorithm with some set of features

Lessons learnt

- Initial weights play crucial role
- Use learning parameters that have been known to work
- Weight update is easy to get wrong
- Co-evaluation techniques are not very useful
- The input representation matters

Acknowledgements

- Martin Fierz - checkerboard program
- Rich Sutton – pseudo code for TD-λ
- Cliff Kotnik – pointers into SNNS & TD-λ
- SNNS – initial experimentation

Algorithmic Composition & Artificial Intelligence

By Brian McNaboe

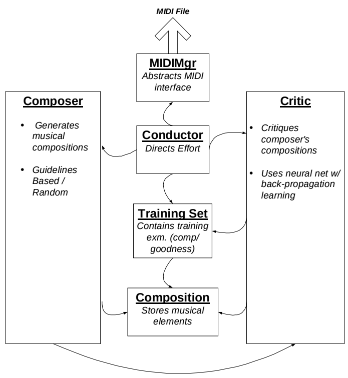
Outline

⌘Objective
⌘Approach
⌘Results
⌘Examples/Demo
⌘References

Objective

⌘Write a program that can generate "pleasant" sounding harmonized melodies autonomously.

DISCLAIMER: I do not consider myself a musician, nor do I have any formal training in music theory.



Approach - Composer

- ⌘Uses guidelines from music theory to limit state space.
- ⌘Randomly chooses chords and melody notes w/ in bounds.
- ⌘Surprisingly good results w/ few simple constraints.

Approach - Critic

- ⌘2-layer feedforward neural net of sigmoid threshold units.
- ⌘Configurable # of hidden units.
- ⌘Configurable between full and stochastic gradient decent back-propagation learning.

Approach - Critic (cont.)

- ⌘Back-propagation loop termination based on combo. of max_iters & max_acceptable % weight change (more on this later).
- ⌘Network inputs composed of 14 numerical quantifications of composition
  - ⍁total number of notes
  - ⍁note/chord tension
  - ⍁etc.

Results

- ⌘Rules based approach alone worked better than expected.
- ⌘So far, critic has been trained to critique w/ up to 80% accuracy for single training set.
- ⌘However, not enough training to successfully generalize yet (best case so far 60% train/ 60% validation).

Results (cont.)

- ⌘Still tweaking critic parameters
  - ⍁Loop termination criteria
  - ⍁Learning rate
  - ⍁Number of hidden units
- ⌘Haven't found magic formula yet...

Examples & Demo

References

⌘Mitchell, *Machine Learning.*

⌘Widmer, *Qualitative Perception Modeling and Intelligent Musical Learning.*

⌘Jacob, *Algorithmic Composition as a Model of Creativity*.

⌘Cope, *Computer Modeling of Musical Intelligence in EMI.*

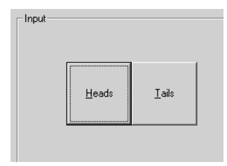
⌘Various books on music theory.

Player Move Prediction

- 3 games:
  - Penny Matching
  - Rock Paper Scissors
  - Position Tracking
- N-Gram Method
- Sequential Prediction Method
- Note: Random = Unpredictable

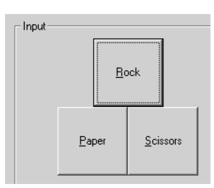
The Games

- Penny Matching
  - Computer tries to predict your choice
  - Game introduced in SEER paper



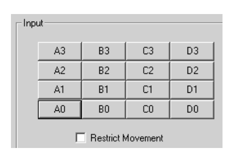
The Games (cont.)

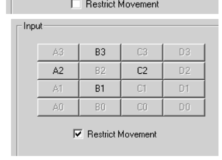
- Rock Paper Scissors
  - Traditional game
  - Tie is possible
  - Human randomness more difficult



The Games (cont.)

- Position Tracking
  - 16 choices
  - Movement representation
  - Option to restrict movement





N-Gram Method

- From speech recognition research; shown in class:
  - Unigram, Bigram, Trigram
- General case: N-Gram
- Tally occurrences of permutations of N moves.
- Example of N-Gram(4):
  - Player's last 3 moves: H-T-T
  - H-T-T occurred 4 times in past followed by T
  - H-T-T occurred 2 times in past followed by H
  - Computer predicts player's move will be T

N-Gram Results

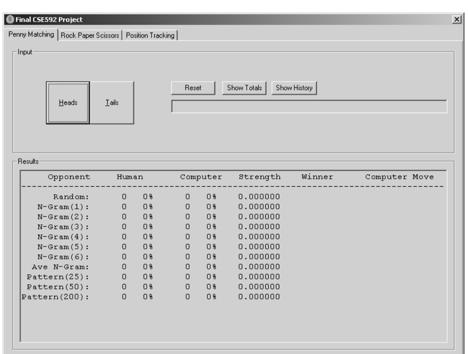
- Tested games with N from 1 to 6
- Preliminary Testing:
  - Penny Matching best with 4
  - Rock Paper Scissors best with 3 (2 & 4 close)
  - Positional Tracking best with 2
- Experimented with summing all N-Grams, with each weighted by its confidence
  - Generally performs in top 25%
  - Avoids picking a specific N-Gram that could underperform

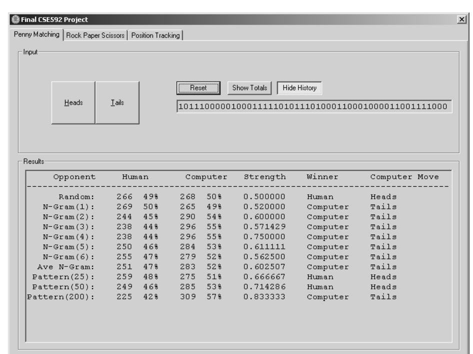
Sequential Prediction Method

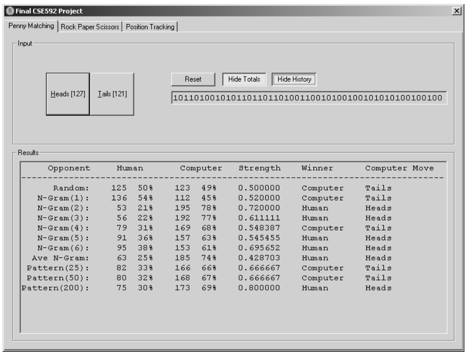
- Search for longest substring that matches tail of sequence.
- Optimization
  - For each move, maintain list of positions of occurrences
  - Generate match size for list & select longest
  - Runs in O(N) vs. O(N²)

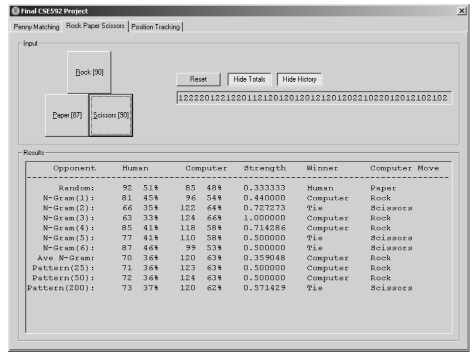
Sequential Prediction Results

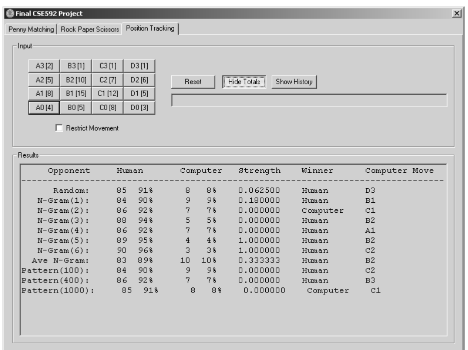
- Good performance in general
  - Consistently over 50%
  - Somewhat worse than best-performing N-Grams
- Outperforms N-Grams on restricted movement position tracking

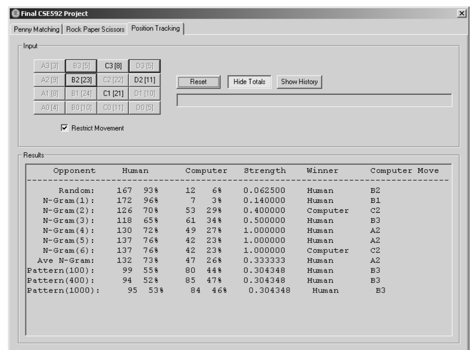












.Net Terrarium Animal as a Reactive Agent

Jack Richins
CSE 592

Motivation

- Creatures need to move to a plant or animal to eat.
- Sometimes, they get blocked by other creatures or other plants.
- Animals only get 2 to 5 milliseconds a turn
- Best First Search was too slow
- Community Astar implementation faster, but still failed to find path sometimes.

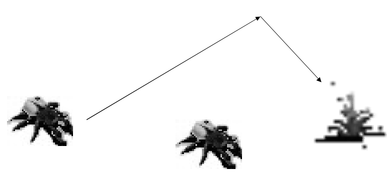
Reactive or Simple Reflex Agent

- Reactive agents react to input from sensors with simple actions based on simple rules.
- Sensor: Output of scan() is list of all creatures, plant or animal, in the world.
- Rule: Try different angles off of direct path until un-obstructed path is found

Example

Direct path is obstructed

Example



Method of Evaluation

- Evaluated by success in controlled Terrarium world.
  - not hooked up to network where creatures from other Terrarium's can be transported in or out
- Tested exactly 2 animals at a time
- Tested until population showed a clear winner and loser, or 45 minutes.

Results

	Population	Births
	Start - 12:57	
plant	**11**	**11**
greedymoveherb	**10**	**10**
minobstaclesherb	**10**	**10**
	39 Minutes	
plant	**58**	**232**
greedymoveherb	**13**	**280**
minobstaclesherb	**102**	**434**

Example

Exclude Moving Creatures from Obstacles List

	Population	Births
plant	10	10
minobstaclesherb	10	10
excludemovers	10	10
	47 minutes	
plant	19	133
minobstaclesherb	4	201
excludemovers	16	259

Best Reactive Agent versus A*

	Population	Births
astar	10	10
excludemovers	10	10
plant	10	10
	42 minutes	
astar	22	180
excludemovers	20	305
plant	21	155

Conclusion

- Reactive agent show considerable improvement over no reaction at all
- For this problem space, comparable to A* performance.
- For such a simple implementation compared to A*, this seems impressive to me.

Bayesian Inference in Double Six Dominos

Carlos Garcia Jurado Suarez
03/13/2003

Outline

- Game description
- Approach
- Performance measurements
- Remaining and future work

Game Description

Game Description …

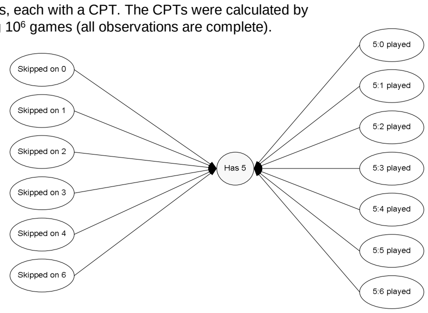
- There are 2 teams of 2 players each. Team mates sit across from each other in a squared table.
- There are 7 numbers and 28 dominos (from 0:0 to 6:6)
- The goal is for either of the team players to get rid of all his/her dominos (before the other team does).
- A domino can be played by matching the number of dots with the ones in either end of the game.
- When somebody finishes the team is awarded a number of points equal to the sum of the points that the other team had in their remaining dominos.
- The first team to reach 100 points wins the match.

Approach

- To win in dominos the basic strategy is: avoid skipping turns and force your opponents to skip.
- Dominos should be played such that the probability of the team members to skip is low and the probability of opponents to skip is high.
- We need a way to infer such probabilities

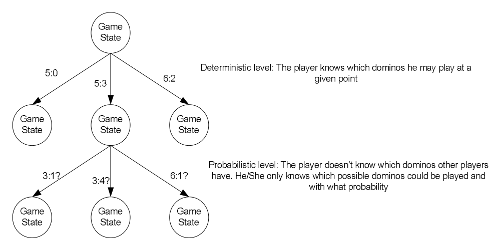
Approach … Bayesian net

7 networks, each with a CPT. The CPTs were calculated by simulating 10^6 games (all observations are complete). Example:



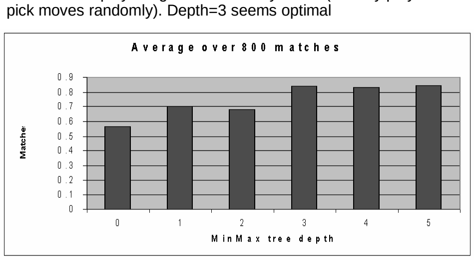
Approach … MinMax

- MinMax is used to select a move. Each game state is evaluated based on the strategy previously described.



Performance measurements

- 800 matches played against a "Dummy" team (dummy players pick moves randomly). Depth=3 seems optimal



Remaining and future work / QA

- Further improvements may include:
  - Better approximations to the probability of non-deterministic moves.
  - Implementing Alfa-Beta pruning
  - Learning the utility function (genetic programming or neural network?)
- Questions?

Studying the Effects of Parallelism on Current Planners

James Welle

Overview

- Study several different state of the art planners (FF, IPP, and Blackbox) on variations of the Sokoban world, where the amount of parallelism can be controlled by having different numbers of Sokoban

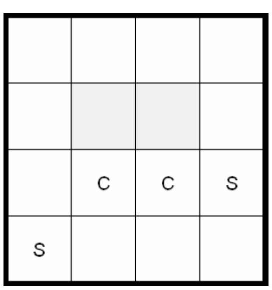
Purpose and Goals

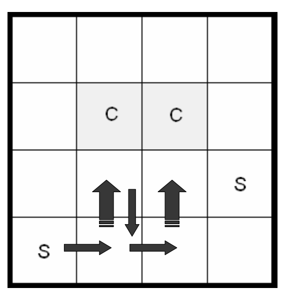
- How will these planners be affected by introducing multiple Sokobans into the problem?
- Will adding resource bounds to the Sokoban domain affect these planners?
- How will these planners scale as the number of Sokobans grows.

Measuring planners

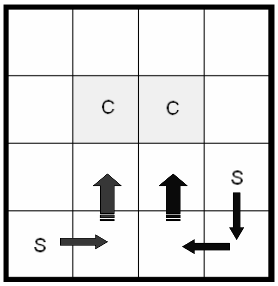
- Speed in Plan Creation
- Plan Quality
  - Plan Length
  - Time
  - Resources (fuel, energy, $, etc.)

What's the best plan?





- Plan Length = 5
- Time Steps = 5



- Plan Length = 5
- Time Steps = 3

Resource Bounded Logistics

(:action MOVE-SOKOBAN
:parameters
(… ?r)
:precondition
(and …
(resource ?r)
(can-use ?sokoban ?r))
:effect
(and …
(not (resource ?r)))

Planners Considered

- FF
  - FF is a forward chaining heuristic state space planner.
  - Generate a heuristic by generating an explicit solution to a relaxed problem (using GRAPHPLAN) and using the number of actions in the relaxed solutions is used as a goal distance estimate.
  - Use *enforced hill-climbing*: uses breadth first search to find a *strictly better*, possibly indirect, successor.
  - If local search fails, then skip everything done so far and switch to a complete best-first algorithm that simply expands *all* search nodes by increasing order of goal distance evaluation.

Planners Considered (cont.)

- IPP
  - Based on GRAPHPLAN – builds the planning graph starting from initial facts
  - RIFO – Removing Irrelevant Facts and Operators
  - RIFO tries to determine such irrelevant information (ground operators and initial facts) using a "backchaining" process and removes them from the planning task.
  - Depending on the heuristic and union method chosen, different kinds of "possibility sets" of relevant objects and facts are created. These sets can be used in different ways to decide over relevance or irrelevance of ground operators and initial facts

Planners Considered (cont.)

- BLACKBOX
  - Uses GRAPHLAN to create satisfiability problems from planning problems
  - Can invoke a number of different satisfiability solvers on the problem
    - WALKSAT, SATZ, etc.
    - I focused specifically on the CHAFF solver

Approach

- Run the planners on the modified Sokoban domain and compare results
- Introduce resource bounds into the domains from AIPS 2002 and compare results
- Experiment with how the planners scale as the number of Sokobans grows

Results

- IPP and BLACKBOX, much better than FF on parallelism
  - Expected, as they have a sense of time and FF does not
- Introducing resource bounds into AIPS domains
- Experimenting with scalability

Mining the Weather

Using AI Techniques to Make Weather Predictions

Reid Wilkes
CSE 592
University of Washington
February 13, 2003

Motivation

- Prediction normally done by modeling physical processes.
- Even powerful computer models are much less than perfect, and require a deep understand of the science of Meteorology.
- Can machine learning be used to identify patterns in historical data and make predictions as well as the computer models?
- Chance to experiment with various machine learning techniques.

Problem Statement

- Try to use machine learning methods to analyze historical data and make predictions of what the weather conditions will be at a given location at some time in the future.
- In practice, focused on predicting the conditions in Seattle (Boeing Field) 6 or 12 hours in the future.
- Output of system is probability for each possible condition (Rain, Sun, Cloudy, etc…)

Approach – Collecting Data

- Picked 12 Locations Across State of Washington
  - Bellingham
  - Boeing Field (Renton)
  - Everett
  - Forks
  - Hoquiam
  - Olympia
  - Port Angeles
  - Shelton
  - Stampede Pass
  - Vancouver (WA)
  - Wenatchee
  - Yakima
- The 1st of many informed but arbitrary decisions!

Approach – Collecting Data

- Collected 6 data points for each location.
  - Current Conditions (Rain, Cloudy, etc…)
  - Temperature
  - Humidity
  - Barometric Pressure
  - Wind Speed
  - Wind Direction
- Data taken every hour from http://iwin.nws.noaa.gov/iwin/wa/hourly.html
- Small utility parses HTML page every hour and inserts new readings into SQL Server database.
- Collected data starting on Feb 10.

Approach – Preparing Data

- Once data was collected, it had to be worked into a usable form.
- To make life easier, data was discretized.
  - Temperature, Humidity, Pressure were divided into 5 – unit buckets.
  - Conditions are aggregated into 9 condition types.
    - Sunny/Clear
    - Cloudy
    - Partly Cloudy
    - Rain
    - Freezing Rain
    - Fog
    - Snow
    - Mix snow/rain
    - Hail

Approach – Preparing Data

- In addition to absolute conditions, condition changes were also used.
  - DTemperature
  - DWind
  - DHumidity
  - DPressure
- There was no DConditions value – only current conditions considered.
- Wind Speed and Wind Direction were discretized together in a way that takes into account both the change in Speed and Direction as well as the current state (56 total possible values).
- The systems written were designed to take a parameter which determines how big an interval over which to calculate the differentials.

First Analysis Method – Naïve Bayes

- Naïve Bayes seemed to be a good first shot at predicting.
  - Deals with probabilities, which is really what we'd like the system to output in the end.
  - Not too hard to be naïve enough to claim that the all of the data collected at one point in time is conditionally independent given the conditions in Seattle in the future.
  - It's much, much harder to try to understand conditional dependencies between the data points, if we were to try a more structured Bayesian Network.

Naïve Bayes - Implementation

- C# application uses stored procedures in SQL Server to do some of the counting, and uses ADO.NET data sets in memory to do the rest of the counting.
- All floating point computations done in the C# app – SQL Server returns nothing but integers.
- Basically, build a giant SQL temporary table that has all the data we need already discretized and work from there.

Naïve Bayes – Better Implementation

- Keep tables around with counts of different values and update them when data is inserted into master table.
- Helps offset the cost of the counting at prediction time.
- Would be absolutely necessary with more historical data.

Naïve Bayes - Results

- At first, didn't perform so well… ~70% for six hour forecast with 6 hour interval.
- Problem was that the artificial sample (used to prevent 0 terms in product) was WAY too high (100).
- When artificial sample size was reduced to 1, accuracy shot up to ~85% for 6 hour and ~83% for 12 hour forecasts!
- Accuracy calculated as number of the most likely condition the system predicts is correct.

Naïve Bayes Net Results

- Calculated accuracy over same data set used to make predictions!!! (there just wasn't enough data to go around)

Other Possible Approaches

- Experimented with hybrid Neural net – Probabilistic inference method.
  - Treat each location as perceptron. Weight inputs (P(Cond | input)) and aggregate predictions.
  - Have 1 perceptron that aggregates input from each location. Apply weights to each input.
  - Inputs/Outputs from each perceptron was a vector of probabilities for each possible conditions.
  - Without training, it ALWAYS forecast cloudy conditions.
  - Train using variant of stochastic gradient descent – because we are dealing with vectors the math and logic get pretty weird.
  - Actually was successful in training the network! Unfortunately in the wrong direction….

Other Possible Approaches

- True Bayesian Network.
  - Final prediction is dependant on locations.
  - Each location is dependant on the data from that location.
  - Could possibly also introduce dependencies from time of day to certain variables like temperature.

Summary

- Was it successful?
  - Naïve Bayes was more successful than expected.
  - Neural Network Idea bombed so far, but I still have hope it could yield positive results.

Bottom Line – Data was insufficient to make any conclusive statements!

Applying AI to Network Intrusion Response

Brett M. Wilson

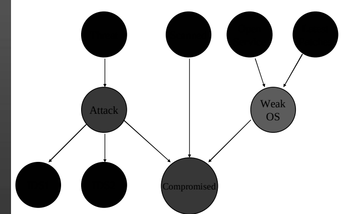
Background

- **IDS witnesses patterned or anomalous behavior and categorizes it as an attack**
- **Traceback determines the final source and destination of the network traffic involved**
- **Temporary blocking rules are inserted for immediate response**
- **Human operator fine tunes the rules and takes any other necessary precautions**

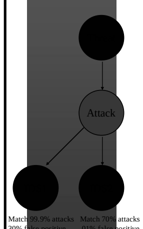
Design

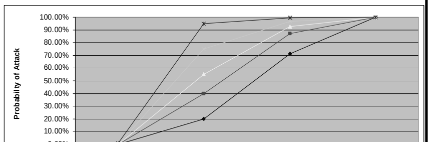
- Didn't easily fall into the realm of one tool
- Split into different tasks for different tools
  - Knowledge base for collecting data and deducing new statements from it
    - Prolog
  - Bayesian network to determine the probabilities of events in question, given evidence
    - EBayes (programmatic interface to JavaBayes)
  - Utility theory to weigh the tradeoff of letting an attack spread versus blocking off part of the network or service
    - TBD

Pretty Pictures



Pretty Graphs





	Neither	IDS1	IDS2	Both
Low	0.01%	19.99%	71.43%	99.98%
Normal	0.03%	39.98%	86.96%	99.99%
Medium	0.05%	54.98%	92.44%	100.00%
High	0.13%	74.98%	96.77%	100.00%
Extreme	0.81%	95.00%	99.48%	100.00%

Detector Alert

Future Ideas

- More inputs, more inputs, more inputs....
- Applying machine learning or game theory to predicting an adversary's next move or final goal

FLIP

First-order Logic Inference Prover

Jonathan Wray

CSE 592 Winter 2003

First-Order Logic

- Extends Propositional logic to include objects, relations, and functions.
- Similar in some respects to how humans reason.
- Predicates: *King*(*x*) ∧ *Greedy*(*x*) → *Evil*(*x*).
- Functions: *Sibling*(*Son*(*x*), *Daughter*(*x*)).
- Universal Instantiation: @*x Likes*(*x*, IceCream).
- Existential Instantiation: #*x Killed*(*x*, *Tuna*).
- Chapter 8 in R&N

Inference in FOL

- Forward Chaining
  - deductive databases, production systems
- Backward Chaining
  - logic programming (e.g. Prolog)

Both are restricted to Horn Clauses

- Resolution
  - theorem provers
  - "refutation-complete"
- Chapter 9 in R&N

Unification

- Generalized Modus Ponens:
  - For atomic sentences p_i, p_i', and q, where there is a substitution θ s.t. SUBST(θ, p_i') = SUBST(θ,q), for all i,

$$\frac{p_1', p_2', \ldots, p_n', (p_1 \wedge p_1 \wedge p_1 \rightarrow q)}{\mathrm{SUBST}(\theta, q)}$$

- Example:

  @x *King*(*x*) ∧ *Greedy*(*x*) → *Evil*(*x*).
  King(*John*).
  Greedy(*John*). {*x*/*John*} → *Evil*(*John*)

Conversion to Conjunctive Normal Form

- "Everyone who loves all animals is loved by someone."

```
@x (@y Animal(y) -> Loves(x,y)) -> (#y Loves(y,x)).
```

- Eliminate implications, move negation inwards, standardize variables, *skolemize*, drop universal qualifiers, distribute ∧ over V:

```
Animal(F:0(x:0)) | Loves((F:1(x:0)), x:0) &
!Loves(x:0, (F:0(x:0))) | Loves((F:1(x:0)), x:0)
```

- *Skolem function*: "arguments are all universally quantified variables in whose scope the existential quantifier appears" (Say what?)

Resolution Inference Rule

- Binary resolution:
  - For each pair of clauses, try to eliminate two complementary literals (where one unifies with negation of other)
  - Apply substitution to remaining literals
  - Remaining literals form the *resolvent* clause
- Factoring:
  - For each clause, try to remove redundant literals (those which unify)
  - Apply substitutions to other literals

Did Curiosity kill the Cat?

- **Knowlege Base:** **"Everyone who loves all animals is loved by someone.**
  Anyone who kills an animal is loved by no one.
  Jack loves all animals.
  Either Jack or Curiosity killed the cat, who is named Tuna."

```
@x (@y Animal(y) -> Loves(x,y)) -> (#y Loves(y,x)).
@x (#y Animal(y) & Kills(x,y)) -> (@z !Loves(z,x)).
@x Animal(x) -> Loves(%Jack, x).
Kills(%Jack, %Tuna) | Kills(%Curiosity, %Tuna).
Cat(%Tuna).
@x Cat(x) -> Animal(x).
```

- **Query:** **"Curiosity killed the Cat."**

```
Kills(%Curiosity, %Tuna).
```

Yes, and here's the proof:

```
Animal(F:0(%Jack))
    !Loves(z:5, %Jack)
        Animal(%Tuna)
            Cat(%Tuna)
            Animal(x:7) | !Cat(x:7)
        !Animal(%Tuna) | !Loves(z:5, %Jack)
            Kills(%Jack, %Tuna)
                !Kills(%Curiosity, %Tuna)
                Kills(%Curiosity, %Tuna) | Kills(%Jack, %Tuna)
            !Animal(y:4) | !Kills(x:3, y:4) | !Loves(z:5, x:3)
    Animal(F:0(x:0)) | Loves((F:1(x:0)), x:0)
!Animal(F:0(%Jack))
    !Loves(%Jack, (F:0(%Jack)))
        !Loves(z:5, %Jack)
            Animal(%Tuna)
                Cat(%Tuna)
                Animal(x:7) | !Cat(x:7)
            !Animal(%Tuna) | !Loves(z:5, %Jack)
                Kills(%Jack, %Tuna)
                    !Kills(%Curiosity, %Tuna)
                    Kills(%Curiosity, %Tuna) | Kills(%Jack, %Tuna)
                !Animal(y:4) | !Kills(x:3, y:4) | !Loves(z:5, x:3)
        Loves((F:1(x:0)), x:0) | !Loves(x:0, (F:0(x:0)))
    Loves(%Jack, x:6) | !Animal(x:6)
```

Is West a criminal?

- **Knowledge Base:**

```
@x,y,z American(x) & Weapon(y) & Sells(x,y,z) & Hostile(z) -> Criminal(x).
@x Missile(x) & Owns(%Nono, x) -> Sells(%West, x, %Nono).
@x Enemy(x, %America) -> Hostile(x).
@x Missile(x) -> Weapon(x).
Owns(%Nono, %M1).     Missile(%M1).
American(%West).      Enemy(%Nono, %America).
```

- **Query:** `Kills(%Curiosity, %Tuna).`

```
American(%West)
!American(%West)
    Weapon(%M1)
        Missile(%M1)
        Weapon(x:3) | !Missile(x:3)
    !American(%West) | !Weapon(%M1)
        Sells(%West, %M1, %Nono)
            Missile(%M1)
            Sells(%West, %M1, %Nono) | !Missile(%M1)
                Owns(%Nono, %M1)
                Sells(%West, x:1, %Nono) | !Missile(x:1) | !Owns(%Nono, x:1)
        !American(%West) | !Sells(%West, y:0, %Nono) | !Weapon(y:0)
            Hostile(%Nono)
                Enemy(%Nono, %America)
                Hostile(x:2) | !Enemy(x:2, %America)
            !American(%West) | !Hostile(z:0) | !Sells(%West, y:0, z:0) | !Weapon(y:0)
                !Criminal(%West)
                Criminal(x:0) | !American(x:0) | !Hostile(z:0) | !Sells(x:0, y:0, z:0) | !Weapon(y:0)
```

Implementation Details

- FOL parser written using the Spirit Parser Framework:
  - "Spirit is an object oriented recursive descent parser generator framework implemented using template meta-programming techniques. Expression templates allow us to approximate the syntax of Extended Backus Normal Form (EBNF) completely in C++. Parser objects are composed through operator overloading and the result is a backtracking LL(∞) parser that is capable of parsing rather ambiguous grammars."
- Conversion to CNF, Unification, etc. were implemented as recursive operations on an abstract syntax tree (AST) representation.
- The AST was sorted during resolution according to a canonical ordering designed to improve efficiency of various operations.
- *Unit Preference:* Prefer to do resolutions where one of the sentences is a single literal, or *unit clause*.
- Support for strings, including concatenation and splitting.
- Limited support for equality.

FOL Grammar

```
Identifier = lexeme_d[token_node_d[(alpha_p | '_' | '$') >> *(alnum_p | '_' | '$')]];
Constant = lexeme_d[token_node_d[ch_p('%') >> +(alnum_p | '_' | '$')]];
String = token_node_d[confix_p('\"', *c_escape_ch_p, '\"')];
Function = Identifier;
Predicate = Identifier;
Variable = Identifier;
TermList = infix_node_d[(Plus | Term) % ','];
VariableList = infix_node_d[Variable % ','];
Term = Function >> inner_node_d['(' >> TermList >> ')']
     | Constant | Variable | String | inner_node_d['[' >> Plus >> ']'];
Plus = infix_node_d[Term >> ch_p('+') >> Term];
Equal = infix_node_d[(Plus | Term) >> ch_p('=') >> (Plus | Term)];
AtomicSentence = Predicate >> inner_node_d['(' >> TermList >> ')']
               | inner_node_d['[' >> Equal >> ']']
               | inner_node_d['(' >> Universal >> ')'];
Negation = root_node_d[ch_p('!')] >> AtomicSentence | AtomicSentence | Equal;
Disjunction = infix_node_d[Negation  % '|'];
Conjunction = infix_node_d[Disjunction % '&'];
Implication = infix_node_d[Conjunction >> !("->" >> Disjunction)];
Biconditional = infix_node_d[Implication >> !("<->" >> Implication)];
Existential = root_node_d[ch_p('#')] >> VariableList >> Biconditional | Biconditional;
Universal = root_node_d[ch_p('@')] >> VariableList >> Existential | Existential;
SentenceList = infix_node_d[*(Universal >> '.')];
```

Resolution

```
while (true)
    shortcut = false
    for each (C_i, C_j in clauses)                  // clauses are already sorted
        if (size(C_i) > threshold)
            shortcut = true                         // unit preference optimization
            break
        {unified, resolvent} ← Resolve(C_i, C_j)    // binary resolution
        if unified
            if (resolvent = Ø) return true          // proof succeeded
            sort resolvents
            Factor(resolvents)                      // factoring
            new ← new + resolvent
    if (new = Ø)
        if (shortcut) ++threshold, continue
        else return false
    new ← new + clauses                             // combine clauses
    sort new
    unique new                                      // removes duplicates
    if (new = clauses)
        if (shortcut) ++threshold, continue
        else return false
    clauses ← new
```

Did Jack run up the hill?

- **Extension to FOL: support for strings as a primitive type**
- **Knowledge Base:**

```
@n Noun(n) -> NounPhrase(n).
@a,n Article(a) & Noun(n) -> NounPhrase([a + " "] + n).
@p Preposition(p) -> PrepositionalPhrase(p).
@p,n Preposition(p) & NounPhrase(n) -> PrepositionalPhrase([p + " "] + n).
@v Verb(v) -> VerbPhrase(v).
@v,p VerbPhrase(v) & PrepositionalPhrase(p) -> VerbPhrase([v + " "] + p).
@n,v NounPhrase(n) & VerbPhrase(v) -> Sentence([[n + " "] + v] + ".").
Noun("Jack").
Verb("ran").
Preposition("up").
Article("the").
Noun("hill").
```

- **Query:** `Sentence("Jack ran up the hill.").`

Jack ran up the hill.

```
PrepositionalPhrase("up the hill")
    NounPhrase("the hill")
        Noun("hill")
        NounPhrase("the " + n:1) | !Noun(n:1)
            Article("the")
            NounPhrase(a:1 + " " + n:1) | !Article(a:1) | !Noun(n:1)
    PrepositionalPhrase("up " + n:3) | !NounPhrase(n:3)
        Preposition("up")
        PrepositionalPhrase(p:3 + " " + n:3) | !NounPhrase(n:3) | !Preposition(p:3)
!PrepositionalPhrase("up the hill")
    !VerbPhrase("ran up the hill")
        !Sentence("Jack ran up the hill.")
        Sentence("Jack " + v:6 + ".") | !VerbPhrase(v:6)
            NounPhrase("Jack")
                Noun("Jack")
                NounPhrase(n:0) | !Noun(n:0)
            Sentence(n:6 + " " + v:6 + ".") | !NounPhrase(n:6) | !VerbPhrase(v:6)
    VerbPhrase("ran " + p:5) | !PrepositionalPhrase(p:5)
        VerbPhrase("ran")
            Verb("ran")
            VerbPhrase(v:4) | !Verb(v:4)
        VerbPhrase(v:5 + " " + p:5) | !PrepositionalPhrase(p:5) | !VerbPhrase(v:5)
```

Past Tense

- **Knowledge Base:**

```
@s Verb(s) -> PastTense(s, s+"ed").
@s Verb(s+"y") -> PastTense(s+"y", s+"ied").
Verb("jump").
Verb("carry").
```

- **Query:** `PastTense("jump", "jumped").`

```
PastTense("jump", "jumped")
    Verb("jump")
    PastTense(s:0, (s:0 + "ed")) | !Verb(s:0)
!PastTense("jump", "jumped")
```

- **Query:** `PastTense(“carry", “carried").`

```
PastTense("carry", "carried")
    Verb("carry")
    PastTense((s:1 + "y"), (s:1 + "ied")) | !Verb(s:1 + "y")
!PastTense("carry", "carried")
```

Further Research

- Extensions to FOL:
  - sets, lists, numbers
  - full support for equality
  - higher-order logics
- Explore other efficiency strategies
  - linear resolution
  - subsumption
- Inductive Logic Programming (ILP)
  - inverse resolution
  - application to natural language processing

Artificial Intelligence Techniques to Recover Lost USGS Datafiles

United States Geological Survey elevation datafiles are the product of publicly funded development to provide terrain elevation details in digital form. Until ~ 2000, these were available through a simple ftp tree from a site in the mid-West. Now, their access has been scattered through a thicket of for-fee products on a maze of pages belonging to "partners" of the USGS.

How can we recover them?

Goal

- Determine the shortest path from a common index page within one of the partners to the public datafiles.
- Accomplish this using naïve bayes approach, by determining from page qualities and words used whether a page is likely to lead to public datafiles.
- Use other AI techniques in the process: the use of a heuristic in a depth-first search provides the corpus of a 'happy' selection path.

World

- **The pages within which the datafiles are to be found are modern database driven pages, lots of graphics.**
  - Containing <img= … --> tags that break parser, occasional post transactions and script.
- **It takes about 5 correct jumps to get to the free datasets.**
  - Index to states to detour to counties to products to the green icon gateway
- **A heuristic to evaluate target URLs leading to free datasets is presented. This is utilized to gather a teaching corpus.**
  - Each jump heuristic is unique to its level

Learning

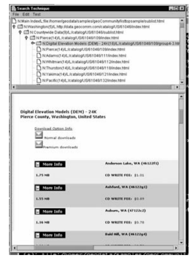
- **Each page will be characterized by particular qualities as well as the words contained.**
  - Number of links.
  - Proportion of image links to text links.
  - Link descriptions short and capitalized.
- **When a fruitful leaf is found on the tree, all the intervening nodes will be tagged productive. When at branch is found without fruitful leaves at the fifth level the search continues past it.**
  - Character and words from the productive set will be compared to the same from the unproductive set. Significant differences will considered to develop a training set of parameters.

Link Heuristics

- Look for local files first. If the context page (the page the link was found in) lacks a host specification, increment quality-index. Same again for the target.
- Directories containing sublists of state regions have the form …/nnnnn/- sublist.html. Directories containing sublists of counties have the form …/nnnnn/nnn/index.html. Increment so its noticed.
- The desired elevation files will be designated "(DEM) - 24K" in any link text. Another quality boost.
- The space examined requires directory searches about five deep to find its goal.
- A heuristic applied to get through the first gate is a small increment to the quality.
- For the subsequent gates, when a desirable condition is found, the target quality is increased more.
- For negative conditions, such as mangled URL or a file already read, the quality is taken down.

The heuristics will be turned off to gather a corpus from the whole space under the tree.

Corpus Examination Tool



An ordered list of unexamined links is presented in the top window of the central split pane window.

When a page is loaded, the tree element (ideally) turns into a branching node, listing the next links underneath.

The links are ordered according to a heuristic that increases near free dataset links.

Careful examination may reveal that the list of files shown is merely the entire ordered list shown under the last selected page, the presentation in a tree display simply coincidence. User interface rationalization is secondary to the demonstration of ai.

The page currently being evaluated is shown it the bottom window. The green arrow is the gateway to file downloads.

Exploring Tree

- **For learning the happy path, the link evaluation heuristic is turned on.**
  - After five 'green' leafs are found at a branch, the branch will pop control. Link prioritization will ensure many green hits.
- **The heuristic will be turned off to gather a corpus for the whole vocabulary. Search depth will be limited. Also, after five leaves are read at a node, the node pops control.**
- **Evaluation at each node goes like this**
  - Page (node) loaded, all links found are compared and collated into the 'play-list': A list of all links encountered through the whole run. Any links new to 'play-list' get added under the node. If the is found, all the path is marked as good.
  - The first in the play list is checked, if its parent branch has fewer than five files examined, that file is loaded. If more than five files have been examined, then unread siblings in the play-list are marked 'crowded out'.
- **After 300 files loaded, the tree is examined**

Attribute learning

- **Determine the overall chance of being either green or red (herein, means "not green"):**
  - Discover the number of green/red-path documents, and the total number of documents in general.
- **Collate all the words of all green-path nodes into the '`big-green-file`', determine its population, similarly collate all the words from the others into '`big-red-file`'**
- **For each word in the whole vocabulary, check green and red.**
  - Count the number of times it is found in `big-green-file` (or on second pass, `big-red-file`)
  - Thus determine its chances for it being in this sort of node:
    - $P(w_k \mid v_j)$ = (# of time in red/green file + 1)/(population of red/green file + population of vocabulary)

Classification - the final test

- **For a new document, get the chance of it being green from**
  - $P_{green} \prod$ P (each word in the new document being found in a green document)
  - $P_{red} \prod$ P (each word in the new document being found in a red document)
- **Classify the document according to which is greater.**

Creating a smart animal in Terrarium

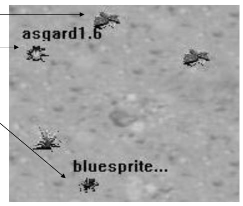
CSE 592

Yuan Zhang

Problem domain

- Plants
- Carnivores
- Herbivores

- Move
- Eat
- Attack

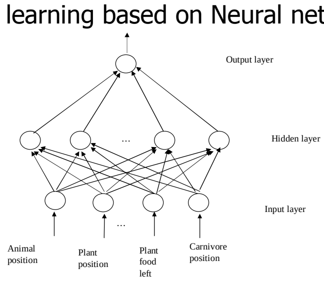


My goals

- Create a smart herbivore
- Only deal with movement
- Look for plants to eat
- Hide from Carnivores

Methodology

- TD learning based on Neural network



Methodology - con

- $w_{t+1} - w_t = \alpha(Y_{t+1} - Y_t)\sum_{k=1}^{t}\lambda^{t-k}\nabla_w Y_k$
- Input nodes: 147
- Hidden nodes: 20
- Alpha – learning rate: 0.3
- Lambda: 0.3
- Reward 1 (can eat plant)
- Punishment 0 (attacked by caniv)
- Other 0.5 (no eating in 20 ticks)

Issues

- No IO in Terrarium
  - animal starts learning from empty every time it gets loaded
  - Cannot save weights it got trained
- Event driven mode
  - Computation only happens when animal gets a tick
  - One animal gets 600 ticks in its lifetime
  - Converge after hundreds of generations
- Will be wiped off if animal thinks more than 5 seconds in one tick

Workaround

- Wrote a Terrarium simulator
- Finished training before jumping into real world

Results

- Able to learn looking for plants and hiding from carnivores (100,000 iterations)

0.553	0.623	0.715	0.745	Plant
0.547	0.584	0.644	0.710	0.752
0.384	Animal 0.484	0.619	0.672	0.693
Caniv	0.403	0.547	0.649	0.682

Lesson learned

- Terrarium is not good for algorithms that need heavy computation
- To control animal's actions and movement is much harder than thought
- Should use Q function (action, state) to search best policy

Questions?