AI & Computational Intelligence

Note Title

CS 380 Artificial Intelligence

Production Systems, Problem Solving

Search Methods

- Hill Climbing
- Backtracking
- Graph Search

→ heuristics

Knowledge Representation

- Propositional Calculus
- Predicate Calculus

PROLOG

Otter .... symbolic reasoning

Adversarial Search

- 2 player games
- minimax
- pruning
- heuristics

Machine Learning

Supervised learning
↓
you tell machine
- facts
- techniques
- solving techniques
machine learns from this

unsupervised learning
↓
machine observes things
+ reaches conclusions

generalization? - given sets of info, can machine deduce general rules at work?

"Computational Intelligence"

- neural networks
- genetic algorithms
- evolutionary computing
- fuzzy sets / logic
- particle swarm optimization
- ant colony optimization

} "microscopic" intelligence

Genetic Algorithms

1. Representation for a candidate solution to a problem (x)
2. "Fitness function" for measuring quality of a solution $f(x)$
   Goal is to find x with maximum fitness
3. Way for two candidate solutions to produce offspring: $x + y \rightarrow z$
   [offspring should retain some characteristics of parents]

4. way for a candidate solution to be mutated $x \to x'$

5. "natural selection" – how do x & y get chosen as candidates to produce offspring?

6. Population of N candidate solutions $P^0 = \{x_1 \dots x_N\}$

7. Choose pairs of candidate solutions e.g. x_5/x_{11}, x_3/x_8, ...

8. produce offspring $x_5 + x_{11} \to y_1 + y_2$

$x_3 + x_8 \to y_3 + y_4$

⋮ ⋮

9. measure quality of offspring $f(y_1), f(y_2), \dots$ $\hat{P}^0 = \{y_1 \dots y_K\}$

10. Take some subset of $P^0 + \hat{P}^0 \to P^1$ (new population)

Repeat as long as desired/needed

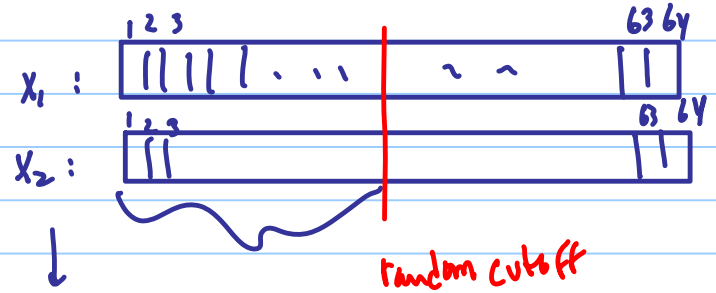
IDEA: Populations continue to get more fit

Randomness

"Survival of the fittest"

Step 3: Candidate solutions produce offspring:

e.g., fitness function $f(x) = -(x-15)^2$ achieves max when $x = 15$



64-bit floating-pt number

offspring: first part of x_1, second part of x_2 → y_1

first part of x_2, second part of x_1 → y_2

new random cutoff

y_3

y_4 ⋮

$x_1 + x_2$ can provide $2k$ offspring (k = # different cutoff pts)

Or, instead of cutoff pts, do it bit-by-bit —

for each bit of y_1, flip a random coin, and either choose the corresponding bit from x_1 or x_2

do the opposite to create y_2

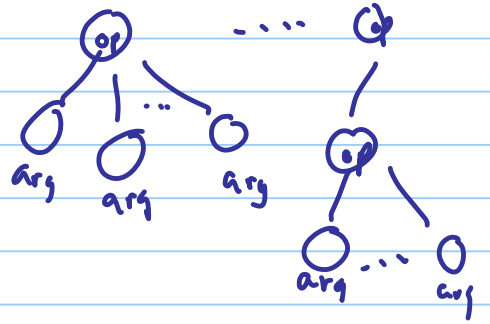
Genetic Programming —

You need a program that solves a given problem

Evolve a program!

Fitness function: how well it solves the problem

"program": LISP function

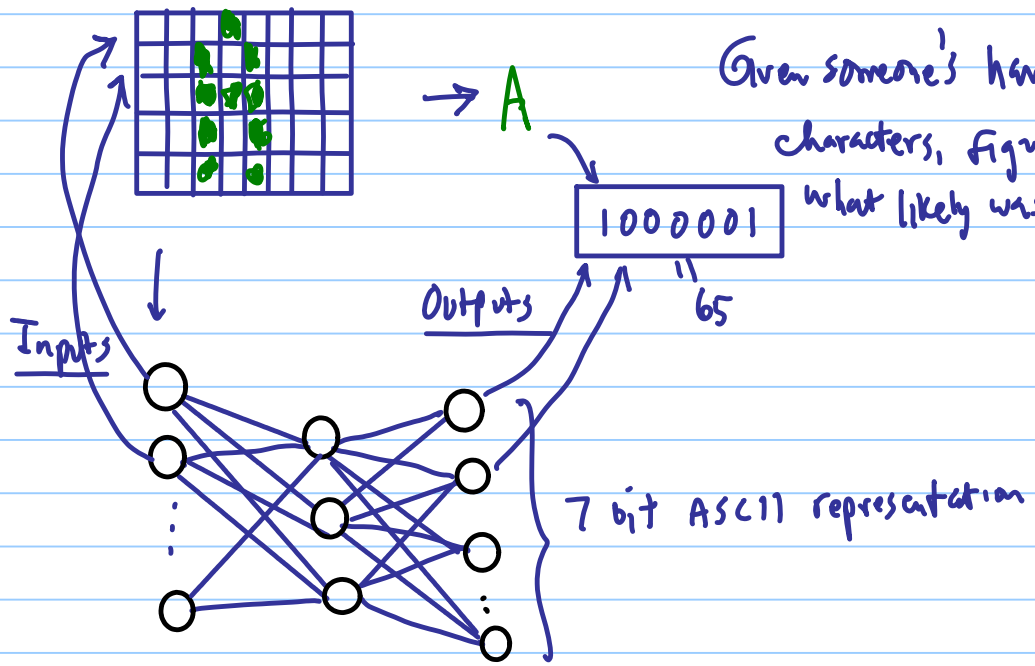


Neural Network

"recognition problem"

handwriting → imprecision

Given someone's handwritten characters, figure out what likely was meant



The problem: assign weights to the arcs in the neural net in such a way that processing each input produces the desired output

Training Set of inputs with known answers

(e.g., ask user to write the letters A, B, ... 10 times each
a, b ...
0 1 ... 9

Testing

- Use a subset of the training values to determine a set of weights for the arcs [Hopefully all produce correct answers for the training set
- Use the rest of the training values to test how well they work. [Might work OK, Might not ...

Combining Results Try this 25 times, and come up with 25 different sets of weights
Decide which are best? Decide how to combine results?

Voting : 15 votes : c

8 votes : e

1 vote : o

1 vote a

8 votes : n

8 votes : h

8 votes : m

1 vote : w

go beyond the "1 person/ 1 vote" model to weighted votes

perhaps some are right more often than others

others need to be trained longer? Discarded? etc