

An Agent Architecture for Predicting Protein Secondary Structures

G. Armano, (*) L. Milanesi, (^) and A. Orro (*)

(*) *DIEE - University of Cagliari, Cagliari, Italy*

email: {armano,orro}@diee.unica.it

(^) *ITB - CNR, Milano, Italy*

email: milanesi@itba.mi.cnr.it

Outline of the Talk

- ▶ *Introduction*
- ▶ *Focusing on the Problem ...*
- ▶ *The Proposed Solution (Conceptual Level)*
- ▶ *The Proposed Solution (Architectural Level)*
- ▶ *The Proposed Solution (Design Level)*
- ▶ *Experimental Results*
- ▶ *Concluding Remarks*

Inputs Encoding

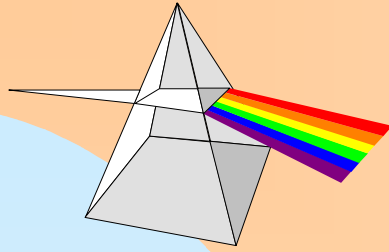
?

Notes on XCSs

?

Notes on NXCSs

?



Introduction

...

Why Predicting Secondary Structures ?

- *Finding the actual labeling through existing techniques may become too expensive if performed on a large scale*
- *Predicting the actual labeling is less expensive ...*

Existing Methods for Predicting Secondary Structures

- *Purely syntactic methods*
 - ◆ Based on the analysis of the primary structure performed using grammar-based and / or machine learning approaches
- *Comparative Modeling*
- *Fold Recognition*
- *Ab-initio Methods*

Existing Methods for Predicting Secondary Structures

- *Purely syntactic methods*
- *Comparative Modeling*
 - ◆ Based on the similarity between test sequences and the ones available in structural databases
- *Fold Recognition*
- *Ab-initio Methods*

Existing Methods for Predicting Secondary Structures

- *Purely syntactic methods*
- *Comparative Modeling*
- *Fold Recognition*
 - ◆ Based on structural templates whose matching sequences have a known spatial folding
- *Ab-initio Methods*

Existing Methods for Predicting Secondary Structures

- *Purely syntactic methods*
- *Comparative Modeling*
- *Fold Recognition*
- *Ab-initio Methods*
 - ◆ Use a lattice model to predict the structure by minimizing an energy function



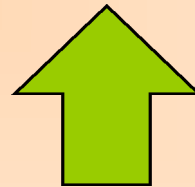
Focusing on the Problem ...

Let's get to the point ...

Focusing on the Problem ...

- *Given an amino acidic sequence, predict its secondary structure (α -helix, β -sheet, or coil)*

a	a	a	a	a	-	b	b	b	b	b	-	c	c	-	b	b	b	b	b
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---



A	B	C	D	A	K	L	H	I	I	B	L	M	S	R	D	F	D	S	A
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Using a Global Model ?

➤ *Global models ...*

- ◆ Often rely on a "state-based" approach (e.g., HMMs, Recurrent ANNs)
- ◆ Must be trained on large input sequences, to (hopefully) be able to identify the underlying system
- ◆ Lack of generalization ability (in terms of underfitting)

Using Local Models ?

➤ *Local models ...*

- ◆ Do not require a “state-based” approach (they can be “context-based”)
- ◆ Do not require to be trained on large input sequences
- ◆ Lack of generalization ability (in terms of overfitting)

Context- vs. State-Based Approach

- *Contexts usually apply to classification tasks*
 - ◆ e.g., to classify a pixel in a digital image, a limited window of surrounding pixels can be taken into account
- *Contexts may be summarized by suitable metrics (thus reducing the complexity of the learning task)*
 - ◆ e.g., one or more filters can be applied to a given window of pixels. The results summarize the relevant features of the window

Context-Based Classification

- *Why adopting a “context-based” approach also for prediction tasks ?*
 - ◆ Context identification can be successfully exploited to split the input domain
 - ◆ Regions –in the case of secondary structures prediction– are input subsequences that show similar characteristics
 - ◆ The “similarity” criteria act as context selectors



The Proposed Solution (Conceptual Level)

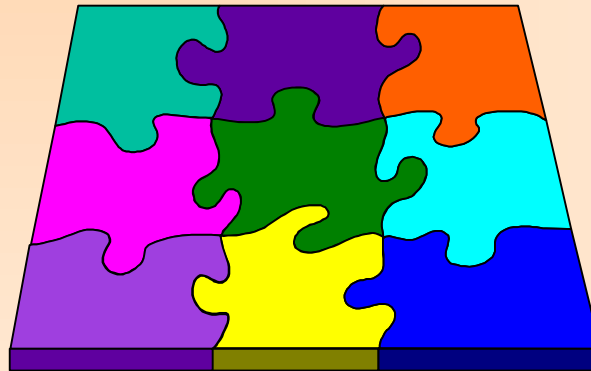
Syntactic sugar? No, thanks.

Solution (Conceptual Level)

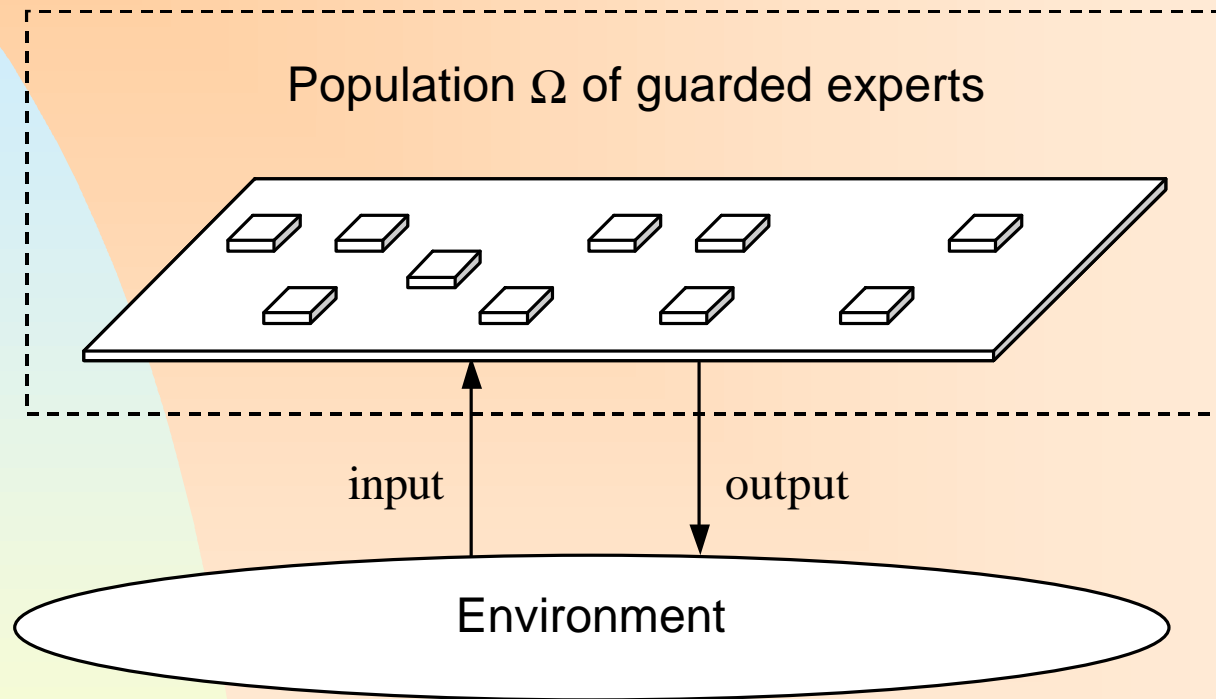
- *Using local models (context-based approach)*
- *Devising a population of experts*
- *Each expert participates to the prediction process only on a (usually small) subset of the input sequences*

Underlying Assumption

- *Splitting the input space allows to make it easier the classification task, in a multiple-experts perspective*



Zoom Out ...



Zoom In ...

- *Micro-Architecture ...*

- ◆ Defining guarded experts

- *Macro-Architecture ...*

- ◆ Handling a population of guarded experts

Micro-Architecture

- *A guarded expert is a triple $\langle g, h, w \rangle$ where:*
 - ◆ h is a total or partial function that maps an input space (I) to an output space (O)
 - ◆ g is a boolean function devoted to control the activation of h (i.e., g is a “guard” that identifies a subset of inputs for which the mapping exists)
 - ◆ w is a weighting function, which identifies the strength of the expert

Micro-Architecture

- II

➤ *In symbols:*

$G = \langle g, h, w \rangle = \textit{guarded expert}$

$G : I_g \rightarrow O, D(G) = I_g$

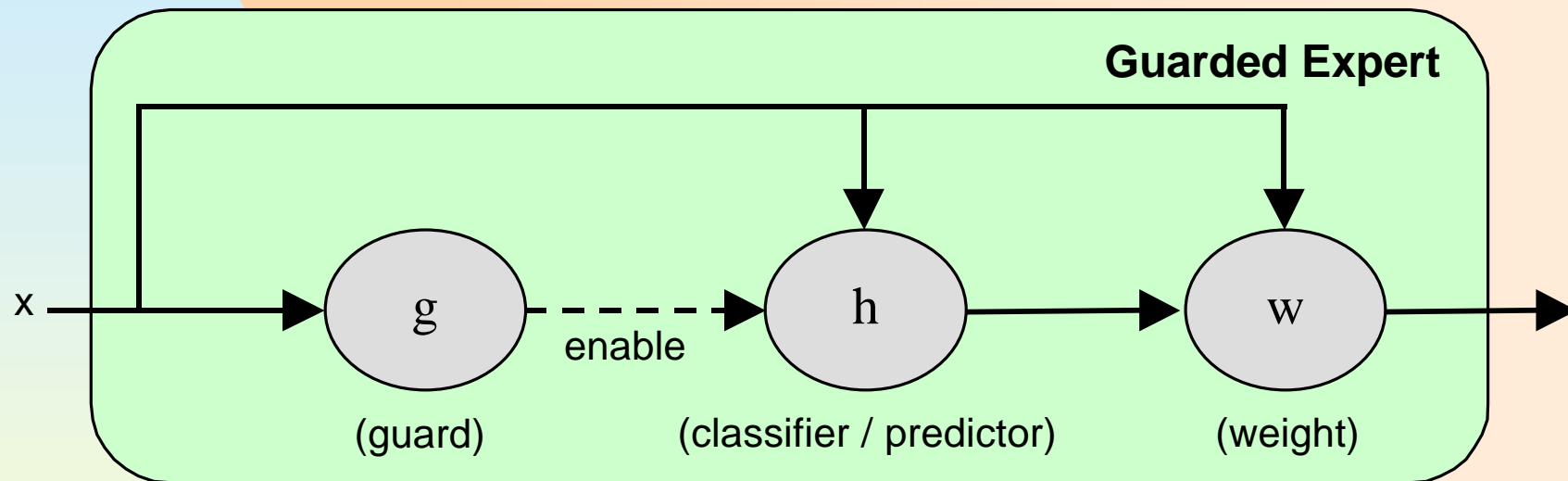
$G(x) = \text{if } g(x) \text{ then } w(x) \diamond h(x) \text{ else } \perp$

where

- ◆ $g = \text{boolean guard}$ $I_g = \{ x \in I \mid g(x) = \textit{true} \}$
- ◆ $h = \text{total or partial function}$ $I_g \subseteq I_h \subseteq I$
- ◆ $w = \text{weighting function}$

Micro-Architecture

- III



Macro-Architecture

- *Guarded experts can be arranged into a population ...*

$$\mathbf{W} = \left\{ \mathbf{G}_i \mid \mathbf{G}_i = \langle g_i, h_i, w_i \rangle, i = 1, 2, \dots, n \right\}$$

- *Domain of a population of guarded experts*

$$D(\mathbf{W}) = \bigcup_{\mathbf{G}_i \in \mathbf{W}} D(\mathbf{G}_i) = \left\{ x \in I \mid \exists i = 1, 2, \dots, n \text{ s.t. } g_i(x) = \text{true} \right\}$$

Macro-Architecture

- II

- *To handle a population of guarded experts several decision must be taken:*
 - ◆ Training strategy and technique ?
 - ◆ Region Splitting Criteria (boundaries and overlapping) ?
 - ◆ Experts Selection Mechanism (usually required) ?
 - ◆ Outputs Blending Mechanism (usually required) ?
 - ◆ Voting Policy (usually required) ?



The Proposed Solution (Architectural Level)

Experimenting multiple experts technology ...

A Hybrid Architecture for Predicting Secondary Structures

➤ *Micro-Architecture ...*

- ◆ Devising a hybrid guarded expert using eXtended Classifier Systems (XCSs) and Artificial Neural Networks (ANNs)

➤ *Macro-Architecture ...*

- ◆ Implementing the population of experts as a society of agents
- ◆ Using simple coordination policies

XCS » reinforcement learning + genetic algorithms

Micro-Architecture: NXCS Experts

- *A “Neural XCS” expert (NXCS expert for short) is a Guarded Expert where ...*
 - ◆ g = an XCS-like classifier (maps its inputs to bool)
 - ◆ h = an ANN (suitably customized)
 - ◆ w = expert's fitness
- *In case of multiple outputs ...*
 - ◆ $h = \langle h_1, h_2, \dots \rangle$

Macro-Architecture: Handling a Population of NXCS Experts

- *Training strategy: batch*
- *Training technique: Darwinian selection (XCS guards) + backpropagation (ANNs)*
- *Region splitting: hard, with overlapping*
- *Experts' Selection: match-set formation*
- *Outputs blending: fitness-weighted averaging*
- *Voting: plurality rule*

Selection, Outputs Blending, and Voting

$$O(x) = \text{choose}(\text{combine}(\text{select}(W)))|_x$$

where ...

- ◆ Ω denotes the population of experts
- ◆ x denotes the current input
- ◆ $\text{select}()$ creates the match-set
- ◆ $\text{combine}()$ blends outputs of all experts that belong to the match-set
- ◆ $\text{choose}()$ enforces the adopted voting policy

Selection, Outputs Blending, and Voting - II

➤ Selection (match-set formation)

◆ $select(\mathbf{W})|_x \rightarrow \{e_1, e_2, \dots, e_L\} = M$ Match Set

➤ Outputs blending (fitness-weighted average)

◆ $combine(M)|_x \rightarrow \langle o_a, o_b, o_c \rangle = O(x)$ Overall Output

◆
$$o_k = \frac{\sum_{e \in M} f_e \cdot h_{ek}(x)}{\sum_{e \in M} f_e} \quad k \in \{ \alpha, \beta, c \}$$

➤ Voting (plurality rule)

◆ $choose(O) \rightarrow k^* = \arg \max_{k \in \{a, b, c\}} o_k$ Selected Output

Inputs to a Genetic Guard

- *A pattern of physico-chemical properties is generated, to be matched with a moving window of residuals*

Inputs to a Genetic Guard

- *A pattern of physico-chemical properties is generated, to be matched with a moving window of residuals*

Inputs to a Genetic Guard

➤ *Acidity*

- ◆ Acid (+)
- ◆ Basic (-)
- ◆ Neutral (=)


➤ *Hydrophobicity*

- ◆ Hydrophobic
- ◆ Hydrophilic

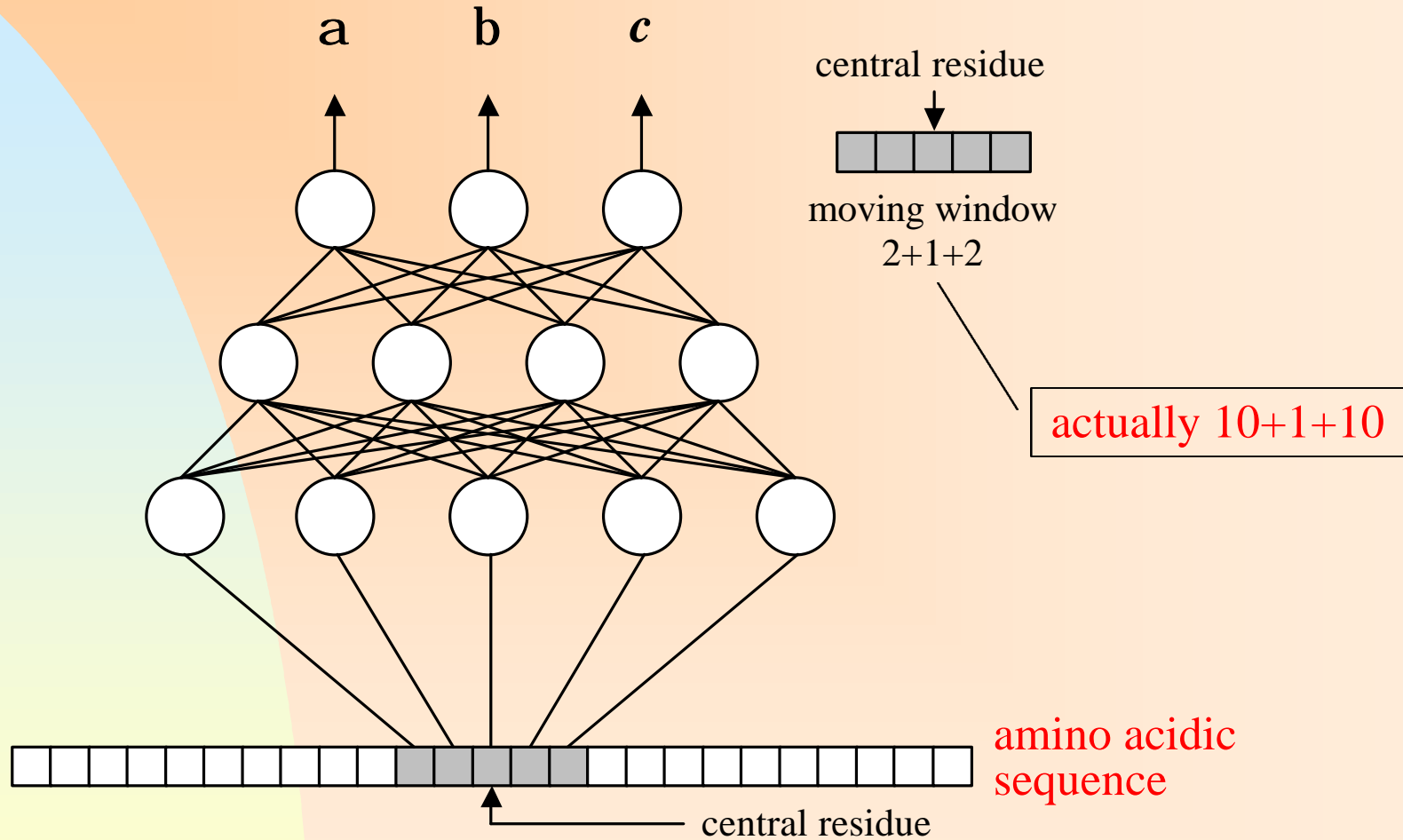
➤ *Polarity*

- ◆ Polar
- ◆ Aromatic
- ◆ Aliphatic

*o-chemical properties is
atched with a moving*



I/O of a Neural Predictor



I/O of a Neural Predictor

- II

➤ *Inputs*

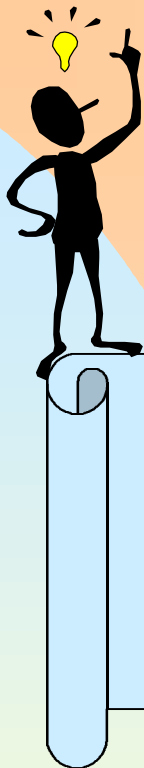
- ◆ A moving window of 21 = 10+1+10 residues

➤ *Outputs*

- ◆ Three separate outputs: α -helix, β -sheet, coil
- ◆ Currently, no rejection option



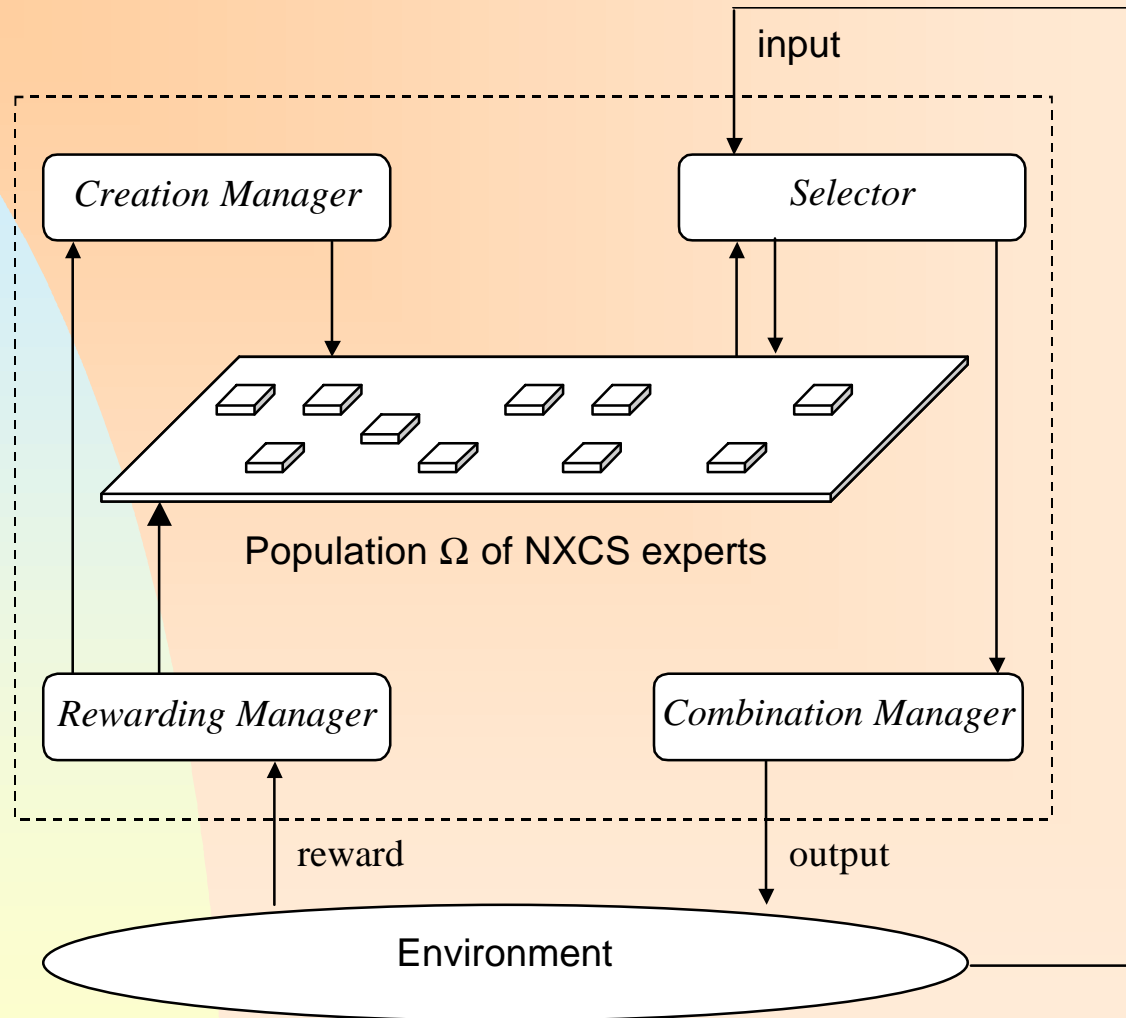
central
residue



The Proposed Solution (Design Level)

Experimenting agent-oriented technology ...

Agent-Oriented Implementation of the Proposed Architecture



Implementation of Agents

- *Coordination agents:*
 - ◆ Selector
 - ◆ Combination Manager
 - ◆ Rewarding Manager
 - ◆ Creation Manager
- *NXCS experts population*
 - ◆ A “society” of NXCS experts

Implementation of Agents

- *Both coordination agents and NXCS experts are implemented as active objects*
- *Each active object embodies two interacting subsystems*
 - ◆ a communication subsystem (entrusted with I/O)
 - ◆ an engine (entrusted with operations execution)

Implementation of Agents : the Communication Subsystem

- *Equipped with an input queue, hosting incoming messages*
- *Output messages are not queued.*
- *Synchronous and asynchronous message passing*

Implementation of Agents : the Engine

- *Non-preemptive thread scheduler (writers are mutually exclusive, whereas readers are not)*
- *Reactive behavior is hand-coded*
 - ◆ The current task can be suspended and possibly resumed after serving the interrupt (no goal stacking, though!)
- *Proactive behavior is hand-coded*
 - ◆ The current goal is selected among a predefined set of "behaviors"

Enforcing Coordination Policies

➤ *Selector*

- ◆ Once informed that there is an input to process, after a preliminary check, aimed at forming the match set, the selector interacts with the platform of NXCS experts
- ◆ The match set (together with the current input) is forwarded to the Combination Manager

Enforcing Coordination Policies

➤ *Combination Manager*

- ◆ After receiving information about the match set and the current input, the combination manager produces the final output by interacting with all NXCS experts that belong to the match set
- ◆ In the special case of an empty match set, the combination manager either informs the creation manager that a new expert, able to cover the current input, must be created (training status) or outputs a “void” prediction (test status)

Enforcing Coordination Policies

➤ *Rewarding Manager*

- ◆ Informs all NXCS experts belonging to the match set that they should update their fitness, according to the reward obtained by the environment (training status)

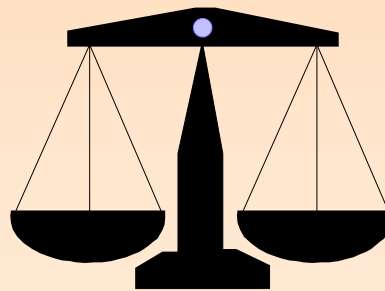
Enforcing Coordination Policies

➤ *Creation Manager*

- ◆ Handles experts' creation, being able to perform covering, crossover, and mutation operations
- ◆ It is also responsible for experts deletion, with a probability inversely proportional the fitness of each expert



Experimental Results



Experiments

- *About 126 training sequences have been taken into account*
 - ◆ RS126 dataset
- *About 396 test sequences have been taken into account*
 - ◆ CB396 dataset

Metrics Adopted to Assess the Prediction Capability

➤ Q_3

- ◆ Percent of residuals correctly predicted vs. the overall number of residuals

➤ *SOV (Segment Overlap)*

- ◆ Measures, for each conformational status, the overlapping between predicted and correct elements

Comparative Results

		<u>RS126</u> <u>protein set</u>			<u>CB396</u> <u>protein set</u>	
<u>Method</u>		<u>Q3</u>	<u>SOV</u>		<u>Q3</u>	<u>SOV</u>
PHD		73.5	73.5		71.9	75.3
DSC		71.1	71.6		68.4	72.0
PREDATOR		70.3	69.9		68.6	69.8
NNSSP		72.7	70.6		71.4	71.3
CONSENSUS		74.8	74.5		72.9	75.4
MASSP		71.4	68.9		69.1	70.5



Concluding Remarks

Conclusions

- *To tackle the problem of predicting amino acidic secondary structures ...*
 - ◆ We experimented the multiple experts technology
 - ◆ Their implementation followed the guidelines of agent-oriented programming
- *Results are encouraging, although the focus was on experimenting new technologies rather than on improving other systems' performances*

Future Work

- *The next release of the system, able to implement a subset of FIPA ACL is currently under way*
- *The final architecture will be a society of heterogeneous agents headed at predicting the secondary structure of amino acidic sequences*

Well...

Questions ?



Inputs Encoding

Inputs Encoding

- *Amino acids are encoded according to the following physico-chemical properties*
 - ◆ Acidity
 - ◆ Hydrophobicity
 - ◆ Polarity

Inputs Encoding

- II

➤ *Acidity*

- ◆ Acid (+) 001
- ◆ Basic (-) 010
- ◆ Neutral (=) 100

➤ *Hydrophobicity*

- ◆ Hydrophobic 01
- ◆ Hydrophilic 10

➤ *Polarity*

- ◆ Polar 001
- ◆ Aromatic 010
- ◆ Aliphatic 100
- ◆ Other 000

Inputs Encoding

- III

A	100	01	100	(a)	L	100	01	100	(a)
R	010	10	000	(b)	K	010	10	000	(b)
N	100	10	001	(c)	M	100	01	001	(f)
D	001	10	000	(d)	F	100	01	010	(g)
C	100	10	001	(c)	P	100	01	100	(a)
Q	100	10	001	(c)	S	100	10	001	(c)
E	001	10	000	(d)	T	100	10	001	(c)
G	100	10	100	(e)	W	100	01	010	(g)
H	010	10	000	(b)	Y	100	10	010	(h)
I	100	01	100	(a)	V	100	01	100	(a)

(a) / 5, (b) / 3, (c) / 5, (d) / 2, (e) / 1, (f) / 1, (g) / 2, (h) / 1



Notes on NXCS

NXCS- TYPICAL TRAINING LOOP

0. Start with an empty or existing population of experts.
1. Given an input x , build the match set **M**.
2. If **M** is empty, generate a new expert able to cover x .
3. Select an action a^* according to a suitable policy (typically, a fitness-weighted majority / plurality rule).
4. Update p , e , and f of each classifier in **M**.
5. When needed, generate a new pair of experts using genetic operators (crossover and mutation). Insert the pair of predictors in the population.
6. When needed, delete a pair of experts from the population.
7. Go to Step 1.

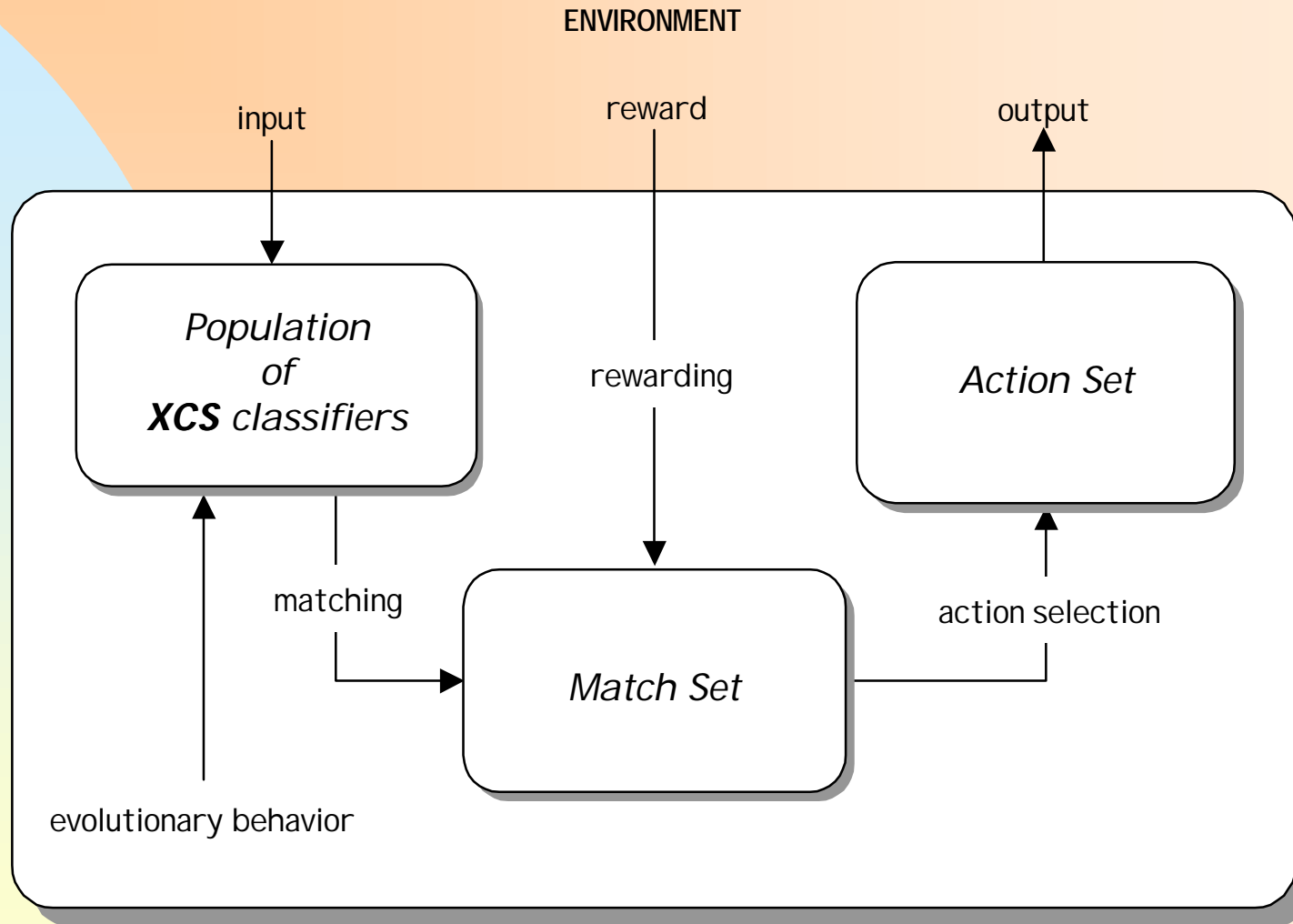


Notes on XCSs

XCS Key Concepts

- *An XCS is an evolutionary learning system consisting of the following components ...*
 - ◆ performance
 - ◆ reinforcement
 - ◆ discovery

Basic XCS Architecture



XCS Classifiers - Main Parameters

➤ *The most important parameters of an XCS classifier are:*

◆ Prediction (p)

$$p_{t+1} = \mathbf{l}_p(p_t, r)$$

◆ Prediction Error (e)

$$\mathbf{e}_{t+1} = \mathbf{l}_e(\mathbf{e}_t, r, p_{t+1})$$

◆ Fitness (f)

$$f_{t+1} = \mathbf{l}_f(f_t, r, k'_{t+1})$$

◆ Relative Accuracy (k')

$$k'_{t+1} = \mathbf{l}'_k(k_{t+1}, \bar{\mathbf{K}}_{M_{a^*}})$$

◆ Accuracy (k)

$$k = \mathbf{l}_k(\mathbf{e}_{t+1})$$

XCS Classifiers - Main Parameters

Operation	Formula	Where ...
Prediction Update (single step)	$p_{t+1} = p_t + \mathbf{b} \cdot (r - p_t)$	r is the actual reward obtained by the system from the environment.
Prediction Error Update	$\mathbf{e}_{t+1} = \mathbf{e}_t + \mathbf{b} \cdot (p_{t+1} - r - \mathbf{e}_t)$	
Fitness Update	$f_{t+1} = f_t + \mathbf{b} \cdot (k' - f_t)$	k' is the relative accuracy of a classifier.
Relative Accuracy	$k' = \frac{k}{\sum_{c \in M_{a^*}} k_c}$	The accuracy of a classifier is normalized over the action set corresponding to the selected action a^* .
Accuracy	$k = \exp \left[\ln(\mathbf{a}) \cdot \frac{\mathbf{e} - \mathbf{e}_0}{\mathbf{e}_0} \right]$	α = accuracy fall-off ε_0 = accuracy threshold (note that $k = \alpha$ when $\varepsilon = 2\varepsilon_0$).

XCS- TYPICAL TRAINING LOOP

0. Start with an empty, or existing, population of classifiers.
1. Given an input x , build the *match set* **M**.
2. If **M** is empty, generate a new classifier able to cover x .
3. Select an action a^* according to a suitable strategy (typically, a fitness weighted majority rule).
4. Update p , e , and f of each classifier that supports a^* .
5. When needed, generate a new pair of classifiers using standard genetic operators (crossover and mutation). Insert the pair of classifiers in the population.
6. When needed, a pair of classifiers is deleted.
7. Go to Step 1.