MILCS: A Mutual Information Learning Classifier System

Max Jiang
Rob Smith
24/11/06

Outline

- Introduction
- Proposed Work
- Current Progress
- Experiments and Results
- Future Work
- Conclusion

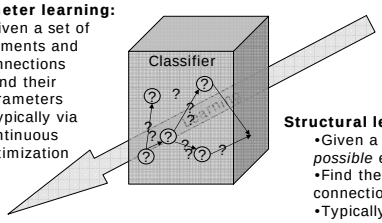
MILCS

- Borrows ideas from Cascade Correlation Networks (CCN) and integrates them into LCS
- Via Information Theory

Two Kinds of Learning:

Using terminology common in Bayesian Networks:

Parameter learning:
- Given a set of elements and connections
- Find their parameters
- Typically via continuous optimization



Structural learning:
- Given a set of *possible* elements
- Find the connections
- Typically via discrete optimization

Structural learning is associated with generalization, parsimony, and explanatory power

Typical Learning Stages in Various Schemes

Approach	*Structural Elements*	*Parameters*	*Active Structural Elements*	*Solution Determined By*	*Parameter Update*	*Structural Update*
Neural Nets	Nodes and Connections	Weights	Hidden Layer Nodes with Significant Output	Summation at Output Nodes	Backprop	None or pruning
Bayesian Nets	Nodes and Connections	Conditional Probabilities	Events with significant posterior prob.	MLE	Update from Data	Often None
Rule-based Systems	Rules	Conflict Resolution Parameters	Matched Rules	Conflict Resolution	Update from Data	Often By Hand
Decision Trees	Nodes	Splitting Thresholds	Active Node	Class at Active Node	Threshold Update	Greedy Node Addition
Function Approx. & Kernel Methods	Basis Functions	Coefficients	Basis Functions with Significant Output	Summation	Coefficient Calculation	Truncation
Learning Classifier Systems	General Rules	Various parameters	Matched Rules	Conflict Resolution	Various Methods	Genetics-Based Methods

Two kinds of learning

Another Perspective...

- Supervised learning
  - Where the learning algorithm is provided with correct outputs for given inputs (or some equivalent information from a database or teacher)
- Reinforcement learning
  - Where the algorithm is not provided with correct outputs, but only feedback on the quality of the outputs that are suggested by the algorithm itself.

LCS

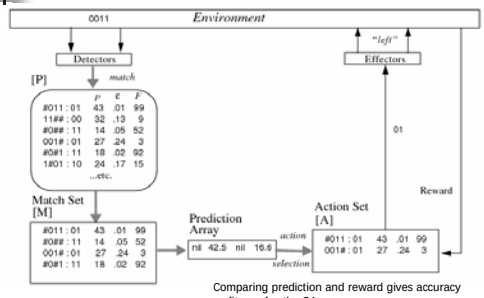
- Strength-based fitness vs. accuracy based fitness
  - A set of classifiers match a set of environmental states and these states usually have different payoff levels. Solution, sharing technique.
  - "Shared" strength-based fitness predicts payoff
    - Explanatory power problem
  - The GA cannot distinguish an accurate classifier with moderate payoff from an overly general classifier with the same payoff level on the average.

this is why Wilson developed XCS

XCS

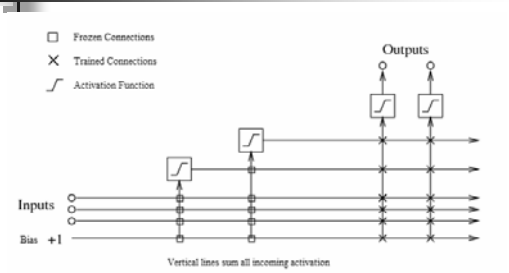
- Classifier Representation
  - Formed by an condition string {0, 1, #} and an action
  - e.g.
    - 0#01011##10 1
- Fitness Function
  - The goal of the fitness function is two-fold: maximizing the number of covered examples and minimizing the number of classification mistakes over the training set of the rule.

XCS - overview



Comparing prediction and reward gives accuracy as fitness for the GA

A NN structural learning method: Cascade Correlation Network



Cascade Correlation Procedure

- Start with no hidden layer nodes, and repeat
  - Train output weights **to minimize error**
  - Add node(s) with inputs from all existing nodes (cascaded)
  - Train new node input weights **to maximize the absolute value of correlation between the output of the new node and existing error**
  - Fully connect output of the new node
  - Continue

Why correlate new nodes to existing error?

- Because that means the "receptive field" of the new node is aligned with the area of the input space that is causing most error
- Then, adjustments of this node will be able to correct existing error (supervised learning)
- This correction is possible due to adjustment of the output weights towards the appropriate correction of existing error

An Analogy between XCS and CCN

- In XCS,
  - Rules define a set of generality (receptive fields) over (usually binary) input variables, and map them to outputs.
  - The XCS conflict resolution scheme uses XCS parameters to mediate between all the rules whose receptive fields are matched
- In CCN,
  - Hidden layer neurons parameterize receptive fields over input variables, usually with a threshold function, which effectively define a range of activation.
  - Output nodes are parameterized reactions to nodes whose receptive fields are stimulated

Why are the stats used in these structural learning schemes so different?

- We correlate to error in CC, because we can update towards the correct output directly
- In XCS, we use (accuracy, the inverse of) variance for fitness, because we don't adjust the GA-assigned outputs, and reward prediction can't mediate the value of these outputs
- This is an artifact of LCSs origins in *reinforcement learning,* rather than *supervised learning*
- *But our PSP problem is one of supervised learning*

Information Theory

- To provide a firmer theoretical basis, we are using *mutual information* rather than correlation, in our fitness function
- We believe this provides a firmer theoretical foundation

Information Theory

- Let's assume we are designing a code
- We design a code such that symbols that are expected more frequently are encoded with fewer bits
- And then let's consider the average length of a coded message (in bits) per symbol

$$H(X) = \sum_{x \in X} p(x) \log\left(\frac{1}{p(x)}\right) = -\sum_{x \in X} p(x) \log(p(x))$$

This measure is the average amount of information per symbol, measured in bits

The Kullback-Leibler Divergence and Mutual Information

- Kullback-Leibler Divergence
  - A entropy-based distance between probability distributions

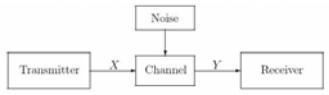
$$D(p \parallel q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}$$

- Mutual Information
  - Recall that if *x* and *y* are independent, *p(x,y)=p(x)p(y)*
  - Thus, let's take the K-L distance between *p(x)p(y)* and *p(x,y)* as a measure of the dependence of *x* and *y*

$$I(X;Y) =$$

$$D(p(x,y) \parallel p(x)p(y)) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

Shannon's Communication Theorem



- Let's imagine sending bits down a noisy channel
- Since we have noise, we have to be careful
- We'll do this by sending the bits as part of a *code*
- The code will help us limit the error we get in the output

Shannon's Theorem Tells Us...

- The Capacity of a Communication Channel *Q*
- Is the maximum information we can convey about *x* by reading *y*
- We can accomplish this by picking the best layout of receptive fields (the best encoding)
- This means maximizing the mutual information between the input signal *X* and the output signal *Y*

$$C(Q) = \max_{r} I(X;Y)$$

One more analogy

- Sensor placement
  - Consider a space where events can take place, which we are to detect
  - The events take place according to some probability distribution over the space
  - Our sensors have known response fields, based on their positioning
- Q: where do we put the sensors?
- A: **Where they maximize mutual information between the events in the space and their responses (via Shannon)**
- Therefore, we have decided to use mutual information to position our XCS receptive fields

Terms in the MI Expression

- Let's say *X* is the distribution of error in the existing system, and *Y* is the distribution of when a rule matches

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

$$= \sum_{y \in Y} \sum_{x \in X} p(x \mid y) p(y) \log \frac{p(x \mid y)}{p(x)}$$

Relevance to existing error (accuracy term)

How often the rule matches (generality term)

Comparison to overall system error (specificity term)

System Workflow

- Start with a set of random rules
- Train the rules with a random training case
  - Repeat the following on all rules
    - Remove one rule from the population
    - Update counters of that rule necessary for the calculation of its fitness based on MI of that rule's matching with the errors of all the other classifiers (note the functional similarity to CCN)
    - Add that rule back to the population
  - Do action selection based on prediction values on 'mature' rules (which have been trained by a certain number of training cases) of both matched rule set and non-matched rule set and calculate reward based on the chosen action
  - Update actions (output) and prediction value of all rules based on reward and previous actions
  - Calculate and update the fitness of action rule set based on MI counters
  - Determine and flag out all the rules of the population set that have been trained 'mature' enough to subsume with other rules and act in the non-panmictic GA
  - Subsume rules in the action rule set
  - (non-panmictic GA) Select based on these mutual-information-based fitness values (supervised learning) and subsume the offspring to the parents if possible
  - Reset counters of fitness calculation of rules in action rule set to zeros
- Repeat with the next random training case

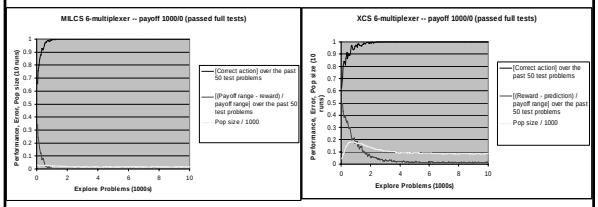
Advantage of MILCS over XCS

- Supervised learning rather than reinforcement learning
- Our system does not require the complete "model building" approach of XCS.
- It naturally exploits *default hierarchical* structure (explanatory power)
- Our classifier system is explicit about accuracy and generality.
- (and we think it's working faster and better)

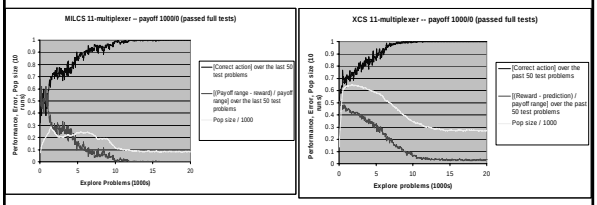
Status and Current Experiments

- We have tested the system on the 6, 11 and 20 multiplexer problems.
- We will show you a full comparison between Wilson's XCS and our MILCS on these problems.

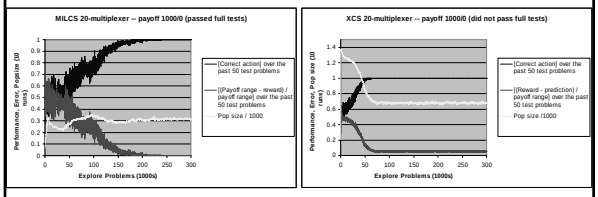
MILCS vs XCS



MILCS vs XCS



MILCS vs XCS

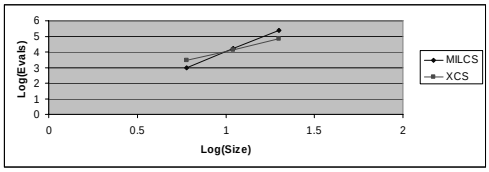


MILCS vs XCS

- MILCS
  - 6 mux < 1min
  - 11 mux < 2mins
  - 20 mux < 60mins
- XCS
  - 6 mux < 1min
  - 11 mux < 2mins
  - 20 mux < 4mins (Although XCS takes less time to do the same number of runs as MILCS does, it does not pass the full test after these runs)

MILCS vs XCS

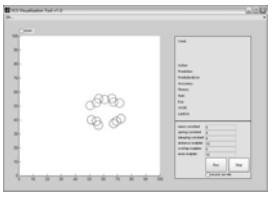
- Log-log graph



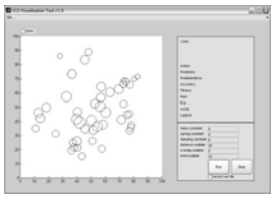
- This graph is not so accurate because XCS does not pass the full test even after 300,000 runs (whereas MILCS does)

MILCS vs XCS

MILCS 6 mux final rule set

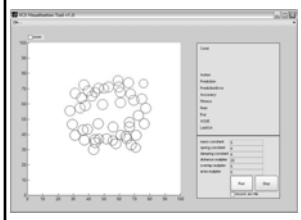


XCS 6 mux final rule set

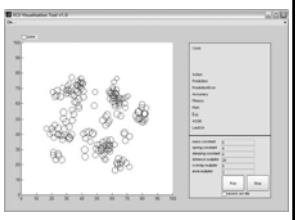


MILCS vs XCS

MILCS 11 mux final rule set

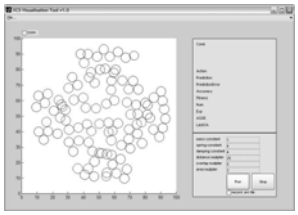


XCS 11 mux final rule set

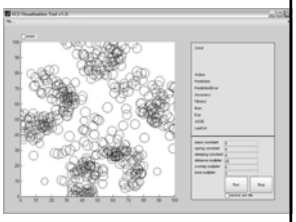


MILCS vs XCS

MILCS 20 mux final rule set



XCS 20 mux final rule set



Future Work

- The system is now fully functioning so we will soon move onto the PSP problems
- and (as time allows) we will investigate multi-step problems.

Conclusion

- We are using an innovative way to develop a learning classifier system which could ultimately aid the research in protein structure prediction problem.
- We are concentrating on generality, accuracy and explanatory power, which are general, scientific goals for machine learning problems.

Future Ideas

- What MILCS does, in a larger sense
  - Performs supervised structural learning by maximizing mutual information between
  - A nonlinear function of inputs, and a signal from the environment
  - In MILCS:
    - Nonlinear function = matching function
    - Signal from environment = current error

Future ideas

- MINN:
  - Nonlinear function = NN hidden node activation function
  - Signal from environment = existing error
- MICLLS (classifier linkage learning):
  - Nonlinear function = parity(?)
  - Signal from environment = fitness function variation
- MRL (reinforcement learning)
  - Nonlinear function = matching
  - Binned Q-values(?)

Questions?