Function Approximation with a Classifier System

Stewart W. Wilson
Prediction Dynamics, Concord, MA 01742
Department of General Engineering
University of Illinois at Urbana-Champaign IL 61801
wilson@prediction-dynamics.com

What is function approximation?

- An approximation $A_{f(x)}[x]$ of $y = f(x)$ is a *process* that, given an x, generates $A_{f(x)}[x] = y^* \approx f(x)$, within some error.

- $f(x)$ may be *unknown* and only available via *samples*.

- $A_{f(x)}[\cdot]$ can be of various kinds: piecewise-constant, piecewise-linear, basis functions (e.g., Fourier series), etc.

Why is function approximation important?

- **Predictive** — of y values for x's not yet seen.

- **Economical** — allows compact representation of high-dimensional functions.

- **Insightful** — may reveal function's structure.

How can a classifier system do function approximation?

- It already does!
- Each classifier continually estimates its payoff.
- In XCS, estimates (called *predictions*) are *accurate* since fitness is based on accuracy.
- In effect, XCS approximates the mapping $\mathsf{X} \times \mathsf{A} \Rightarrow \mathsf{P}$.
- For a single action $a_i \in \mathsf{A}$, the mapping is $\mathsf{X} \times a_i \Rightarrow \mathsf{P}$, a function from input vectors x to scalar payoffs.
- So XCS approximates a separate function for each a_i.
- The purpose is to choose the best action.

What are we doing today?

- Modify XCS for function approximation *per se*.

- Approximate (learn) functions $y = f(x)$, where y is real and x is a vector with integer components $x_1, ..., x_n$.

- As a by-product, uncover an interesting generalization of classifier architecture that may open a way to continuous actions.

What modifications to XCS (to make "XCSF")?

- Adapt the program for integer instead of binary input vectors. I.e., make it like XCSI.

- Restrict to a single (dummy) action.

- Make the payoff prediction directly accessible at the output.

- To approximate $y = f(x)$, let x be the input and y the payoff.

  After sufficient sampling of the input space, the system should, given x, more or less accurately predict the corresponding y.

Modifications for integer input vectors (I.e., like XCSI).

- Assume input $x = (x_1, ..., x_n)$ is a vector of integers.

- Classifier condition is a concatenation of "interval predicates" $int_i = (l_i, u_i)$.

- A match occurs iff $l_i \leq x_i \leq u_i$ for all x_i in x.

- Crossover (two point) as in XCS.

  A crossover point can occur between any two alleles, i.e., within an interval predicate or between predicates.

- Mutation is different.

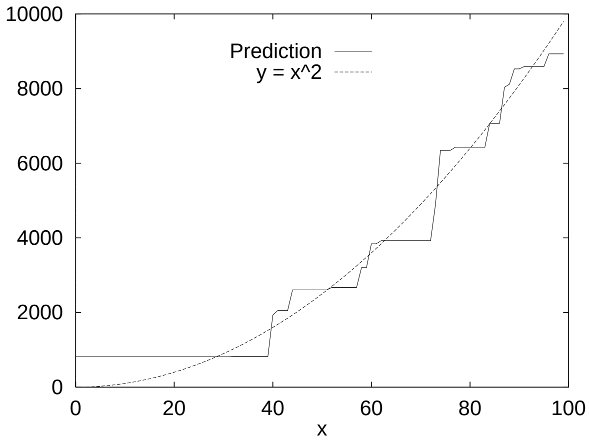
  Mutate an allele by $\pm rand(m_0)$, with m_0 a fixed integer, $rand$ picks randomly from $[0, m_0)$, and the sign is chosen randomly. Resulting values are bounded.

- Condition of a covering classifier is $l_1, u_1, ..., l_n, u_n$ where $l_i = x_i - rand_1(r_0)$, limited by the minimum input value, and $rand_1$ picks randomly from $(0, r_0)$.

  Similarly for u_i.

- Subsumption is similar to XCS, but adapted to interval predicates.

Simplest case: Piecewise-constant approximation.



$\epsilon_0 = 500$

```
                CONDITION           PRED  ERR  FITN NUM

 0. |..................o0| 8992. 283. .796 31
 1. |.................0o.| 7876. 322. .906 32
 2. |..............o00...| 6162. 503. .564 20
 3. |..............000...| 5990. 570. .188  9
 4. |.............o000...| 5988. 569. .028  1
 5. |.............o00o...| 5668. 413. .075  3
 6. |.............o00....| 5534. 281. .090  3
 7. |.............o0o....| 5423. 205. .298 10
 8. |............o000o...| 5053. 524. .002  1
 9. |............o00o....| 4861. 465. .054  5
10. |..........000o......| 3854. 468. .862 30
11. |......o0000.........| 1481. 364. .135  7
12. |......00000.........| 1359. 377. .358 18
13. |000000000o..........|  821. 378. .372 15
14. |00000000o...........|  820. 377. .371 15
```

Can we do piecewise-linear?

- Instead of a constant, let the classifier prediction be a *linear function* $h(x) = w \cdot x'$, where w is a *weight vector* $(w_0, w_1, ..., w_n)$, and x' is the input vector x augmented by a constant x_0, i.e., $x' = (x_0, x_1, ..., x_n)$.

- Over the subspace of X in which it matches, the classifier's prediction $h(x)$ is then a linear approximation to $f(x)$.

- Adapt w according to a version of the delta rule,
$$\Delta w_i = \eta(t - o)x_i,$$
where w_i and x_i are the ith components of w and x', t ("target") is the correct value of y according to $y = f(x)$, o ("output") is the value of $h(x)$, and η is the correction rate.

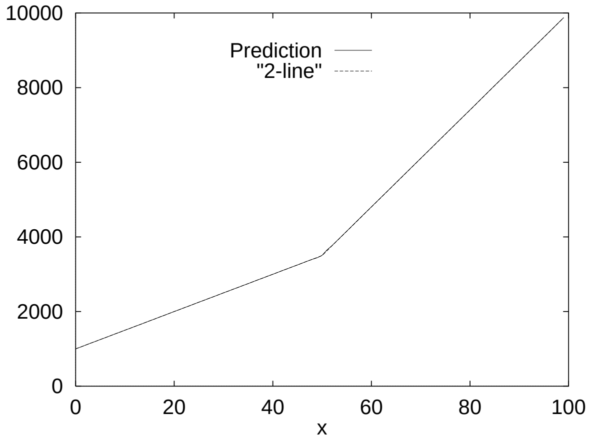
- The rest of XCSF proceeds as usual (updating of classifier parameters other than prediction, GA applied to the classifier conditions).

- Hopefully, the conditions and the weight vectors of all the classifiers will co-adapt to create a good piecewise-linear approximation to $f(x)$.

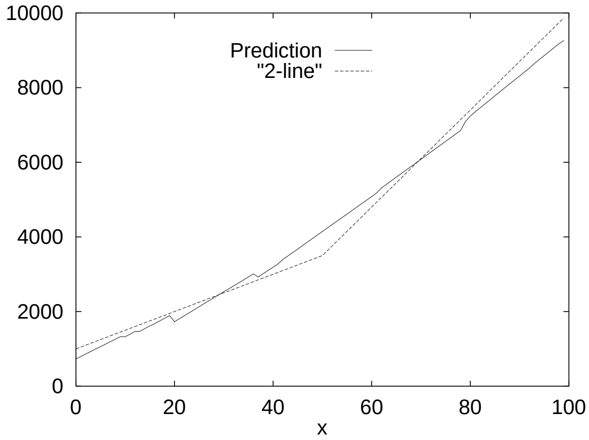
- A few further details were required: slight modification of above delta rule, appropriate choice of x_0, choice of η, weight-vector initialization. Please see paper.

Piecewise-linear approximation to "two-line":

$$
\begin{aligned}
y &= 50x + 1000, \;\; 0 \le x < 50 \\
&= 130x - 3000, \;\; 50 \le x < 100.
\end{aligned}
$$



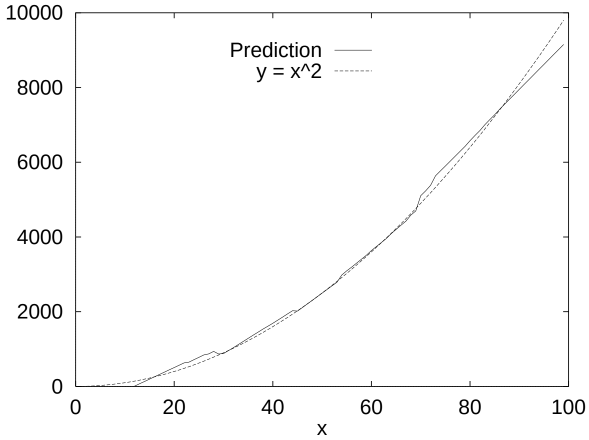
$\epsilon_0 = 10$

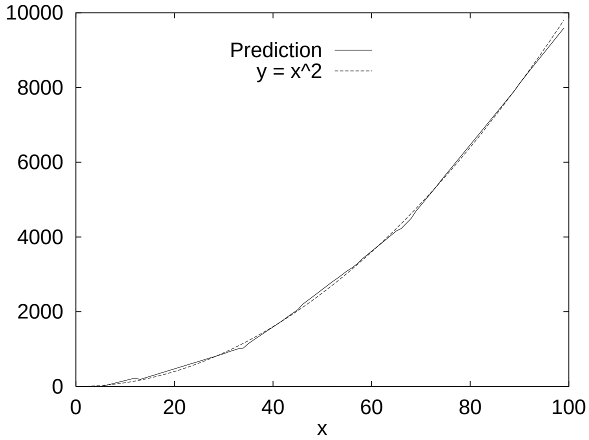


$\epsilon_0 = 500$

```
        CONDITION           PRED   ERR  FITN NUM

0. |.........oOOOOOOOOOO| 4705.  19. .158  62
1. |..........oOOOOOOOOO| 4670.   0. .706 274
2. |..........oOOOOOOOOO| 4670.   0. .044  18
3. |..........OOOOOOOOOO| 4670.   0. .090  29
4. |OOOOOOOOOOo.........| 3057.  16. .005   6
5. |OOOOOOOOOOo.........| 3050.   0. .676 247
6. |OOOOOOOOOo..........| 3050.   0. .377 164
```

Parabola:





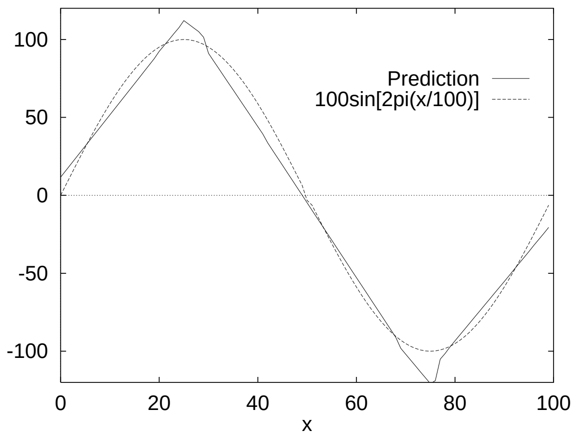
$\epsilon_0 = 500$

```
      CONDITION          PRED  ERR   FITN NUM

0. |.........OOOOOOOOOOO| 9375. 246. .286 151
1. |......OOOOOOOOOOOOOO| 9014. 431. .122  54
2. |.....oOOOOOOOOOOOOOO| 8985. 432. .176  80
3. |.....oOOOOOOOOOOOOOO| 8906. 456. .037  37
4. |....oOOOOOOOOOOOOOOO| 8744. 456. .035  33
5. |OOOOOOOOOOOOOOo.....| 4184. 379. .077  35
6. |OOOOOOOOOOOOOO......| 3756. 247. .258 122
7. |OOOOOOOOOOOOOo......| 3564. 256. .066  32
8. |OOOOOOOOOOOOo.......| 3193. 262. .035  18
9. |OOOOOOOOOOo.........| 2250. 204. .211 125
```

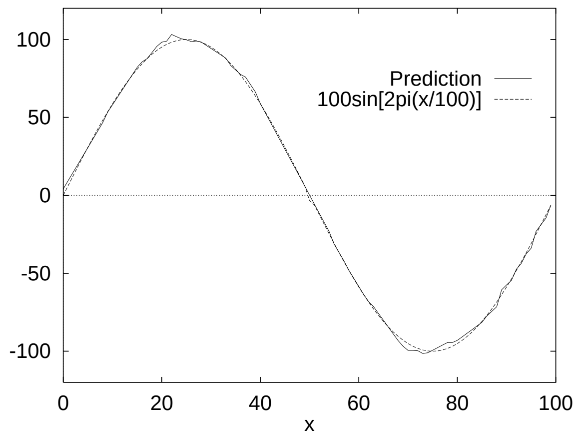
$\epsilon_0 = 100$

```
      CONDITION          PRED  ERR   FITN NUM

0. |.............oOOOOOO| 9591.  77. .823 212
1. |...........oOOOOOO..| 7734.  75. .081  21
2. |..........oOOOOOOO..| 7687.  89. .035   9
3. |......oOOOOOOo......| 4438.  79. .731 184
4. |......oOOOOo........| 3149.  26. .123  41
5. |..oOOOOOOo..........| 1868.  79. .228  39
6. |..oOOOOOo...........| 1621.  61. .072  13
7. |OOOOOOO.............|  967. 104. .607 219
8. |OOOOOOo.............|  749.  85. .074  16
```

Sine:



$\epsilon_0 = 10$



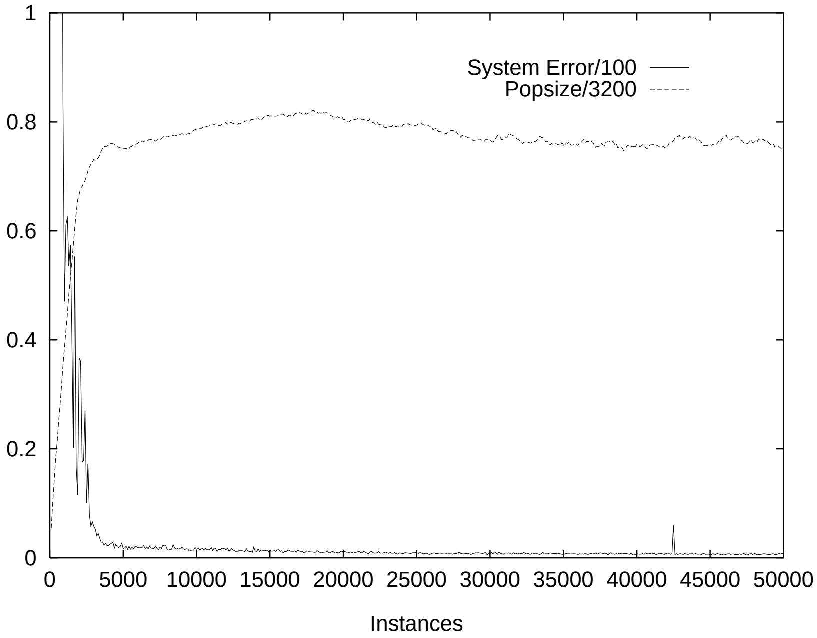
$\epsilon_0 = 2$

	CONDITION	PRED	ERR	FITN	NUM
0.	|OOOOOO..............|	120.	10.	.292	76
1.	|OOOOOo..............|	118.	10.	.083	22
2.	|OOOOO...............|	111.	7.	.572	158
3.	|....OOOOo...........|	66.	5.	.032	15
4.	|...............oOOOO|	-20.	4.	.792	173
5.	|.............oOOOOOO|	-23.	12.	.101	78
6.	|.....OOOOOOOOOOo....|	-130.	8.	.888	234

What about multi-dimensional functions?

Tested on “rms” function:

$y = [(x_1^2 + ... + x_n^2)/n]^{1/2}, 0 \leq x_i < 100$



$\epsilon_0 = 1$ (1% of range)

Summary

- XCSF, derived from XCS, used the prediction mechanism to learn piecewise-linear approximations to functions.

- Tests on simple one-dimensional functions yielded arbitrarily close approximations, according to setting of ϵ_0.

- On a six-dimensional nonlinear function, XCSF rapidly formed a highly accurate approximation, though a large number of classifiers were required.

- Future work should attempt to measure the complexity (classifiers required) on multi-dimensional functions. Probably this relates to the degree of linearity, since $O(1)$ classifiers are needed for linear functions of arbitrary dimensionality.

- Make comparisons with fuzzy classifier systems. Fuzzy computes function using more than one classifier; XCSF in principle requires just one.

- Applications include on-line learning of any mapping from a vector of input values to an output value. Example: financial time-series prediction, where a future value is an approximable function of earlier values in the series.

...Architecture

Traditional classifier

1#00##10 : 1 $\Rightarrow$ 45

<conjunctive predicate>:<fixed action> $\Rightarrow$ <constant prediction>

S-classifier

$(< x_2 \ x_5)$: turn-left $\Rightarrow$ 45

<truth function $t(x)$ >:<fixed action> $\Rightarrow$ <constant prediction>

Generalized Classifier (fixed action)

$(< x_2 \ x_5)$: turn-left $\Rightarrow$ 3.2$(x_5 - x_2)$

$t(x)$:<fixed action> $\Rightarrow$ <prediction function approx. $A_{p(x)}[x]$ >

Generalized Classifier (variable single action)

$(< x_2 \ x_5)$: $(.38 < a \leq .52) \Rightarrow 3.2(x_5 - x_2) + .4a$

$t(x)$: <action restriction $r(a)$ > $\Rightarrow A_{p(x,a)}[x, a]$

Generalized Classifier (action vector)

$(< x_2 \ x_5)$: $(.38 < a_1 \leq .52), (0.0 < a_2 \leq .7) \Rightarrow$
$3.2(x_5 - x_2) + .4a_1 + .23a_2$

$$t(x) \ : \ r(a) \ \Rightarrow \ A_{p(x,a)}[x, a]$$

Good 'ole 6-Multiplexer

- Classifier pair from XCS:

  10##1# : 1 $\Rightarrow$ 1000
  10##1# : 0 $\Rightarrow$ 0

- As Generalized Classifier:

  10##1# : # $\Rightarrow$ 1000a

$t(x)\ :\ r(a) \Rightarrow A_{p(x,a)}[x,a]$

<truth function> : <action restriction> $\Rightarrow$
<prediction function approximation>

"Given that the input satisfies $t(x)$ and the action is within $r(a)$, the predicted payoff is approximated by $A_{p(x,a)}[x,a]$."